



User Guide

AWS CloudHSM



AWS CloudHSM: User Guide

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

Die Handelsmarken und Handelsaufmachung von Amazon dürfen nicht in einer Weise in Verbindung mit nicht von Amazon stammenden Produkten oder Services verwendet werden, durch die Kunden irregeführt werden könnten oder Amazon in schlechtem Licht dargestellt oder diskreditiert werden könnte. Alle anderen Handelsmarken, die nicht Eigentum von Amazon sind, gehören den jeweiligen Besitzern, die möglicherweise zu Amazon gehören oder nicht, mit Amazon verbunden sind oder von Amazon gesponsert werden.

Table of Contents

Was ist AWS CloudHSM?	1
Anwendungsfälle	3
Funktionsweise	5
Cluster	5
Benutzer in AWS CloudHSM	6
Schlüssel rein AWS CloudHSM	7
Kunde SDKs	8
Sicherungen	9
Unterstützte Regionen für AWS CloudHSM	10
Preisgestaltung für AWS CloudHSM	10
Erste Schritte	11
Erstellen von IAM-Administratoren	11
Erstellen von IAM-Benutzern und -Administratorgruppen	12
Erstellen einer VPC	14
Erstellen eines Clusters	15
Überprüfen der Cluster-Sicherheitsgruppe	20
Einen EC2-Client starten	21
Konfigurieren Sie EC2-Instance-Sicherheitsgruppen	24
Schritt 1. Ändern der Standardsicherheitsgruppe	24
Schritt 2. Connect die Amazon EC2 EC2-Instance mit dem Cluster AWS CloudHSM	25
Erstellen eines HSM	26
Überprüfen der HSM-Identität (optional)	28
Schritt 1. Zertifikate aus dem HSM laden	30
Schritt 2. Die Stammzertifikate laden	33
Schritt 3. Zertifikatketten prüfen	34
Schritt 4. Extrahieren und Vergleichen der öffentlichen Schlüssel	35
Initialisieren des Clusters	36
-Übersicht	36
Schritt 1. Anfordern der Cluster-CSR	37
Schritt 2. Erstellen Sie einen privaten Schlüssel für Ihre Root-CA	39
Schritt 3. Signieren der CSR	40
Schritt 4. Initialisieren des Clusters	50
Installieren Sie CloudHSM-CLI	53
Aktivieren des Clusters	57

mTLS einrichten (empfohlen)	60
Schritt 1. Erstellen und registrieren Sie einen Vertrauensanker auf dem HSM	61
Schritt 2. Aktivieren Sie mTLS für AWS CloudHSM	65
Schritt 3. Stellen Sie die mTLS-Durchsetzung ein für AWS CloudHSM	70
Schlüssel erstellen und verwenden in AWS CloudHSM	72
Best Practices	74
Clusterverwaltung	74
Skalieren Sie Ihren Cluster, um den Spitzenverkehr zu bewältigen	74
Richten Sie Ihren Cluster auf Hochverfügbarkeit aus	74
Halten Sie mindestens drei Schlüssel bereit HSMs , um die Haltbarkeit neu generierter Schlüssel zu gewährleisten	75
Sicherer Zugriff auf Ihren Cluster	75
Senken Sie die Kosten, indem Sie sie an Ihre Bedürfnisse anpassen	75
Benutzerverwaltung	76
Schützen Sie die Anmeldeinformationen Ihrer HSM-Benutzer	76
Haben Sie mindestens zwei Administratoren, um eine Aussperrung zu verhindern	76
Aktivieren Sie das Quorum für alle Benutzerverwaltungsvorgänge	77
Erstellen Sie mehrere Krypto-Benutzer, von denen jeder über eingeschränkte Berechtigungen verfügt	77
Schlüsselverwaltung	77
Wählen Sie den richtigen Schlüsseltyp	78
Verwalten Sie wichtige Speicherlimits	78
Verwaltung und Sicherung der Schlüsselverpackung	79
Anwendungsintegration	79
Führen Sie ein Bootstrap für Ihr Client-SDK durch	79
Authentifizieren Sie sich, um Operationen auszuführen	80
Effektives Verwalten von Schlüsseln in Ihrer Anwendung	81
Verwenden Sie Multithreading	82
Gehen Sie mit Drosselungsfehlern um	82
Integrieren Sie Wiederholungsversuche bei Clustervorgängen	82
Implementieren Sie Strategien für die Notfallwiederherstellung	83
Gestaffeln Sie Ihre Anwendungsbereitstellung	83
Überwachen	84
Überwachen Sie die Client-Protokolle	84
Überwachen Sie die Überwachungsprotokolle	84
Überwachen AWS CloudTrail	85

Überwachen Sie CloudWatch Amazon-Metriken	85
Cluster	86
Clusterarchitektur	86
Synchronisierung von Clustern	88
Hohe Verfügbarkeit und Load Balancing von Clustern	89
Cluster-Modi	90
HSM-Typen	91
Verbinden mit dem Cluster	93
Platzieren Sie das ausstellende Zertifikat auf jeder EC2-Instance	93
Geben Sie den Speicherort des ausstellenden Zertifikats an	93
Bootstrap für das Client-SDK	96
Skalierung HSMs	100
Hinzufügen eines HSM	100
Entfernen eines HSM	103
Löschen eines Clusters	104
Cluster aus Sicherungen erstellen	105
Cluster aus Sicherungen erstellen (Konsole)	105
Cluster aus Sicherungen erstellen (AWS CLI)	106
Erstellen Sie Cluster aus Backups (API)AWS CloudHSM	107
Migration von HSM-Clustertypen	107
Migration von hsm1.medium zu hsm2m.medium	108
HSM-Benutzer	117
Benutzerverwaltung mit CloudHSM CLI	117
Voraussetzungen	118
Benutzertypen	120
Tabelle mit den Berechtigungen	121
Admin erstellen	124
Erstellen CUs	125
Auflisten aller Benutzer	126
Passwörter ändern	126
Löschen von Benutzern	128
Benutzer-MFA verwalten	129
Quorum-Authentifizierung verwalten (M von N)	145
Benutzerverwaltung mit CMU	170
Voraussetzungen	171
Benutzertypen	175

Tabelle mit den Berechtigungen	177
Benutzer erstellen	179
Auflisten aller Benutzer	182
Passwörter ändern	184
Löschen von Benutzern	187
2FA für Benutzer verwalten	188
Verwendung von CMU zur Verwaltung der Quorum-Authentifizierung	198
Schlüssel	219
Schlüsselsynchronisation und Haltbarkeit	219
Konzepte	220
Grundlegendes zur Schlüsselsynchronisierung	221
Ändern Sie die Einstellungen für die Haltbarkeit von Client-Schlüsseln	222
Synchronisieren von Schlüsseln zwischen geklonten Clustern	229
AES Key Wrapping	229
Unterstützte Algorithmen	229
Verwenden Sie den AES-Schlüsselumbruch AWS CloudHSM	231
Vertrauenswürdige Schlüssel	233
Grundlegendes zu vertrauenswürdigen Schlüsseln	233
Vertrauenswürdige Schlüsselattribute	234
Wie verwendet man vertrauenswürdige Schlüssel, um Datenschlüssel zu umschließen	235
Wie entpacke ich einen Datenschlüssel mit einem vertrauenswürdigen Schlüssel	238
Schlüsselverwaltung mit CloudHSM CLI	239
Schlüssel generieren	239
Löschen von Schlüsseln	247
Freigabe und Aufheben der Freigabe von Schlüsseln	249
Nach Schlüsseln filtern	257
Markieren Sie einen Schlüssel als vertrauenswürdig	264
Verwalten Sie die Quorum-Authentifizierung (M von N)	265
Schlüsselverwaltung mit KMU	287
Schlüssel generieren	288
Schlüssel importieren	290
Schlüssel exportieren	292
Löschen von Schlüsseln	295
Freigabe und Aufheben der Freigabe von Schlüsseln	295
Markieren Sie einen Schlüssel als vertrauenswürdig	296
Sicherungen von Clustern	297

Arbeiten mit Backups	297
Abgelaufene Schlüssel oder inaktive Benutzer werden entfernt	298
Notfallwiederherstellung erwägen	298
Sicherungen löschen	298
Backups wiederherstellen	300
Konfigurieren Sie die Aufbewahrung von Backups	302
Verwaltete Aufbewahrung von Backups	302
Regionsübergreifendes Kopieren von Sicherungen	306
Sicherungen in verschiedene Regionen kopieren (Konsole)	307
Sicherungen in verschiedene Regionen kopieren (AWS CLI)	307
Backups in verschiedene Regionen kopieren (AWS CloudHSM API)	307
Arbeiten mit gemeinsam genutzten Backups	308
Voraussetzungen für die gemeinsame Nutzung von Backups	308
Ein Backup teilen	309
Die gemeinsame Nutzung eines gemeinsam genutzten Backups rückgängig machen	313
Identifizieren eines gemeinsam genutzten Backups	313
Berechtigungen für gemeinsam genutzte Backups	314
Fakturierung und Messung	314
Geklonte Cluster	315
Verwandte Themen	315
Markieren von Ressourcen	316
Fügen Sie Stichwörter hinzu oder aktualisieren Sie sie	316
Auflisten von Tags	318
Tags entfernen	318
Befehlszeilen-Tools	320
Configure-Tool	321
Configure-Tool für das Client-SDK 5	322
Configure-Tool für das Client-SDK 3	356
CloudHSM-CLI	365
Unterstützte Plattformen	366
Erste Schritte	367
Befehlsmodi	375
Schlüsselattribute	377
Erweiterte -Konfigurationen	385
Referenz	392
AWS CloudHSM Verwaltungsdienstprogramm	653

Unterstützte Plattformen	653
Erste Schritte	654
Installieren des Clients (Linux)	659
Installieren des Clients (Windows)	662
Referenz	663
Schlüsselverwaltungsdienstprogramm	726
Erste Schritte	727
Installieren des Clients (Linux)	731
Installieren des Clients (Windows)	734
Referenz	735
Kunde SDKs	869
Überprüfen Sie Ihre Version	870
Vergleichen Sie die Komponentenunterstützung	872
PKCS-#11-Bibliothek	873
CloudHSM-Verwaltungsdienstprogramm (CMU)	873
Schlüsselverwaltungsdienstprogramm (KMU)	873
JCE-Anbieter	873
OpenSSL Dynamic Engine	873
Schlüsselspeicheranbieter (KSP)	874
Migration zum neuesten SDK	874
Migrieren Sie die PKCS #11 -Bibliothek	875
Migrieren Sie die OpenSSL Dynamic Engine	880
Migrieren Sie den Key Storage Provider (KSP)	883
Migrieren Sie den JCE-Anbieter	891
Client-SDK 5	905
Vorteile des neuesten SDK	905
Unterstützte Plattformen	907
PKCS-#11-Bibliothek	909
OpenSSL Dynamic Engine	963
OpenSSL-Anbieter	971
Schlüsselspeicheranbieter (KSP)	978
JCE-Anbieter	1011
Frühere Versionen	1053
Aktualisieren Sie das Client-SDK 3	1054
Unterstützte Plattformen	1063
PKCS-#11-Bibliothek	1065

OpenSSL Dynamic Engine	1110
JCE-Anbieter	1114
KSP- und CNG-Anbieter	1146
Integrieren von Drittanbieter-Anwendungen	1160
SSL/TLS-Auslagerung	1160
Funktionsweise	1161
Offload auf Linux mit OpenSSL Engine	1163
Offload auf Linux mit OpenSSL Provider	1208
Offload auf Linux mit JSSE	1239
Auslagern unter Windows	1251
Load Balancer hinzufügen (optional)	1267
Windows Server-CA	1274
Client-SDK 5 mit Windows Server-CA	1274
Client-SDK 3 mit Windows Server-CA	1280
Oracle Database-Verschlüsselung	1286
Einrichten der Voraussetzungen	1288
Schritt 2: Aktualisieren Sie die Oracle-Datenbankkonfiguration	1289
Schritt 3: Generieren Sie den Oracle TDE-Master-Verschlüsselungsschlüssel	1290
Microsoft SignTool	1292
Client-SDK 5 mit Microsoft SignTool	1292
Client-SDK 3 mit Microsoft SignTool	1297
Java Keytool und Jarsigner	1302
Client SDK 5 mit Java Keytool und Jarsigner	1302
Client SDK 3 mit Java Keytool und Jarsigner	1315
Tool zur Generierung und Bearbeitung von Microsoft-Manifests	1332
Schritt 1: Einrichten der Voraussetzungen	1333
Schritt 2: Erstellen Sie ein Signaturzertifikat	1334
Schritt 3: Signieren Sie eine Datei	1335
Weitere Integrationen von Drittanbietern	1336
Überwachen	1338
Client-SDK-Protokolle	1338
Protokollierung im Client-SDK 5	1339
Protokollierung im Client-SDK 3	1340
AWS CloudTrail	1343
AWS CloudHSM Informationen in CloudTrail	1343
AWS CloudHSM Logdateieinträge verstehen	1344

Prüfungsprotokolle	1346
Funktionsweise der Protokollierung	1346
Anzeigen von Protokollen	1347
Interpretieren von Protokollen	1350
Protokollreferenzen	1366
CloudWatch Metriken	1369
Leistung	1371
Leistungsdaten	1371
HSM-Drosselung	1372
Sicherheit	1373
Steuern Sie den API-Zugriff mit IAM-Richtlinien	1374
Aktualisieren Sie die IAM-Richtlinien auf IPv6	1374
Datenschutz	1377
Verschlüsselung im Ruhezustand	1379
Verschlüsselung während der Übertragung	1379
End-to-end Verschlüsselung	1379
Sicherungen von Clustern	1381
Identity and Access Management	1382
Erteilen von Berechtigungen mithilfe von IAM-Richtlinien	1383
API-Aktionen für AWS CloudHSM	1384
Bedingungsschlüssel für AWS CloudHSM	1385
Vordefinierte AWS-verwaltete Richtlinien für AWS CloudHSM	1385
Vom Kunden verwaltete Richtlinien für AWS CloudHSM	1385
Service-verknüpfte Rollen	1389
Compliance-Validierung	1391
PCI-PIN FAQs	1393
Veraltungen	1395
Ausfallsicherheit	1396
Sicherheit der Infrastruktur	1397
Netzwerkisolierung	1397
Autorisierung von Benutzern	1397
VPC-Endpunkte (AWS PrivateLink)	1397
Überlegungen zu AWS CloudHSM VPC-Endpunkten	1398
Erstellen eines Schnittstellen-VPC-Endpunkts für AWS CloudHSM	1398
Erstellen einer VPC-Endpunktrichtlinie für AWS CloudHSM	1399
Update-Management	1400

Fehlerbehebung	1401
AWS CloudHSM bekannte Probleme	1401
Bekannte Probleme für alle HSM-Instances	1402
Bekannte Probleme für hsm1.medium	1408
Bekannte Probleme für hsm2m.medium	1409
Bekannte Probleme für den PKCS#11-Bibliothek	1413
Bekannte Probleme für das JCE-SDK	1419
Bekannte Probleme für die OpenSSL Dynamic Engine	1424
Bekannte Probleme beim Key Storage Provider (KSP)	1428
Bekannte Probleme für den OpenSSL Provider	1429
Bekannte Probleme mit Amazon-EC2-Instances, auf denen Amazon Linux 2 ausgeführt wird	1430
Bekannte Probleme bei der Integration von Drittanbieter-Anwendungen	1431
Bekannte Probleme bei der Clustermodifikation	1431
Bekannte Probleme aufgrund von Betriebsfehlern bei Verwendung AWS CloudHSM der Client-Version 5.12.0 auf hsm2.medium	1432
Fehler bei der Schlüsselsynchronisierung des Client-SDK 3	1433
Das Client-SDK 3 verifiziert die Leistung	1434
Empfehlungen testen	1436
Konfigurierbare Optionen für das Tool pkpspeed	1436
Tests, die mit dem Tool pkpspeed ausgeführt werden können	1437
Beispiele	1438
Der Client SDK 5-Benutzer enthält inkonsistente Werte	1441
Fehler bei der Replikation von Client-SDK 5-Benutzern	1447
Problem: Der ausgewählte Benutzer ist nicht im gesamten Cluster synchronisiert	1448
Problem: Auf dem Zielcluster ist ein Benutzer mit unterschiedlichen Attributen vorhanden .	1449
Problem: Die Benutzerreplikation von hsm2m.medium nach hsm1.medium schlägt fehl	1450
Fehler bei der Schlüsselreplikation des Client-SDK 5	1450
Problem: Der ausgewählte Schlüssel ist nicht im gesamten Cluster synchronisiert	1450
Problem: Ein Schlüssel mit derselben Referenz ist im Zielcluster mit unterschiedlichen Informationen oder Attributen vorhanden	1452
AWS CloudHSM Bei der Überprüfung der Schlüsselverfügbarkeit ist ein Fehler aufgetreten ...	1453
Extrahieren von Schlüsseln mit JCE	1454
getEncoded oder getPrivateExponent GetS gibt null zurück	1454
getEncoded oder getS geben Schlüsselbytes außerhalb des HSM zurück	
getPrivateExponent	1454

HSM-Drosselung	1454
Auflösung	1455
Aufrechterhalten der Synchronität von HSM-Benutzern	1456
Verbindung getrennt	1457
Fehlende AWS CloudHSM Audit-Logs CloudWatch	1460
Nicht konforme AES-Schlüsselumbrüche	1460
Stellen Sie fest, ob Ihr Code unwiederbringliche verpackte Schlüssel generiert	1460
Maßnahmen, die Sie ergreifen müssen, wenn Ihr Code unwiederbringliche verpackte Schlüssel generiert	1462
Behebung von Fehlern bei der AWS CloudHSM Clustererstellung	1463
Fügen Sie die fehlende Berechtigung hinzu	1464
Manuelles Erstellen der serviceverknüpften Rolle	1464
Verwenden Sie einen Benutzer, der nicht an einen Verbund angeschlossen ist	1464
AWS CloudHSM Client-Konfigurationsprotokolle abrufen	1465
Support-Tool für das Client-SDK 5	1466
Support-Tool für das Client-SDK 3	1468
Kontingente	1470
Downloads	1472
Neueste Version	1472
Version 5 des Client-SDK: Version 5.17.1	1473
Frühere Version	1479
Veraltete Versionen	1524
Veraltete Versionen des Client SDK 5	1524
Veraltete Versionen des Client SDK 3	1539
End-of-life Veröffentlichungen	1549
Dokumentverlauf	1550
Neueste Aktualisierungen	1550
Frühere Aktualisierungen	1557
.....	mdlix

Was ist AWS CloudHSM?

AWS CloudHSM kombiniert die Vorteile der AWS Cloud mit der Sicherheit von Hardware-Sicherheitsmodulen (HSMs). Ein Hardwaresicherheitsmodul (HSM) ist eine Hardwarekomponente bzw. Appliance, die kryptografische Operationen durchführt und sicheren Speicher für kryptographische Schlüssel bietet. Mit AWS CloudHSM haben Sie die vollständige Kontrolle über HSMs die Hochverfügbarkeit in der AWS-Cloud, haben Zugriff mit geringer Latenz und eine sichere Vertrauensbasis, die das HSM-Management (einschließlich Backups, Bereitstellung, Konfiguration und Wartung) automatisiert.

AWS CloudHSM bietet Kunden eine Vielzahl von Vorteilen:

Zugriff auf FIPS- und Nicht-FIPS-Cluster

AWS CloudHSM bietet Cluster in zwei Modi: FIPS und Nicht-FIPS. Im FIPS-Modus können nur nach dem Federal Information Processing Standard (FIPS) validierte Schlüssel und Algorithmen verwendet werden. Der Nicht-FIPS-Modus bietet alle Schlüssel und Algorithmen, die von unterstützt werden AWS CloudHSM, unabhängig von der FIPS-Zulassung. Weitere Informationen finden Sie unter [AWS CloudHSM Cluster-Modi](#).

HSMs sind Allzweck-Einzelmandanten und entweder nach FIPS 140-2 Level-3 oder FIPS 140-3 Level-3 für Cluster im FIPS-Modus validiert

AWS CloudHSM verwendet Allzweck-Services HSMs , die im Vergleich zu vollständig verwalteten AWS-Services, die über vordefinierte Algorithmen und Schlüssellängen für Ihre Anwendung verfügen, mehr Flexibilität bieten. Wir bieten standardkonforme Single-Tenant-Systeme an, HSMs die entweder nach FIPS 140-2 Level-3 oder FIPS 140-3 Level-3 für Cluster im FIPS-Modus validiert sind. Für Kunden mit Anwendungsfällen, die außerhalb der Einschränkungen der FIPS 140-2- oder FIPS 140-3 Level-3-Validierung liegen, werden auch Cluster im Nicht-FIPS-Modus angeboten. AWS CloudHSM Weitere Informationen finden Sie unter [AWS CloudHSM Cluster](#).

Die E2E-Verschlüsselung ist für AWS nicht sichtbar

Da Ihre Datenebene end-to-end (E2E) verschlüsselt und für AWS nicht sichtbar ist, kontrollieren Sie Ihre eigene Benutzerverwaltung (außerhalb der IAM-Rollen). Der Nachteil dieser Kontrolle besteht darin, dass Sie mehr Verantwortung tragen, als wenn Sie einen verwalteten AWS-Service verwenden würden.

Vollständige Kontrolle über Ihre Schlüssel, Algorithmen und Anwendungsentwicklung

AWS CloudHSM gibt Ihnen die volle Kontrolle über die Algorithmen und Schlüssel, die Sie verwenden. Sie können kryptografische Schlüssel (einschließlich Sitzungsschlüssel, Token-Schlüssel, symmetrische Schlüssel und asymmetrische Schlüsselpaare) erzeugen, speichern, importieren, exportieren, verwalten und verwenden. Darüber hinaus haben AWS CloudHSM SDKs Sie die volle Kontrolle über die Anwendungsentwicklung, die Anwendungssprache, das Threading und darüber, wo Ihre Anwendungen physisch existieren.

Migrieren Sie Ihre kryptografischen Workloads in die Cloud

Kunden, die eine Public-Key-Infrastruktur migrieren, die Public Key Cryptography Standards #11 (PKCS #11), Java Cryptographic Extension (JCE), Cryptography API: Next Generation (CNG) oder Key Storage Provider (KSP) verwendet, können mit weniger Änderungen an ihrer Anwendung zu migrieren. AWS CloudHSM

Weitere Informationen darüber, was Sie damit machen können, finden Sie in den folgenden Themen. AWS CloudHSM Wenn Sie bereit sind, loszulegen AWS CloudHSM, finden Sie weitere Informationen unter [Erste Schritte](#).

Note

Wenn Sie einen verwalteten Dienst für die Erstellung und Steuerung Ihrer Verschlüsselungsschlüssel benötigen, aber keinen eigenen Dienst betreiben möchten oder müssen HSMs, sollten Sie ihn verwenden [AWS Key Management Service](#). Wenn Sie nach einem elastischen Service suchen, der Zahlungen HSMs und Schlüssel für Zahlungsverarbeitungsanwendungen in der Cloud verwaltet, sollten Sie die Verwendung von [AWS Payment Cryptography](#) in Betracht ziehen.

Inhalt

- [AWS CloudHSM Anwendungsfälle](#)
- [Wie AWS CloudHSM funktioniert](#)
- [Preisgestaltung für AWS CloudHSM](#)

AWS CloudHSM Anwendungsfälle

AWS CloudHSM kann verwendet werden, um eine Vielzahl von Zielen zu erreichen. Der Inhalt dieses Themas bietet einen Überblick darüber, was Sie damit machen können AWS CloudHSM.

Einhaltung gesetzlicher Vorschriften

Unternehmen, die sich an Unternehmenssicherheitsstandards halten müssen AWS CloudHSM , können private Schlüssel zum Schutz hochvertraulicher Daten verwenden. Die von HSMs bereitgestellten Geräte AWS CloudHSM sind nach FIPS 140-2 Level 3 zertifiziert und entsprechen PCI DSS. Darüber hinaus AWS CloudHSM ist es PCI-PIN-konform und PCI-3DS-konform. Weitere Informationen finden Sie unter [Konformitätsprüfung für AWS CloudHSM](#).

Verschlüsseln und Entschlüsseln von Daten

Wird AWS CloudHSM zur Verwaltung von privaten Schlüsseln zum Schutz hochvertraulicher Daten, zur Verschlüsselung bei der Übertragung und zur Verschlüsselung im Ruhezustand verwendet. Darüber hinaus AWS CloudHSM bietet es eine standardkonforme Integration mit mehreren kryptografischen Daten. SDKs

Signieren und verifizieren der Dokumente mit privaten und öffentlichen Schlüsseln

In der Kryptografie ermöglicht die Verwendung eines privaten Schlüssels zum Signieren eines Dokuments den Empfängern, mithilfe eines öffentlichen Schlüssels zu überprüfen, ob Sie (und nicht jemand anderes) das Dokument tatsächlich gesendet haben. Wird verwendet AWS CloudHSM , um asymmetrische öffentliche und private Schlüsselpaare zu erstellen, die speziell für diesen Zweck entwickelt wurden.

Authentifizieren Sie Nachrichten mit und HMACs CMACs

In der Kryptografie werden Cipher Message Authentication Codes (CMACs) und Hash-basierte Message Authentication Codes (HMACs) verwendet, um Nachrichten zu authentifizieren und deren Integrität zu gewährleisten, die über unsichere Netzwerke gesendet werden. Mit können Sie sicher symmetrische Schlüssel erstellen und verwalten AWS CloudHSM, die und unterstützen. HMACs CMACs

Nutzen Sie die Vorteile von und AWS CloudHSM AWS Key Management Service

Kunden können wichtige Materialien in einer Single-Tenant-Umgebung kombinieren AWS CloudHSM und [AWS KMS](#) speichern und gleichzeitig die wichtigsten Vorteile von AWS KMS Management, Skalierung und Cloud-Integration nutzen. Einzelheiten dazu finden Sie im Entwicklerhandbuch AWS Key Management Service unter [AWS CloudHSM Key Stores](#).

Die SSL/TLS Verarbeitung für Webserver auslagern

Um Daten sicher über das Internet zu senden, verwenden Webserver öffentlich-private Schlüsselpaare und SSL/TLS öffentliche Schlüsselzertifikate, um HTTPS-Sitzungen einzurichten. Dieser Prozess erfordert eine Menge Rechenleistung für Webserver, aber Sie können den Rechenaufwand reduzieren und gleichzeitig zusätzliche Sicherheit bieten, indem Sie einen Teil davon auf Ihren Cluster auslagern. AWS CloudHSM Hinweise zum Einrichten von SSL/TLS Offload mit finden Sie unter. AWS CloudHSM [SSL/TLS-Auslagerung](#)

Aktivieren Sie die transparente Datenverschlüsselung (TDE)

Transparent Data Encryption (TDE) wird für die Verschlüsselung von Datenbankdateien eingesetzt. Mit TDE verschlüsselt die Datenbanksoftware die Daten, bevor sie auf der Festplatte gespeichert werden. Sie können mehr Sicherheit erreichen, indem Sie den TDE-Hauptverschlüsselungsschlüssel HSMs in Ihrem speichern. AWS CloudHSM Hinweise zur Einrichtung von Oracle TDE mit finden Sie AWS CloudHSM unter. [Oracle Database-Verschlüsselung](#)

Verwalten Sie die privaten Schlüssel einer ausstellenden Zertifizierungsstelle (CA)

Eine Zertifizierungsstelle (CA) ist eine vertrauenswürdige Entität, die digitale Zertifikate ausstellt, die einen öffentlichen Schlüssel an eine Identität (eine Person oder Organisation) binden. Um eine CA zu betreiben, müssen Sie das Vertrauen aufrechterhalten, indem Sie den privaten Schlüssel schützen, der die von Ihrer CA ausgestellten Zertifikate signiert. Sie können solche privaten Schlüssel in Ihrem AWS CloudHSM Cluster speichern und sie dann verwenden, HSMs um kryptografische Signaturvorgänge durchzuführen.

Generieren von Zufallszahlen

Die Generierung von Zufallszahlen zur Erstellung von Verschlüsselungsschlüsseln ist ein zentraler Bestandteil der Online-Sicherheit. AWS CloudHSM kann zur sicheren Generierung von Zufallszahlen unter Ihrer HSMs Kontrolle verwendet werden und sind nur für Sie sichtbar.

Wie AWS CloudHSM funktioniert

Dieses Thema bietet einen Überblick über die grundlegenden Konzepte und die Architektur, mit denen Sie Daten sicher verschlüsseln und kryptografische Operationen ausführen. HSMs AWS CloudHSM arbeitet in Ihrer eigenen Amazon Virtual Private Cloud (VPC). Bevor Sie ihn verwenden können AWS CloudHSM, erstellen Sie zunächst einen Cluster, fügen ihn HSMs hinzu, erstellen Benutzer und Schlüssel und verwenden dann den Client, SDKs um Ihren in Ihre HSMs Anwendung zu integrieren. Sobald dies erledigt ist, verwenden Sie Client SDK-Protokolle AWS CloudTrail, Audit-Logs und Amazon CloudWatch zur [Überwachung AWS CloudHSM](#).

Lernen Sie AWS CloudHSM die grundlegenden Konzepte kennen und erfahren Sie, wie sie zusammenarbeiten, um Ihre Daten zu schützen.

Themen

- [AWS CloudHSM Cluster](#)
- [Benutzer in AWS CloudHSM](#)
- [Schlüssel rein AWS CloudHSM](#)
- [Kunde SDKs für AWS CloudHSM](#)
- [AWS CloudHSM Cluster-Backups](#)
- [Unterstützte Regionen für AWS CloudHSM](#)

AWS CloudHSM Cluster

Es kann schwierig sein, einzelne Personen auf synchronisierte, redundante und hochverfügbare Weise HSMs zusammenarbeiten zu lassen, AWS CloudHSM nimmt Ihnen jedoch die Arbeit ab, da Hardware-Sicherheitsmodule (HSMs) in Clustern bereitgestellt werden. Ein Cluster ist eine Sammlung von Personen HSMs , die AWS CloudHSM synchron bleiben. Wenn Sie eine Aufgabe oder einen Vorgang auf einem HSM in einem Cluster ausführen, werden die anderen HSMs in diesem Cluster automatisch auf dem neuesten Stand gehalten.

AWS CloudHSM bietet Cluster in zwei Modi an: FIPS und Nicht-FIPS. Im FIPS-Modus können nur nach dem Federal Information Processing Standard (FIPS) validierte Schlüssel und Algorithmen verwendet werden. Der Nicht-FIPS-Modus bietet alle Schlüssel und Algorithmen, die von unterstützt werden AWS CloudHSM, unabhängig von der FIPS-Zulassung. AWS CloudHSM bietet auch zwei Typen von HSMs: hsm1.medium und hsm2m.medium. Einzelheiten zu den Unterschieden zwischen den einzelnen HSM-Typen und Clustermodi finden Sie unter [AWS CloudHSM Cluster-Modi](#). Der HSM-Typ hsm1.medium hat das Ende der Unterstützung erreicht, sodass mit diesem Typ keine neuen Cluster erstellt werden können. Weitere Informationen finden Sie unter Benachrichtigungen über [veraltete](#) Versionen.

Um Ihre Ziele in Bezug auf Verfügbarkeit, Beständigkeit und Skalierbarkeit zu erreichen, legen Sie die Anzahl der Mitglieder HSMs in Ihrem Cluster auf mehrere Availability Zones fest. Sie können einen Cluster mit 1 bis 28 erstellen HSMs (das [Standardlimit](#) liegt bei 6 HSMs pro AWS Konto und [AWS Region](#)). Sie können sie HSMs in verschiedenen [Availability Zones](#) in einer AWS Region platzieren. Wenn Sie mehr HSMs zu einem Cluster hinzufügen, erzielen Sie eine höhere Leistung. Wenn Sie Cluster über mehrere Availability Zones hinweg verteilen, bietet dies Redundanz und hohe Verfügbarkeit.

Weitere Informationen zu Clustern finden Sie unter [Cluster in AWS CloudHSM](#).

Informationen zum Erstellen eines Clusters finden Sie unter [Erste Schritte](#).

Benutzer in AWS CloudHSM

Im Gegensatz zu den meisten AWS Diensten und Ressourcen verwenden Sie keine AWS Identity and Access Management (IAM-) Benutzer oder IAM-Richtlinien, um auf Ressourcen innerhalb Ihres Clusters zuzugreifen. AWS CloudHSM Stattdessen verwenden Sie HSM-Benutzer direkt HSMs in Ihrem Cluster. AWS CloudHSM

HSM-Benutzer unterscheiden sich von IAM-Benutzern. IAM-Benutzer, die über die richtigen Anmeldeinformationen verfügen, können HSMs durch Interaktion mit Ressourcen über die AWS-API etwas erstellen. Da die E2E-Verschlüsselung für AWS nicht sichtbar ist, müssen Sie HSM-Benutzeranmeldedaten verwenden, um Vorgänge auf dem HSM zu authentifizieren, da die Anmeldeinformationen direkt auf dem HSM erfolgen. Das HSM authentifiziert jeden HSM-Benutzer anhand von Anmeldeinformationen, die Sie definieren und verwalten. Jeder HSM-Benutzer hat einen Typ, der bestimmt, welche Operationen dieser Benutzer auf dem HSM durchführen kann. Jedes HSM authentifiziert jeden HSM-Benutzer anhand von Anmeldeinformationen, die Sie mit der [CloudHSM-CLI](#) definieren.

Wenn Sie die [vorherige SDK-Versionsserie](#) verwenden, verwenden Sie [CloudHSM Management Utility \(CMU\)](#).

Schlüssel rein AWS CloudHSM

AWS CloudHSM ermöglicht es Ihnen, Ihre Verschlüsselungsschlüssel, die sich in Ihrem Cluster befinden, sicher in einem einzigen Mandanten zu generieren HSMs , zu speichern und zu verwalten. AWS CloudHSM Schlüssel können symmetrisch oder asymmetrisch sein, sie können Sitzungsschlüssel (kurzlebige Schlüssel) für einzelne Sitzungen und Tokenschlüssel (persistente Schlüssel) für den langfristigen Gebrauch sein. Sie können aus und importiert werden. AWS CloudHSM Schlüssel können auch zur Ausführung gängiger kryptografischer Aufgaben und Funktionen verwendet werden:

- Führen Sie kryptografische Datensignierung und Signaturverifizierung mit symmetrischen und asymmetrischen Verschlüsselungsalgorithmen durch.
- Verwenden Sie Hashfunktionen, um Nachrichtenübersichten und Hash-basierte Nachrichtenauthentifizierungscode zu berechnen (). HMACs
- Wickeln und schützen Sie andere Schlüssel.
- Zugriff auf kryptographisch sichere Zufallsdaten.

Die maximale Anzahl von Schlüsseln, die ein Cluster haben kann, hängt von der Art der Schlüssel ab HSMs , die sich im Cluster befinden. Beispielsweise speichert hsm2m.medium mehr Schlüssel als hsm1, medium. Einen Vergleich finden Sie unter. [AWS CloudHSM Kontingente](#)

Darüber hinaus AWS CloudHSM gelten einige grundlegende Prinzipien für die Verwendung und Verwaltung von Schlüsseln:

Es stehen viele Schlüsseltypen und Algorithmen zur Auswahl

Damit Sie Ihre eigenen Lösungen anpassen können, AWS CloudHSM bietet es viele Schlüsseltypen und Algorithmen zur Auswahl. Algorithmen unterstützen eine Reihe von Schlüsselgrößen. Weitere Informationen finden Sie auf den Seiten zu Attributen und Mechanismen von jedem [Offload-Operationen mit dem Client AWS CloudHSM SDKs](#).

Wie verwalten Sie Schlüssel

AWS CloudHSM Schlüssel werden über SDKs Befehlszeilentools verwaltet. Informationen zur Verwendung dieser Tools zur Verwaltung von Schlüsseln finden Sie unter [Schlüssel rein AWS CloudHSM](#) und [Bewährte Methoden für AWS CloudHSM](#).

Wem gehören Schlüssel

In AWS CloudHSM gehört er dem Krypto-Benutzer (CU), der den Schlüssel erstellt. Der Besitzer kann die key unshare Befehle key share und verwenden, um den Schlüssel mit anderen CUs zu teilen oder die gemeinsame Nutzung aufzuheben. Weitere Informationen finden Sie unter [Schlüssel mit CloudHSM CLI teilen und deren Freigabe rückgängig machen](#).

Zugriff und Nutzung können mit attributebasierter Verschlüsselung gesteuert werden

AWS CloudHSM ermöglicht Ihnen die Verwendung der attributbasierten Verschlüsselung, einer Form der Verschlüsselung, mit der Sie anhand von Schlüsselattributen steuern können, wer Daten auf der Grundlage von Richtlinien entschlüsseln kann.

Kunde SDKs für AWS CloudHSM

Bei der Verwendung AWS CloudHSM führen Sie kryptografische Operationen mit [AWS CloudHSM Client Software Development Kits \(\) SDKs](#) durch. AWS CloudHSM SDKs Zu den Kunden gehören:

- Public Key Cryptography Standards #11 (PKCS #11)
- JCE-Anbieter
- OpenSSL Dynamic Engine
- Key Storage Provider (KSP) für Microsoft Windows

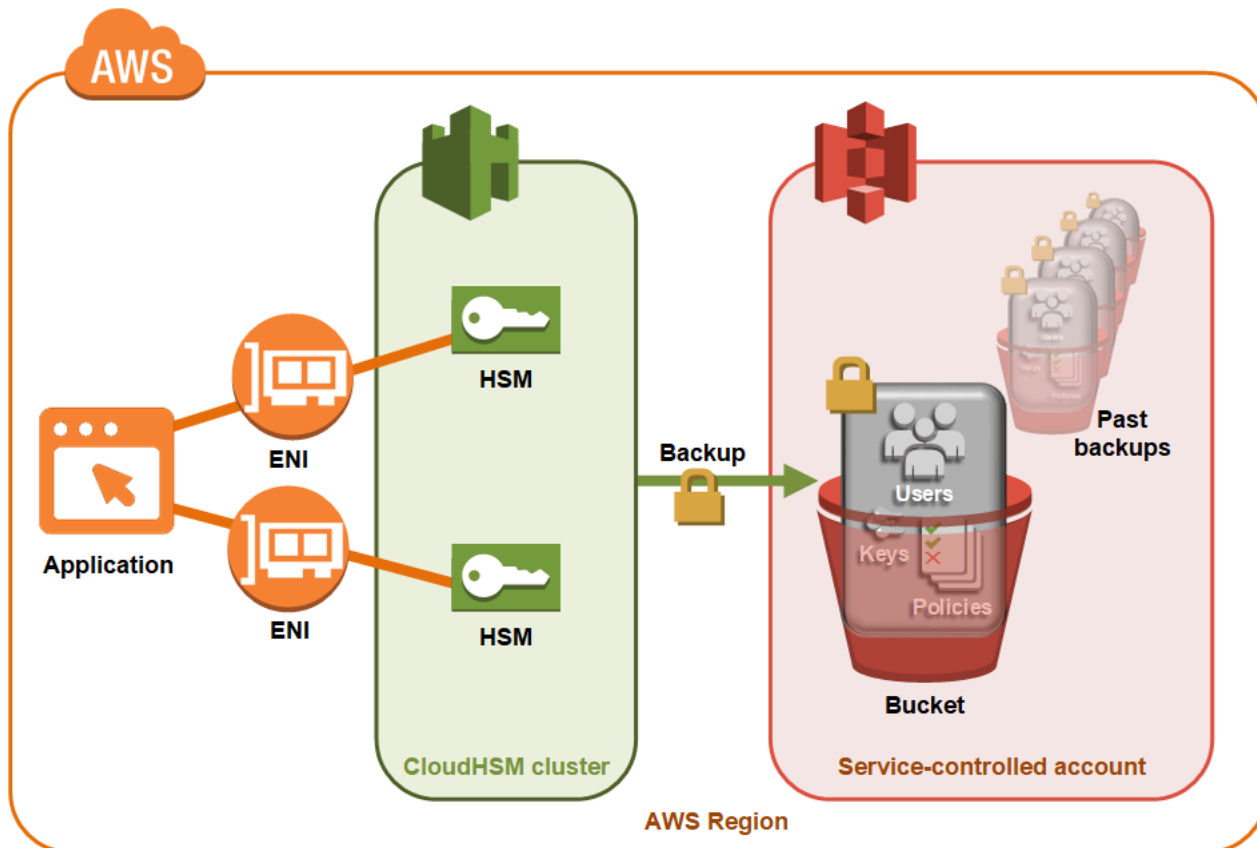
Sie können einige oder alle dieser SDKs in Ihrem AWS CloudHSM Cluster verwenden. Schreiben Sie Ihren Anwendungscode, SDKs um diese für die Ausführung kryptografischer Operationen in Ihrem zu verwenden. HSMs Informationen darüber, welche Plattformen und HSM-Typen die einzelnen SDKs unterstützen, finden Sie unter [AWS CloudHSM Client SDK 5 unterstützte Plattformen](#)

Dienstprogramme und Befehlszeilentools werden nicht nur für die Verwendung, SDKs sondern auch für die Konfiguration der Anmeldeinformationen, Richtlinien und Einstellungen Ihrer Anwendung benötigt. Weitere Informationen finden Sie unter [AWS CloudHSM Befehlszeilentools](#).

Weitere Informationen zur Installation und Verwendung des Client-SDK oder zur Sicherheit der Client-Verbindung finden Sie unter [Kunde SDKs](#) und [End-to-end Verschlüsselung](#).

AWS CloudHSM Cluster-Backups

AWS CloudHSM erstellt regelmäßig Backups der Benutzer, Schlüssel und Richtlinien im Cluster. Sicherungen sind sicher, dauerhaft und werden nach einem vorhersehbaren Zeitplan aktualisiert. Die folgende Abbildung zeigt die Beziehung zwischen Ihren Sicherungen und dem Cluster.



Weitere Informationen zur Arbeit mit Sicherungen finden Sie unter [Sicherungen von Clustern](#).

Sicherheit

Wenn AWS CloudHSM ein Backup vom HSM erstellt wird, verschlüsselt das HSM all seine Daten, bevor es sie an sendet. AWS CloudHSM Die Daten werden zu keinem Zeitpunkt unverschlüsselt aus dem HSM ausgegeben. Darüber hinaus können Backups nicht entschlüsselt werden, AWS da AWS es keinen Zugriff auf den Schlüssel hat, der zum Entschlüsseln der Backups verwendet wurde. Weitere Informationen finden Sie unter [Sicherheit von Cluster-Sicherungen](#).

Haltbarkeit

AWS CloudHSM speichert Backups in einem servicegesteuerten Amazon Simple Storage Service (Amazon S3) -Bucket in derselben Region wie Ihr Cluster. Sicherungen haben eine Beständigkeit von 99,999999999 %, genau wie jedes in Amazon S3 gespeicherte Objekt.

Unterstützte Regionen für AWS CloudHSM

Informationen zu den unterstützten Regionen für AWS CloudHSM finden Sie unter [AWS CloudHSM Regionen und Endpunkte](#) in der Allgemeine AWS-Referenz oder der [Regionentabelle](#).

AWS CloudHSM ist möglicherweise nicht in allen Availability Zones einer bestimmten Region verfügbar. Dies sollte sich jedoch nicht auf die Leistung auswirken, da der Lastenausgleich für alle HSMs in einem Cluster AWS CloudHSM automatisch erfolgt.

Wie bei den meisten AWS Ressourcen HSMs handelt es sich auch bei Clustern und um regionale Ressourcen. Sie können einen Cluster nicht in mehreren Regionen wiederverwenden oder ihn auf mehrere Regionen erweitern. Sie müssen alle unter [Erste Schritte mit AWS CloudHSM](#) aufgelisteten erforderlichen Schritte ausführen, um einen neuen Cluster in einer neuen Region zu erstellen.

AWS CloudHSM ermöglicht es Ihnen, Backups Ihres AWS CloudHSM Clusters zu Notfallwiederherstellungszwecken von einer Region in eine andere zu kopieren. Weitere Informationen finden Sie unter [AWS CloudHSM Cluster-Backups](#).

Preisgestaltung für AWS CloudHSM

Mit AWS CloudHSM zahlen Sie stundenweise ohne langfristige Verpflichtungen oder Vorauszahlungen. Weitere Informationen finden Sie auf der AWS Website unter [AWS CloudHSM Preise](#).

Erste Schritte mit AWS CloudHSM

Die folgenden Themen helfen Ihnen beim Erstellen, Initialisieren und Aktivieren eines Clusters in AWS CloudHSM. Nachdem Sie diese Verfahren abgeschlossen haben, können Sie Benutzer verwalten, Cluster verwalten und die integrierten Software-Bibliotheken zur Durchführung kryptografischer Operationen nutzen. Für eine optimale Benutzererfahrung sollten Sie die Themen in der angegebenen Reihenfolge befolgen.

Inhalt

- [Erstellen Sie IAM-Administrationsgruppen für AWS CloudHSM](#)
- [Erstellen Sie eine virtuelle private Cloud \(VPC\) für AWS CloudHSM](#)
- [Erstellen Sie einen Cluster in AWS CloudHSM](#)
- [Überprüfen Sie die Sicherheitsgruppe für Ihren Cluster in AWS CloudHSM](#)
- [Starten Sie eine Amazon EC2 EC2-Client-Instance für die Interaktion mit AWS CloudHSM](#)
- [Konfigurieren Sie die Sicherheitsgruppen der Amazon EC2 EC2-Client-Instanz für AWS CloudHSM](#)
- [Erstellen Sie ein HSM in AWS CloudHSM](#)
- [Überprüfen Sie die Identität und Authentizität des HSM Ihres Clusters in AWS CloudHSM \(optional\)](#)
- [Initialisieren Sie den Cluster in AWS CloudHSM](#)
- [CloudHSM-CLI installieren und konfigurieren](#)
- [Aktiviere den Cluster in AWS CloudHSM](#)
- [Richten Sie gegenseitiges TLS zwischen Client und AWS CloudHSM \(empfohlen\) ein](#)
- [Schlüssel erstellen und verwenden in AWS CloudHSM](#)

Erstellen Sie IAM-Administrationsgruppen für AWS CloudHSM

Der erste Schritt für den Einstieg AWS CloudHSM besteht darin, IAM-Berechtigungen einzurichten.

Als [bewährte Methode sollten](#) Sie Ihre nicht verwenden, Root-Benutzer des AWS-Kontos um mit AWS anderen zu interagieren. AWS CloudHSM Verwenden Sie stattdessen AWS Identity and Access Management (IAM), um einen IAM-Benutzer, eine IAM-Rolle oder einen Verbundbenutzer zu erstellen. Folgen Sie den Schritten im Abschnitt [Erstellen von IAM-Benutzern und -Administratorgruppen](#), um eine Administratorgruppe zu erstellen und ihr die AdministratorAccessRichtlinie anzuhängen. Erstellen Sie dann einen neuen Administratorbenutzer

und fügen Sie den Benutzer der Gruppe hinzu. Fügen Sie der Gruppe bei Bedarf weitere Benutzer hinzu. Jeder Benutzer, den Sie hinzufügen, erbt die AdministratorAccessRichtlinie von der Gruppe.

Eine weitere bewährte Methode besteht darin, eine AWS CloudHSM Administratorgruppe zu erstellen, die nur über die für die Ausführung AWS CloudHSM erforderlichen Berechtigungen verfügt. Fügen Sie dieser Gruppe bei Bedarf einzelne Benutzer hinzu. Jeder Benutzer übernimmt die eingeschränkten Berechtigungen, die der Gruppe zugeordnet sind, anstatt uneingeschränkten AWS-Zugriff zu erhalten. Der folgende [Vom Kunden verwaltete Richtlinien für AWS CloudHSM](#) Abschnitt enthält die Richtlinie, die Sie Ihrer AWS CloudHSM Administratorgruppe zuordnen sollten.

AWS CloudHSM definiert eine [dienstbezogene Rolle](#) für Ihr AWS Konto. Die dienstbezogene Rolle definiert derzeit Berechtigungen, die es Ihrem Konto ermöglichen, Ereignisse zu protokollieren. AWS CloudHSM Die Rolle kann automatisch von Ihnen AWS CloudHSM oder manuell erstellt werden. Sie können die Rolle zwar nicht bearbeiten, aber Sie können sie löschen. Weitere Informationen finden Sie unter [Mit Diensten verknüpfte Rollen für AWS CloudHSM](#).

Erstellen von IAM-Benutzern und -Administratorgruppen

Beginnen Sie, indem Sie einen IAM-Benutzer sowie eine Administratorgruppe für diesen Benutzer erstellen.

Melde dich an für eine AWS-Konto

Wenn Sie noch keine haben AWS-Konto, führen Sie die folgenden Schritte aus, um eine zu erstellen.

Um sich für eine anzumelden AWS-Konto

1. Öffnen Sie [https://portal.aws.amazon.com/billing/die Anmeldung](https://portal.aws.amazon.com/billing/die-Anmeldung).
2. Folgen Sie den Online-Anweisungen.

Während der Anmeldung erhalten Sie einen Telefonanruf oder eine Textnachricht und müssen einen Verifizierungscode über die Telefontasten eingeben.

Wenn Sie sich für eine anmelden AWS-Konto, Root-Benutzer des AWS-Kontos wird eine erstellt. Der Root-Benutzer hat Zugriff auf alle AWS-Services und Ressourcen des Kontos. Als bewährte Sicherheitsmethode weisen Sie einem Benutzer Administratorzugriff zu und verwenden Sie nur den Root-Benutzer, um [Aufgaben auszuführen, die Root-Benutzerzugriff erfordern](#).

AWS sendet Ihnen nach Abschluss des Anmeldevorgangs eine Bestätigungs-E-Mail. Du kannst jederzeit deine aktuellen Kontoaktivitäten einsehen und dein Konto verwalten, indem du zu <https://aws.amazon.com/> gehst und Mein Konto auswählst.

Erstellen eines Benutzers mit Administratorzugriff

Nachdem Sie sich für einen angemeldet haben AWS-Konto, sichern Sie Ihren Root-Benutzer des AWS-Kontos AWS IAM Identity Center, aktivieren und erstellen Sie einen Administratorbenutzer, sodass Sie den Root-Benutzer nicht für alltägliche Aufgaben verwenden.

Sichern Sie Ihre Root-Benutzer des AWS-Kontos

1. Melden Sie sich [AWS-Managementkonsole](#) als Kontoinhaber an, indem Sie Root-Benutzer auswählen und Ihre AWS-Konto E-Mail-Adresse eingeben. Geben Sie auf der nächsten Seite Ihr Passwort ein.

Hilfe bei der Anmeldung mit dem Root-Benutzer finden Sie unter [Anmelden als Root-Benutzer](#) im AWS-Anmeldung -Benutzerhandbuch zu.

2. Aktivieren Sie die Multi-Faktor-Authentifizierung (MFA) für den Root-Benutzer.

Anweisungen finden Sie unter [Aktivieren eines virtuellen MFA-Geräts für Ihren AWS-Konto Root-Benutzer \(Konsole\)](#) im IAM-Benutzerhandbuch.

Erstellen eines Benutzers mit Administratorzugriff

1. Aktivieren Sie das IAM Identity Center.

Anweisungen finden Sie unter [Aktivieren AWS IAM Identity Center](#) im AWS IAM Identity Center - Benutzerhandbuch.

2. Gewähren Sie einem Administratorbenutzer im IAM Identity Center Benutzerzugriff.

Ein Tutorial zur Verwendung von IAM-Identity-Center-Verzeichnis als Identitätsquelle finden Sie IAM-Identity-Center-Verzeichnis im Benutzerhandbuch unter [Benutzerzugriff mit der Standardeinstellung konfigurieren](#).AWS IAM Identity Center

Anmelden als Administratorbenutzer

- Um sich mit Ihrem IAM-Identity-Center-Benutzer anzumelden, verwenden Sie die Anmelde-URL, die an Ihre E-Mail-Adresse gesendet wurde, als Sie den IAM-Identity-Center-Benutzer erstellt haben.

Hilfe bei der Anmeldung mit einem IAM Identity Center-Benutzer finden Sie [im AWS-Anmeldung Benutzerhandbuch unter Anmeldung beim AWS Access-Portal](#).

Weiteren Benutzern Zugriff zuweisen

1. Erstellen Sie im IAM-Identity-Center einen Berechtigungssatz, der den bewährten Vorgehensweisen für die Anwendung von geringsten Berechtigungen folgt.

Anweisungen hierzu finden Sie unter [Berechtigungssatz erstellen](#) im AWS IAM Identity Center - Benutzerhandbuch.

2. Weisen Sie Benutzer einer Gruppe zu und weisen Sie der Gruppe dann Single Sign-On-Zugriff zu.

Eine genaue Anleitung finden Sie unter [Gruppen hinzufügen](#) im AWS IAM Identity Center - Benutzerhandbuch.

Richtlinien AWS CloudHSM, die Sie Ihrer IAM-Benutzergruppe zuordnen können, finden Sie beispielsweise unter [Identitäts- und Zugriffsmanagement für AWS CloudHSM](#)

Erstellen Sie eine virtuelle private Cloud (VPC) für AWS CloudHSM

Sie benötigen eine Virtual Private Cloud (VPC) für Ihren Cluster in AWS CloudHSM. Wenn Sie noch keine haben, folgen Sie den Schritten in diesem Thema, um eine VPC zu erstellen.

Note

Wenn Sie diese Schritte befolgen, werden öffentliche und private Subnetze erstellt.

So erstellen Sie eine VPC

1. Öffnen Sie die Amazon-VPC-Konsole unter <https://console.aws.amazon.com/vpc/>.

2. Verwenden Sie in der Navigationsleiste die Regionsauswahl, um eine der [AWS Regionen auszuwählen, die AWS CloudHSM derzeit unterstützt](#) werden.
3. Wählen Sie die Schaltfläche VPC erstellen.
4. Wählen Sie unter Zu erstellende Ressourcen die Option VPC und mehr aus.
5. Geben Sie für Automatische Generierung von Namenstags einen eindeutigen Namen ein, z. B. **CloudHSM**.
6. Wählen Sie für IPv6 CIDR-Block den von Amazon bereitgestellten IPv6 CIDR-Block aus, um die IPv6 Konnektivität für Sie zu nutzen, HSMs und lassen Sie Ihrem Cluster einen IPv6 CIDR-Block AWS zuweisen. Diese Einstellung unterstützt den Dual-Stack-Netzwerktyp. Behalten Sie die Standardeinstellung bei, wenn Sie keine IPv6 Konnektivität benötigen.
7. Belassen Sie alle anderen Optionen auf ihren Standardwerten.
8. Wählen Sie VPC erstellen aus.
9. Nachdem die VPC erstellt wurde, wählen Sie VPC anzeigen aus, um die VPC anzuzeigen, die Sie gerade erstellt haben.

Erstellen Sie einen Cluster in AWS CloudHSM

Ein Cluster ist eine Sammlung einzelner Hardware-Sicherheitsmodule (HSMs). AWS CloudHSM synchronisiert die HSMs in jedem Cluster, sodass sie als logische Einheit funktionieren. AWS CloudHSM bietet zwei Typen von HSMs: hsm1.medium und hsm2m.medium. Wenn Sie einen Cluster erstellen, wählen Sie aus, welcher der beiden in Ihrem Cluster enthalten sein soll. Einzelheiten zu den Unterschieden zwischen den einzelnen HSM-Typen und Clustermodi finden Sie unter [AWS CloudHSM Cluster-Modi](#).

Wenn Sie einen Cluster erstellen, AWS CloudHSM erstellt er in Ihrem Namen eine Sicherheitsgruppe für den Cluster. Diese Sicherheitsgruppe steuert den Netzwerkzugriff auf die HSMs im Cluster. Es erlaubt nur eingehende Verbindungen von Amazon Elastic Compute Cloud (Amazon EC2) -Instances, die sich in der Sicherheitsgruppe befinden. Die Sicherheitsgruppe enthält standardmäßig zunächst keine Instances. Später [starten Sie eine Client-Instance](#) und [konfigurieren die Sicherheitsgruppe des Clusters](#), um die Kommunikation und Verbindungen mit dem HSM zu ermöglichen.

Überlegungen

- Im Folgenden finden Sie einige Überlegungen bei der Erstellung eines Clusters in AWS CloudHSM:

- Wenn Sie einen Cluster erstellen, AWS CloudHSM wird eine [serviceverknüpfte Rolle mit](#) dem Namen AWSService RoleForCloud HSM erstellt. Wenn die Rolle AWS CloudHSM nicht erstellt werden kann oder die Rolle noch nicht vorhanden ist, können Sie möglicherweise keinen Cluster erstellen. Weitere Informationen finden Sie unter [Behebung von Fehlern bei der AWS CloudHSM Clustererstellung](#). Weitere Informationen zu serviceverknüpften Rollen finden Sie unter [Mit Diensten verknüpfte Rollen für AWS CloudHSM](#).
- Wenn Sie den [AWS CloudHSM Dual-Stack-Endpoint verwenden](#) (d. h. cloudhsmv2). `<region>.api.aws`), stellen Sie sicher, dass Ihre IAM-Richtlinien entsprechend aktualisiert wurden. IPv6 Weitere Informationen finden Sie im Abschnitt „[IAM-Richtlinien aktualisieren auf](#)“ unter Sicherheit. IPv6

Sie können einen Cluster über die [AWS CloudHSM -Konsole](#), die [AWS Command Line Interface \(AWS CLI\)](#) oder die AWS CloudHSM -API erstellen.

Einzelheiten zu Cluster-Argumenten und APIs finden Sie [create-cluster](#) in der AWS CLI Befehlsreferenz.

Console

Einen Cluster erstellen (Konsole)

1. Öffnen Sie die AWS CloudHSM Konsole zu <https://console.aws.amazon.com/cloudhsm/Hause>.
2. Verwenden Sie die Regionenauswahl in der Navigationsleiste, um eine der [AWS -Regionen auszuwählen, in denen AWS CloudHSM derzeit unterstützt wird](#).
3. Wählen Sie Cluster erstellen.
4. Führen Sie im Abschnitt Cluster-Konfiguration Folgendes aus:
 - a. Wählen Sie für VPC die VPC, die Sie in [Erstellen Sie eine virtuelle private Cloud \(VPC\) für AWS CloudHSM](#) erstellt haben.
 - b. Wählen Sie für Availability Zone(s), neben jeder Availability Zone, ein von Ihnen erstelltes privates Subnetz aus.

Note

Auch wenn dies AWS CloudHSM in einer bestimmten Availability Zone nicht unterstützt wird, sollte die Leistung nicht beeinträchtigt werden, da der

Lastenausgleich für alle HSMs in einem Cluster AWS CloudHSM automatisch erfolgt. Informationen zur Availability Zone-Unterstützung für AWS CloudHSM finden Allgemeine AWS-Referenz Sie unter [AWS CloudHSM Regionen und Endpunkte](#) in der.

- c. Wählen Sie als HSM-Typ den HSM-Typ aus, der in Ihrem Cluster erstellt werden kann, zusammen mit dem gewünschten Modus des Clusters. [Welche HSM-Typen in den einzelnen Regionen unterstützt werden, finden Sie im AWS CloudHSM Preisrechner.](#)



Important

Nachdem der Cluster erstellt wurde, kann der Clustermodus nicht mehr geändert werden. Informationen darüber, welcher Typ und welcher Modus für Ihren Anwendungsfall geeignet sind, finden Sie unter [AWS CloudHSM Cluster-Modi](#).

- d. Wählen Sie unter Netzwerktyp die IP-Adressprotokolle für den Zugriff auf Ihr HSMs. IPv4 schränkt die Kommunikation zwischen Ihrer Anwendung und IPv4 nur HSMs auf ein. Dies ist die Standardoption. Dual-Stack ermöglicht beides IPv4 und IPv6 Kommunikation. Um Dual-Stack zu verwenden, fügen Sie sowohl als auch IPv6 CIDRs zu Ihren VPC IPv4 - und Subnetzkonfigurationen hinzu. Der Netzwerktyp lässt sich nach der Ersteinrichtung nur schwer ändern. Um ihn zu ändern, erstellen Sie eine Sicherungskopie Ihres vorhandenen Clusters und stellen Sie einen neuen Cluster mit dem gewünschten Netzwerktyp wieder her. Weitere Informationen finden Sie unter [AWS CloudHSM Cluster aus Backups erstellen](#)
- e. Geben Sie unter Clusterquelle an, ob Sie einen neuen Cluster erstellen oder einen aus einer vorhandenen Sicherung wiederherstellen möchten.
- Backups von Clustern im Nicht-FIPS-Modus können nur zur Wiederherstellung von Clustern verwendet werden, die sich im Nicht-FIPS-Modus befinden.
 - Backups von Clustern im FIPS-Modus können nur zur Wiederherstellung von Clustern verwendet werden, die sich im FIPS-Modus befinden.
5. Wählen Sie Weiter aus.
6. Geben Sie an, wie lange der Dienst Sicherungen aufbewahren soll.
- Akzeptieren Sie den standardmäßigen Aufbewahrungszeitraum von 90 Tagen oder geben Sie einen neuen Wert zwischen 7 und 379 Tagen ein. Der Dienst löscht automatisch Sicherungen in diesem Cluster, die älter sind als der hier angegebene Wert.

Sie können dies später ändern. Weitere Informationen finden Sie unter [Konfigurieren Sie die Aufbewahrung von Backups](#).

7. Wählen Sie Weiter.
8. (Optional) Geben Sie einen Tag-Schlüssel und einen optionalen Tag-Wert ein. Wählen Sie Tag hinzufügen, wenn Sie mehr als ein Tag zum Cluster hinzufügen möchten.
9. Wählen Sie Überprüfen.
10. Überprüfen Sie Ihre Cluster-Konfiguration und wählen Sie dann Cluster erstellen aus.

Wenn Ihre Versuche, einen Cluster zu erstellen, fehlschlagen, kann dies auf Probleme mit den AWS CloudHSM dienstverknüpften Rollen zurückzuführen sein. Hilfe beim Beheben des Fehlers finden Sie unter [Behebung von Fehlern bei der AWS CloudHSM Clustererstellung](#).

AWS CLI

Erstellen eines Clusters ([AWS CLI](#))

- Führen Sie über die Eingabeaufforderung den folgenden [create-cluster](#)-Befehl aus. Geben Sie den HSM-Instanztyp, den Aufbewahrungszeitraum für Backups und das Subnetz IDs der Subnetze an, die Sie erstellen möchten. HSMs verwenden Sie das Subnetz IDs der privaten Subnetze, die Sie erstellt haben. Geben Sie nur ein Subnetz pro Availability Zone an.

```
$ 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 ist ein erforderlicher Parameter für alle HSM-Typen außer hsm1.medium. --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

```

Wenn Ihre Versuche, einen Cluster zu erstellen, fehlschlagen, kann dies auf Probleme mit den AWS CloudHSM dienstverknüpften Rollen zurückzuführen sein. Hilfe beim Beheben des Fehlers finden Sie unter [Behebung von Fehlern bei der AWS CloudHSM Clustererstellung](#).

AWS CloudHSM API

Um einen Cluster (AWS CloudHSM API) zu erstellen

- Senden Sie eine [CreateCluster](#)-Anforderung. Geben Sie den HSM-Instanztyp, die Backup-Aufbewahrungsrichtlinie und das Subnetz IDs der Subnetze an, die Sie erstellen möchten. HSMs Verwenden Sie das Subnetz IDs der privaten Subnetze, die Sie erstellt haben. Geben Sie nur ein Subnetz pro Availability Zone an.

Wenn Ihre Versuche, einen Cluster zu erstellen, fehlschlagen, kann dies auf Probleme mit den AWS CloudHSM dienstverknüpften Rollen zurückzuführen sein. Hilfe beim Beheben des Fehlers finden Sie unter [Behebung von Fehlern bei der AWS CloudHSM Clustererstellung](#).

Überprüfen Sie die Sicherheitsgruppe für Ihren Cluster in AWS CloudHSM

Wenn Sie einen Cluster erstellen oder einem Cluster ein HSM hinzufügen, AWS CloudHSM erstellt eine Sicherheitsgruppe mit dem Namen, `cloudhsm-cluster-<clusterID>-sg` falls noch keine vorhanden ist. Diese Sicherheitsgruppe enthält eine vorkonfigurierte TCP-Regel, die eine eingehende und ausgehende Kommunikation innerhalb der Cluster-Sicherheitsgruppe über die Ports 2223-2225 zulässt. Diese SG ermöglicht es Ihren EC2-Instances, Ihre VPC für die Kommunikation HSMs in Ihrem Cluster zu verwenden.

Warning

- Löschen oder ändern Sie nicht die vorkonfigurierte TCP-Regel, die in der Cluster-Sicherheitsgruppe vorhanden ist. Diese Regel kann Verbindungsprobleme und unbefugten Zugriff auf Ihre verhindern. HSMs
- Die Cluster-Sicherheitsgruppe verhindert unbefugten Zugriff auf Ihre HSMs. Jeder, der auf Instances in der Sicherheitsgruppe zugreifen kann, kann auf Ihre zugreifen HSMs. Die meisten Operationen erfordern, dass sich ein Benutzer am HSM anmeldet. Es ist jedoch möglich, HSMs ohne Authentifizierung auf Null zu setzen, wodurch das Schlüsselmaterial, die Zertifikate und andere Daten zerstört werden. Wenn dies geschieht, gehen alle Daten, die seit der letzten Sicherung erstellt oder geändert wurden, verloren und können nicht wiederhergestellt werden. Um den unbefugten Zugriff zu verhindern, stellen Sie sicher, dass nur vertrauenswürdige Administratoren die Instances in der Standardsicherheitsgruppe ändern oder auf sie zugreifen können.
- Die `hsm2m.medium`-Cluster führen die mTLS-Funktion ein, mit der nicht autorisierte Benutzer daran gehindert werden, eine Verbindung zum Cluster herzustellen. Unbefugte Benutzer benötigen gültige mTLS-Anmeldeinformationen, um erfolgreich eine Verbindung zum Cluster herzustellen, bevor sie versuchen, sie auf Null zu setzen.

Im nächsten Schritt können Sie [eine Amazon EC2 EC2-Instance starten](#) und sie mit Ihrer verbinden, HSMs indem Sie ihr [die Cluster-Sicherheitsgruppe anhängen](#).

Starten Sie eine Amazon EC2 EC2-Client-Instance für die Interaktion mit AWS CloudHSM

Um mit Ihrem AWS CloudHSM Cluster und Ihren HSM-Instances zu interagieren und diese zu verwalten, müssen Sie in der Lage sein, mit den Elastic Network-Schnittstellen Ihrer HSMs zu kommunizieren. Die einfachste Möglichkeit ist die Verwendung einer EC2-Instance in derselben VPC wie Ihr Cluster. Für die Verbindung zu Ihrem Cluster können Sie auch die folgenden AWS - Ressourcen verwenden:

- [Amazon-VPC-Peering](#)
- [Direct Connect](#)
- [VPN-Verbindungen](#)

Note

Dieses Handbuch enthält ein vereinfachtes Beispiel dafür, wie Sie eine EC2-Instance mit Ihrem Cluster verbinden. AWS CloudHSM Bewährte Methoden rund um sichere Netzwerkkonfigurationen finden Sie [Sicherer Zugriff auf Ihren Cluster](#) unter.

In der AWS CloudHSM Dokumentation wird normalerweise davon ausgegangen, dass Sie eine EC2-Instance in derselben VPC und Availability Zone (AZ) verwenden, in der Sie Ihren Cluster erstellen.

So erstellen Sie eine EC2-Instance

1. Öffnen Sie das EC2-Dashboard unter. <https://console.aws.amazon.com/ec2/>
2. Wählen Sie Instance starten aus. Wählen Sie im Drop-down-Menü die Option Instance starten aus.
3. Geben Sie in das Feld Name einen Namen für Ihre EC2-Instance ein.
4. Wählen Sie im Abschnitt Anwendungen und Betriebssystem-Images (Amazon Machine Image) ein Amazon Machine Image (AMI), das einer von CloudHSM unterstützten Plattform entspricht. Weitere Informationen finden Sie unter [AWS CloudHSM Client SDK 5 unterstützte Plattformen](#).
5. Wählen Sie im Bereich Instance-Typ einen Instance-Typ aus.
6. Verwenden Sie im Abschnitt Schlüsselpaar ein vorhandenes Schlüsselpaar oder wählen Sie Neues Schlüsselpaar erstellen aus und führen Sie die folgenden Schritte aus:

- a. Geben Sie unter Schlüsselpaarname einen Namen für das Schlüsselpaar ein.
- b. Wählen Sie für Schlüsselpaartyp einen Schlüsselpaartyp aus.
- c. Wählen Sie unter Dateiformat des privaten Schlüssels das Dateiformat des privaten Schlüssels.
- d. Wählen Sie Schlüsselpaar erstellen aus.
- e. Laden Sie die private Schlüsseldatei herunter und speichern Sie sie.

 Important

Dies ist die einzige Möglichkeit, die private Schlüsseldatei zu speichern. Laden Sie die Datei herunter und speichern Sie sie an einem sicheren Ort. Sie müssen den Namen für Ihr Schlüsselpaar beim Starten einer Instance angeben. Außerdem müssen Sie jedes Mal, wenn Sie sich mit der Instance verbinden, den entsprechenden privaten Schlüssel angeben und das Schlüsselpaar wählen, das Sie bei der Einrichtung erstellt haben.

7. Wählen Sie in den Netzwerkeinstellungen die Option Bearbeiten aus.
8. Wählen Sie für VPC die zuvor für Ihren Cluster erstellte VPC aus.
9. Wählen Sie für Subnetz das für die VPC erstellte öffentliche Subnetz aus.
10. Wählen Sie für Öffentliche IP automatisch zuweisen Aktivieren aus.
11. Wählen Sie für Auto-Assign IPv6 IP die Option Enable aus, um die IPv6-Konnektivität mit Ihren Clustern und dem Dual-Stack zu nutzen. NetworkType Wenn Sie diese Option aktivieren, aktualisieren Sie die Sicherheitsgruppenregeln, die VPC- und Subnetz-Routing-Tabellen sowie das Netzwerk Ihrer Amazon EC2 EC2-Instance, ACLs um IPv6 ausgehenden Datenverkehr von der Instance zur zuzulassen. HSMs
12. Wählen Sie Bestehende Sicherheitsgruppe auswählen aus.
13. Wählen Sie unter Allgemeine Sicherheitsgruppen die Standardsicherheitsgruppe aus dem Drop-down-Menü aus.
14. Verwenden Sie unter Speicher konfigurieren die Drop-down-Menüs, um eine Speicherkonfiguration auszuwählen.
15. Wählen Sie im Übersichtsfenster die Option Instance starten aus.

 Note

Wenn Sie diesen Schritt abschließen, wird der Prozess zum Erstellen Ihrer EC2 Instance gestartet.

Weitere Informationen zum Erstellen eines Linux-AmazonEC2-Clients, finden Sie unter [Erste Schritte mit Amazon EC2 Linux Instances](#). Informationen zum Herstellen einer Verbindung mit einem ausgeführten Client finden Sie in den folgenden Themen:

- [Herstellen einer Verbindung mit Ihrer Linux-Instance per SSH](#)
- [Herstellung einer Verbindung zu Ihrer Linux-Instance von Windows mit PuTTY](#)

Das Amazon-EC2-Benutzerhandbuch enthält detaillierte Anweisungen zum Einrichten und Verwenden Ihrer Amazon-EC2-Instances. Die folgende Liste gibt einen Überblick über die verfügbare Dokumentation für Linux- und Windows-Amazon-EC2-Clients:

- Um einen Linux-Amazon-EC2-Client zu erstellen, lesen Sie [Erste Schritte mit Amazon-EC2-Linux-Instances](#).

Informationen zum Herstellen einer Verbindung mit einem ausgeführten Client finden Sie in den folgenden Themen:

- [Herstellen einer Verbindung mit Ihrer Linux-Instance per SSH](#)
- [Herstellung einer Verbindung zu Ihrer Linux-Instance von Windows mit PuTTY](#)
- Um einen Windows-Amazon-EC2-Client zu erstellen, lesen Sie [Erste Schritte mit Amazon-EC2-Windows-Instances](#). Weitere Informationen über das Herstellen einer Verbindung mit dem Windows-Client finden Sie unter [Verbinden mit der Windows Instanz](#).

 Note

Ihre EC2-Instance kann alle in diesem Handbuch AWS CLI enthaltenen Befehle ausführen. Wenn die AWS CLI nicht installiert ist, können Sie sie von [AWS Command Line Interface](#) herunterladen. Wenn Sie Windows verwenden, können Sie ein 32- oder 64-Bit-Windows-Installationsprogramm herunterladen. Wenn Sie Linux oder macOS verwenden, können Sie die CLI mit pip installieren.

Konfigurieren Sie die Sicherheitsgruppen der Amazon EC2 EC2-Client-Instanz für AWS CloudHSM

Als Sie eine Amazon EC2 EC2-Instanz für Ihren Cluster gestartet haben AWS CloudHSM, haben Sie sie mit einer standardmäßigen Amazon VPC-Sicherheitsgruppe verknüpft. In diesem Thema wird erläutert, wie Sie die Cluster-Sicherheitsgruppe der EC2-Instanz zuordnen. Diese Zuordnung ermöglicht es dem AWS CloudHSM Client, der auf Ihrer EC2-Instanz ausgeführt wird, mit Ihren HSMs zu kommunizieren. Um Ihre EC2-Instanz mit Ihrem AWS CloudHSM Cluster zu verbinden, müssen Sie die VPC-Standardsicherheitsgruppe ordnungsgemäß konfigurieren und die Cluster-Sicherheitsgruppe der Instanz zuordnen.

Gehen Sie wie folgt vor, um die Konfigurationsänderungen abzuschließen.

Themen

- [Schritt 1. Ändern der Standardsicherheitsgruppe](#)
- [Schritt 2. Connect die Amazon EC2 EC2-Instanz mit dem Cluster AWS CloudHSM](#)

Schritt 1. Ändern der Standardsicherheitsgruppe

Sie müssen die Standardsicherheitsgruppe so ändern, dass diese die SSH- oder RDP-Verbindung zulässt, damit Sie Clientsoftware herunterladen und installieren und mit Ihrem HSM interagieren können.

So ändern Sie die Standardsicherheitsgruppe:

1. Öffnen Sie das EC2-Dashboard unter <https://console.aws.amazon.com/ec2/>.
2. Wählen Sie Instances (running) aus und aktivieren Sie dann das Kontrollkästchen neben der EC2-Instanz, für die Sie den AWS CloudHSM Client installieren möchten.
3. Wählen Sie auf der Registerkarte Sicherheit die Sicherheitsgruppe namens Standard aus.
4. Wählen Sie oben auf der Seite Aktionen und dann Eingangsregeln bearbeiten aus.
5. Wählen Sie Regel hinzufügen aus.
6. Führen Sie für Typ eine der folgenden Aktionen aus:
 - Für eine Windows Server Amazon-EC2-Instanz wählen Sie RDP. Der Port 3389 wird automatisch ausgefüllt.

- Für eine Linux Amazon-EC2-Instance wählen Sie SSH. Der Portbereich 22 wird automatisch ausgefüllt.
7. Setzen Sie bei einer der beiden Optionen die Quelle auf Meine IP, damit Sie mit Ihrer Amazon-EC2-Instance kommunizieren können.

 Important

Geben Sie nicht 0.0.0.0/0 als CIDR-Bereich an, um zu vermeiden, dass jemand auf Ihre Instance zugreifen kann.

8. Wählen Sie Speichern.

Schritt 2. Connect die Amazon EC2 EC2-Instance mit dem Cluster AWS CloudHSM

Sie müssen die Cluster-Sicherheitsgruppe an die EC2-Instance anhängen, damit die EC2-Instance mit HSMs Ihrem Cluster kommunizieren kann. Die Cluster-Sicherheitsgruppe enthält eine vorkonfigurierte Regel, die eingehende Kommunikation über die Ports 2223-2225 zulässt.

Um die EC2-Instance mit dem Cluster zu verbinden AWS CloudHSM

1. Öffnen Sie das EC2-Dashboard unter. <https://console.aws.amazon.com/ec2/>
2. Wählen Sie Instances (running) aus und aktivieren Sie dann das Kontrollkästchen für die EC2-Instance, auf der Sie den AWS CloudHSM Client installieren möchten.
3. Wählen Sie oben auf der Seite Aktionen, Sicherheit und dann Sicherheitsgruppen ändern aus.
4. Wählen Sie die Sicherheitsgruppe mit dem Gruppennamen aus, der Ihrer Cluster-ID entspricht (z. B. `cloudhsm-cluster-<clusterID>-sg`).
5. Wählen Sie Sicherheitsgruppen hinzufügen aus.
6. Wählen Sie Speichern.

 Note

Sie können einer Amazon-EC2-Instance maximal fünf Sicherheitsgruppen zuweisen. Wenn Sie die maximale Grenze erreicht haben, müssen Sie die Standardsicherheitsgruppe der Amazon-EC2-Instance und die Cluster-Sicherheitsgruppe ändern:

Gehen Sie in der Standardsicherheitsgruppe wie folgt vor:

- Fügen Sie eine eingehende Regel hinzu, um den Datenverkehr über das TCP-Protokoll über die Ports 2223-2225 von Cluster-Sicherheitsgruppe zuzulassen.

Gehen Sie in der Cluster-Sicherheitsgruppe wie folgt vor:

- Fügen Sie eine eingehende Regel hinzu, um den Datenverkehr über das TCP-Protokoll über die Ports 2223-2225 der Standardsicherheitsgruppe zuzulassen.

Erstellen Sie ein HSM in AWS CloudHSM

Nachdem Sie einen Cluster erstellt haben AWS CloudHSM, können Sie ein Hardware-Sicherheitsmodul (HSM) erstellen. Ein HSM kann nur dann im Cluster erstellt werden, wenn sich der Cluster im nicht initialisierten Status befindet. Um den Status des Clusters zu ermitteln, rufen Sie die [Clusterseite in der AWS CloudHSM Konsole](#) auf, verwenden Sie den, AWS CLI um den [describe-clusters](#) Befehl auszuführen, oder senden Sie eine [DescribeClusters](#)Anfrage in der AWS CloudHSM API. Sie können ein HSM aus der [AWS CloudHSM -Konsole](#), der [AWS CLI](#) oder der AWS CloudHSM -API erstellen.

Important

Erstellen Sie nur ein HSM, solange sich Ihr Cluster im uninitialisierten Zustand befindet.

Console

Ein HSM erstellen (Konsole)

1. [Öffnen Sie die AWS CloudHSM Konsole zu Hause. https://console.aws.amazon.com/cloudhsm/](https://console.aws.amazon.com/cloudhsm/)
2. Aktivieren Sie das Kontrollkästchen neben dem Cluster, für den Sie ein HSM erstellen möchten.
3. Wählen Sie Aktionen aus. Wählen Sie aus dem Drop-down-Menü die Option Initialisieren.
4. Wählen Sie eine Availability Zone (AZ) für das HSM, das Sie erstellen.
5. Wählen Sie Erstellen aus.

Nachdem Sie einen Cluster und ein HSM erstellt haben, können Sie optional [die Identität des HSM überprüfen](#) oder direkt mit [Initialisieren des Clusters](#) fortfahren.

AWS CLI

Erstellen eines HSM ([AWS CLI](#))

- Führen Sie über die Eingabeaufforderung den folgenden [create-hsm](#)-Befehl aus. Geben Sie die Cluster-ID des zuvor erstellten Clusters und eine Availability Zone für das HSM an. Geben Sie die Availability Zone in der Form us-west-2a, us-west-2b, usw. an.

```
$ 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"  
  }  
}
```

Nachdem Sie einen Cluster und ein HSM erstellt haben, können Sie optional [die Identität des HSM überprüfen](#) oder direkt mit [Initialisieren des Clusters](#) fortfahren.

AWS CloudHSM API

Um ein HSM (AWS CloudHSM API) zu erstellen

- Senden Sie eine [CreateHsm](#)-Anforderung. Geben Sie die Cluster-ID des zuvor erstellten Clusters und eine Availability Zone für das HSM an.

Nachdem Sie einen Cluster und ein HSM erstellt haben, können Sie optional [die Identität des HSM überprüfen](#) oder direkt mit [Initialisieren des Clusters](#) fortfahren.

Überprüfen Sie die Identität und Authentizität des HSM Ihres Clusters in AWS CloudHSM (optional)

Um Ihren Cluster zu initialisieren AWS CloudHSM, signieren Sie eine Certificate Signing Request (CSR), die vom ersten Hardware-Sicherheitsmodul (HSM) des Clusters generiert wird. Bevor Sie dies tun, sollten Sie die Identität und Authentizität des HSM überprüfen.

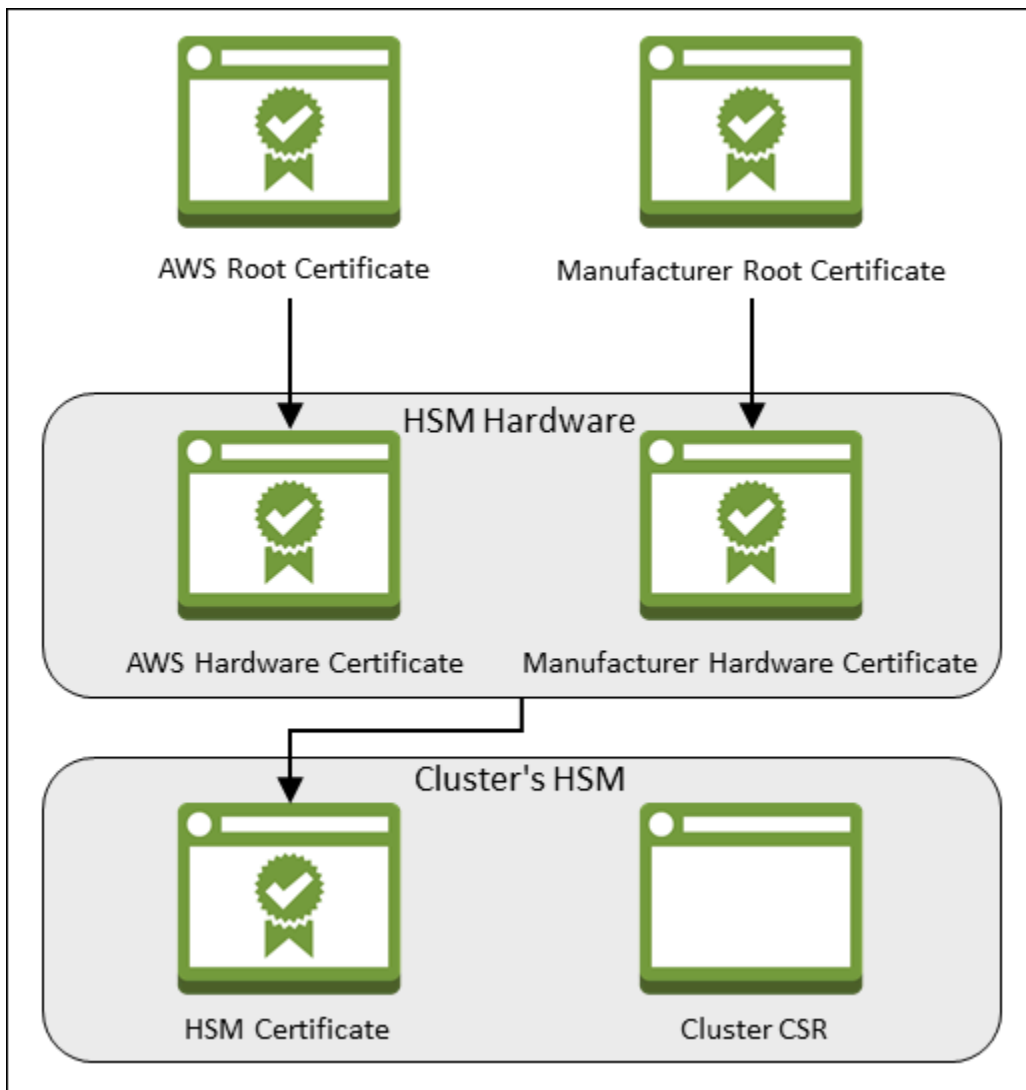
Note

Dieses Verfahren ist optional. Es funktioniert jedoch nur solange, bis ein Cluster initialisiert wird. Nach der Initialisierung des Clusters können Sie diesen Prozess nicht verwenden, um die Zertifikate abzurufen oder zu überprüfen. HSMs

Um die Identität des ersten HSM Ihres Clusters zu überprüfen, führen Sie die folgenden Schritte aus:

1. [Abrufen der Zertifikate und der CSR](#) – In diesem Schritt laden Sie drei Zertifikate und eine CSR aus dem HSM. Sie erhalten außerdem zwei Stammzertifikate, eines vom AWS CloudHSM und eines vom HSM-Hardwarehersteller.
2. [Überprüfen Sie die Zertifikatsketten](#) — In diesem Schritt erstellen Sie zwei Zertifikatsketten, eine für das AWS CloudHSM Stammzertifikat und eine für das Stammzertifikat des Herstellers. Anschließend verifizieren Sie das HSM-Zertifikat anhand dieser Zertifikatsketten, um festzustellen, dass AWS CloudHSM sowohl das HSM-Zertifikat als auch der Hardwarehersteller die Identität und Echtheit des HSM bestätigen.
3. [Vergleichen öffentlicher Schlüssel](#) – In diesem Schritt extrahieren und vergleichen Sie den öffentlichen Schlüssel im HSM-Zertifikat und die Cluster-CSR, um sicherzustellen, dass sie identisch sind. Dadurch sollten Sie das Vertrauen erhalten, dass die CSR von einem authentischen, vertrauenswürdigen HSM generiert wurde.

Das folgende Diagramm zeigt die CSR, die Zertifikate und ihre Beziehung zueinander. In der folgenden Liste sind alle Zertifikate definiert.



AWS Stammzertifikat

Das ist AWS CloudHSM das Stammzertifikat.

Hersteller-Stammzertifikat

Dies ist das Stammzertifikat des Hardwareherstellers.

AWS Hardware-Zertifikat

AWS CloudHSM hat dieses Zertifikat erstellt, als die HSM-Hardware zur Flotte hinzugefügt wurde. Dieses Zertifikat bestätigt, dass der AWS CloudHSM Eigentümer der Hardware ist.

Hersteller-Hardwarezertifikat

Der HSM-Hardwarehersteller hat dieses Zertifikat erstellt, als er die HSM-Hardware hergestellt hat. Dieses Zertifikat versichert, dass die Hersteller die Hardware erstellt hat.

HSM-Zertifikat

Das HSM-Zertifikat wird mit der FIPS-validierten Hardware erstellt, wenn Sie das erste HSM im Cluster anlegen. Dieses Zertifikat versichert, dass die HSM-Hardware das HSM erstellt hat.

Cluster-CSR

Das erste HSM erstellt die Cluster-CSR. Wenn Sie [die Cluster-CSR signieren](#), beanspruchen Sie den Cluster für sich. Anschließend können Sie die signierte CSR nutzen, um den [Cluster zu initialisieren](#).

Schritt 1. Zertifikate aus dem HSM laden

Um die Identität und Authentizität Ihres HSM zu überprüfen, laden Sie zuerst eine CSR und fünf Zertifikate. Sie erhalten drei der Zertifikate vom HSM, was Sie mit der [AWS CloudHSM Konsole](#), der [AWS Command Line Interface \(AWS CLI\)](#) oder der AWS CloudHSM API tun können.

Console

Laden der CSR und der Zertifikate (Konsole)

1. Öffnen Sie die AWS CloudHSM Konsole zu <https://console.aws.amazon.com/cloudhsm/Hause>.
2. Wählen Sie die Optionsschaltfläche neben der Cluster-ID mit dem HSM, das Sie überprüfen möchten.
3. Wählen Sie Aktionen aus. Wählen Sie aus dem Drop-down-Menü die Option Initialisieren.
4. Wenn Sie den [vorherigen Schritt](#) zur Erstellung eines HSM nicht abgeschlossen haben, wählen Sie eine Availability Zone (AZ) für das HSM, das Sie erstellen. Wählen Sie dann Erstellen aus.
5. Wenn die Zertifikate und die CSR bereit sind, finden Sie Links, um sie herunterzuladen.

Certificate signing request

To initialize the cluster, you must download a certificate signing request (CSR) and then [sign it](#) .

 **Cluster CSR**

Cluster verification certificate

Optionally, you may wish to download the HSM certificate below which generated this Cluster CSR and [verify its authenticity](#) .

 **HSM certificate**

6. Klicken Sie auf die einzelnen Links, um die CSR und Zertifikate herunterzuladen und zu speichern. Zur Vereinfachung der nachfolgenden Schritte speichern Sie alle Dateien in demselben Verzeichnis und nutzen die Standard-Dateinamen.

AWS CLI

Abrufen der CSR und der HSM-Zertifikate ([AWS CLI](#))

- Führen Sie über die Eingabeaufforderung den Befehl [describe-clusters](#) viermal aus, extrahieren Sie jedes Mal die CSR und die verschiedenen Zertifikate und speichern Sie sie in Dateien.

- a. Geben Sie zum Extrahieren der Cluster-CSR den folgenden Befehl aus. *<cluster ID>* Ersetzen Sie sie durch die ID des Clusters, den Sie zuvor erstellt haben.

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

- b. Geben Sie zum Extrahieren des HSM-Zertifikats den folgenden Befehl aus. *<cluster ID>* Ersetzen Sie durch die ID des Clusters, den Sie zuvor erstellt haben.

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

- c. Geben Sie den folgenden Befehl ein, um das AWS Hardwarezertifikat zu extrahieren. *<cluster ID>* Ersetzen Sie es durch die ID des Clusters, den Sie zuvor erstellt haben.

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

- d. Geben Sie zum Extrahieren des Hardwarezertifikats des Herstellers den folgenden Befehl aus. *<cluster ID>* Ersetzen Sie durch die ID des Clusters, den Sie zuvor erstellt haben.

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


AWS CloudHSM API

Um die CSR- und HSM-Zertifikate (AWS CloudHSM API) zu erhalten

- Senden Sie eine [DescribeClusters](#)-Anforderung und extrahieren und speichern dann die CSR und die Zertifikate aus der Antwort.

Schritt 2. Die Stammzertifikate laden

Gehen Sie wie folgt vor, um die Stammzertifikate für AWS CloudHSM und den Hersteller zu erhalten. Speichern Sie die Stammzertifikatsdateien in das Verzeichnis, das die CSR und die HSM-Zertifikatsdateien enthält.

Um die Stammzertifikate AWS CloudHSM und die Stammzertifikate des Herstellers zu erhalten

1. Laden Sie das AWS CloudHSM Stammzertifikat herunter: [AWS_CloudHSM_Root-G1.zip](#)
2. Laden Sie das richtige Stammzertifikat des Herstellers für Ihren HSM-Typ herunter:
 - [Stammzertifikat des Herstellers hsm1.medium: liquid_security_certificate.zip](#)
 - [Stammzertifikat des Herstellers hsm2m.medium: liquid_security_certificate.zip](#)

Note

Verwenden Sie die folgenden Links, um jedes Zertifikat von der jeweiligen Landingpage herunterzuladen:

- [Landingpage für das Stammzertifikat des Herstellers von hsm1.medium](#)
- [Landingpage für das Herstellerstammzertifikat von hsm2m.medium](#)

Möglicherweise müssen Sie mit der rechten Maustaste auf den Link Download Certificate (Zertifikat herunterladen) klicken und dann Save Link As... (Link speichern unter) wählen, um die Zertifikatsdatei zu speichern.

3. Nachdem Sie die Dateien heruntergeladen haben, extrahieren (entpacken) Sie den Inhalt.

Schritt 3. Zertifikatketten prüfen

In diesem Schritt erstellen Sie zwei Zertifikatsketten, eine für das AWS CloudHSM Stammzertifikat und eine für das Stammzertifikat des Herstellers. Anschließend verwenden Sie OpenSSL, um das HSM-Zertifikat mit jeder Zertifikatkette zu verifizieren.

Öffnen Sie zum Erstellen der Zertifikatketten eine Linux-Shell. Sie benötigen OpenSSL (in den meisten Linux-Shells verfügbar) und Sie benötigen das [Stammzertifikat](#) und die [HSM-Zertifikatsdateien](#), die Sie heruntergeladen haben. Sie benötigen die AWS CLI für diesen Schritt jedoch nicht, und die Shell muss nicht mit Ihrem AWS Konto verknüpft werden.

Um das HSM-Zertifikat mit dem AWS CloudHSM Stammzertifikat zu verifizieren

1. Navigieren Sie zu dem Verzeichnis, in dem das [Stammzertifikat](#) und die [HSM-Zertifikatsdateien](#) gespeichert sind, die Sie heruntergeladen haben. Die folgenden Befehle gehen davon aus, dass sich alle Zertifikate im aktuellen Verzeichnis befinden und die Standard-Dateinamen verwendet werden.

Verwenden Sie den folgenden Befehl, um eine Zertifikatskette zu erstellen, die das AWS Hardwarezertifikat und das AWS CloudHSM Stammzertifikat in dieser Reihenfolge umfasst. **<cluster ID>** Ersetzen Sie es durch die ID des Clusters, den Sie zuvor erstellt haben.

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

2. Verwenden Sie den folgenden OpenSSL-Befehl, um das HSM-Zertifikat anhand der AWS - Zertifikatkette zu überprüfen. **<cluster ID>** Ersetzen Sie durch die ID des Clusters, den Sie zuvor erstellt haben.

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

Das HSM-Zertifikat anhand des Hersteller-Stammzertifikats überprüfen

1. Verwenden Sie den folgenden Befehl zum Erstellen einer Zertifikatkette, die das Hersteller-Hardware-Zertifikat und das Hersteller-Stammzertifikat (in dieser Reihenfolge) enthält. **<cluster ID>** Ersetzen Sie durch die ID des Clusters, den Sie zuvor erstellt haben.

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

2. Verwenden Sie den folgenden Befehl, um das HSM-Zertifikat anhand der Hersteller-Zertifikatkette zu überprüfen. *<cluster ID>* Ersetzen Sie durch die ID des Clusters, den Sie zuvor erstellt haben.

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

Schritt 4. Extrahieren und Vergleichen der öffentlichen Schlüssel

Verwenden Sie OpenSSL, um die öffentlichen Schlüssel im HSM-Zertifikat und die Cluster-CSR zu extrahieren und zu vergleichen, um sicherzustellen, dass sie identisch sind.

Nutzen Sie zum Vergleichen der öffentlichen Schlüssel die Linux-Shell. Sie benötigen OpenSSL, das in den meisten Linux-Shells verfügbar ist, aber Sie benötigen das AWS CLI für diesen Schritt nicht. Die Shell muss nicht mit Ihrem AWS Konto verknüpft sein.

Extrahieren und Vergleichen der öffentlichen Schlüssel

1. Um den öffentlichen Schlüssel aus dem HSM-Zertifikat zu extrahieren, führen Sie den folgenden Befehl aus.

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

2. Um den öffentlichen Schlüssel aus der Cluster-CSR zu extrahieren, führen Sie den folgenden Befehl aus.

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

3. Verwenden Sie den folgenden Befehl, um die öffentlichen Schlüssel zu vergleichen. Wenn die öffentlichen Schlüssel identisch sind, erzeugt der folgende Befehl keine Ausgabe.

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

Nach dem Überprüfen der Identität und Authentizität des HSM fahren Sie mit fort [Initialisieren des Clusters](#).

Initialisieren Sie den Cluster in AWS CloudHSM

Nachdem Sie Ihren Cluster erstellt und Ihr Hardware-Sicherheitsmodul (HSM) hinzugefügt haben AWS CloudHSM, können Sie den Cluster initialisieren. Führen Sie die Schritte in der folgenden Themen durch, um Ihr -Cluster zu initialisieren.

Note

Bevor Sie den Cluster initialisieren, überprüfen Sie den Prozess, mit dem Sie die [Identität und Authentizität des Clusters überprüfen](#) können. HSMs Dieser Prozess ist optional und funktioniert nur, bis ein Cluster initialisiert wird. Nach der Initialisierung des Clusters können Sie diesen Prozess nicht verwenden, um Ihre Zertifikate abzurufen oder zu überprüfen. HSMs

Themen

- [-Übersicht](#)
- [Schritt 1. Anfordern der Cluster-CSR](#)
- [Schritt 2. Erstellen Sie einen privaten Schlüssel für Ihre Root-CA](#)
- [Schritt 3. Signieren der CSR](#)
- [Schritt 4. Initialisieren des Clusters](#)

-Übersicht

Der Cluster-Initialisierungsprozess legt fest, dass Sie Eigentümer und Kontrolle über den Cluster haben, und zwar HSMs über ein zertifikatsbasiertes Authentifizierungssystem. Dieser Prozess beweist kryptografisch, dass Sie der alleinige Eigentümer des HSMs Clusters sind, und schafft die Vertrauensbasis, die für alle future Verbindungen zu Ihrem erforderlich ist. HSMs

Auf dieser Seite erfahren Sie, wie Sie Folgendes tun können:

- Rufen Sie die Zertifikatsignieranforderung (CSR) Ihres Clusters ab.
- Generieren und verwenden Sie die privaten Schlüssel, um ein selbstsigniertes Stammzertifikat oder eine Zertifikatskette zu erstellen.
- Signieren Sie die CSR Ihres Clusters, um ein signiertes HSM-Zertifikat zu erstellen.
- Initialisieren Sie Ihren Cluster mithilfe des signierten HSM-Zertifikats und des selbstsignierten Zertifikats oder der Zertifikatskette.

Wenn Sie bereit sind, sehen Sie sich [Schritt 1. Anfordern der Cluster-CSR](#) an.

Schritt 1. Anfordern der Cluster-CSR

Bevor Sie den Cluster initialisieren, müssen Sie eine Zertifikatsignieranforderung (CSR, Certificate Signing Request) herunterladen und signieren, die vom ersten HSM des Clusters erstellt wurde. Wenn Sie die Schritte ausgeführt haben, um [die Identität des HSM Ihres Clusters zu überprüfen](#), befinden Sie sich bereits im Besitz der CSR und können sie signieren. Andernfalls rufen Sie die CSR jetzt mithilfe der [AWS CloudHSM Konsole](#), der [AWS Command Line Interface \(AWS CLI\)](#) oder der [API](#) ab. AWS CloudHSM

Console

Die CSR laden (Konsole)

1. Öffne die AWS CloudHSM Konsole zu <https://console.aws.amazon.com/cloudhsm/Hause>.
2. Wählen Sie die Optionsschaltfläche neben der Cluster-ID mit dem HSM, das Sie überprüfen möchten.
3. Wählen Sie Aktionen aus. Wählen Sie aus dem Drop-down-Menü die Option Initialisieren.
4. Wenn Sie den [vorherigen Schritt](#) zur Erstellung eines HSM nicht abgeschlossen haben, wählen Sie eine Availability Zone (AZ) für das HSM, das Sie erstellen. Wählen Sie dann Erstellen aus.
5. Wenn der CSR bereit ist, sehen Sie einen Link, über den Sie sie herunterladen können.

Certificate signing request

To initialize the cluster, you must download a certificate signing request (CSR) and then [sign it](#) .

 **Cluster CSR**

Cluster verification certificate

Optionally, you may wish to download the HSM certificate below which generated this Cluster CSR and [verify its authenticity](#) .

 **HSM certificate**

6. Wählen Sie Cluster-CSR, um die CSR herunterzuladen und zu speichern.

AWS CLI

Abrufen der CSR ([AWS CLI](#))

- Führen Sie an der Eingabeaufforderung den folgenden [describe-clusters](#)-Befehl aus, der die CSR extrahiert und in einer Datei speichert. *<cluster ID>* Ersetzen Sie es durch die ID des Clusters, den Sie [zuvor erstellt haben](#).

```
$ aws cloudhsmv2 describe-clusters --filters clusterIds=<cluster ID> \  
--output text \  
--text-wrap=never
```

```
\
--query 'Clusters[].Certificates.ClusterCsr'

> <cluster ID>_ClusterCsr.csr
```

AWS CloudHSM API

Um die CSR (AWS CloudHSM API) zu erhalten

1. Senden Sie eine [DescribeClusters](#)-Anforderung.
2. Extrahieren und speichern Sie die CSR aus der Antwort.

Schritt 2. Erstellen Sie einen privaten Schlüssel für Ihre Root-CA

Note

Für einen Produktionscluster sollte der Schlüssel, den Sie erstellen wollen, auf sichere Weise mit einer vertrauenswürdigen Zufallsquelle erstellt werden. Wir empfehlen, dass Sie ein sicheres standortexternes und Offline-HSM oder etwas Äquivalentes verwenden. Speichern Sie den Schlüssel sicher. Der Schlüssel legt die Identität des Clusters und Ihre alleinige Kontrolle über die HSMs darin enthaltenen Elemente fest.

Für die Entwicklung und für Tests können Sie ein beliebiges Tool (z. B. OpenSSL) verwenden, um das Cluster-Zertifikat zu erstellen und zu signieren. Im folgenden Beispiel wird gezeigt, wie Sie einen Schlüssel erstellen. Nachdem Sie den Schlüssel verwendet haben, um ein selbstsigniertes Zertifikat zu erstellen (siehe unten), sollten Sie ihn sicher speichern. Um sich bei Ihrer AWS CloudHSM Instance anzumelden, muss das Zertifikat vorhanden sein, der private Schlüssel jedoch nicht.

In der folgenden Tabelle sind die unterstützten Algorithmen, Schlüsselgrößen und Kurven für die Zertifikatsgenerierung aufgeführt.

Algorithmen	Größe/Kurven
RSA 5. PKCSv1	2048, 3072, 4096
RSA-PSS	2048, 3072, 4096

Algorithmen	Größe/Kurven
ECDSA	prime256v1, secp384r1, secp521r1
Digest	SHA-224, SHA-256, SHA-384 und SHA-512

Verwenden Sie den folgenden Beispielbefehl, um einen privaten Schlüssel für Ihre selbstsignierte Root-CA zu erstellen.

```
$ 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:
```

Schritt 3. Signieren der CSR

In den vorherigen Schritten haben Sie die CSR Ihres Clusters abgerufen und einen privaten Schlüssel für Ihre Stammzertifizierungsstelle erstellt. In diesem Schritt verwenden Sie Ihren privaten Schlüssel, um ein Signaturzertifikat zu generieren, mit dem Sie die CSR Ihres Clusters signieren können. Die folgenden Themen führen Sie durch den Prozess der Erstellung eines einzelnen selbstsignierten Zertifikats oder einer Zertifikatskette mit OpenSSL. Sie benötigen das AWS CLI für diesen Schritt nicht, und die Shell muss nicht mit Ihrem AWS Konto verknüpft werden.

Important

Um Ihren Cluster zu initialisieren, muss Ihr Trust Anchor [RFC 5280](#) entsprechen und die folgenden Anforderungen erfüllen:

- Wenn Sie X509v3-Erweiterungen verwenden, muss die X509v3-Erweiterung Basic Constraints vorhanden sein.
- Der Trust Anchor muss ein selbstsigniertes Zertifikat sein.
- Erweiterungswerte dürfen nicht miteinander in Konflikt stehen.

Wählen Sie eine der folgenden Methoden, um die CSR Ihres Clusters zu signieren:

Wählen Sie Ihren Zertifikatsansatz

Sie müssen einen der folgenden beiden Ansätze wählen. Schließen Sie nicht beide Ansätze ab.

Option A: Einzelnes selbstsigniertes Zertifikat

Erstellen Sie ein einzelnes selbstsigniertes Stammzertifikat, um die CSR Ihres Clusters zu signieren. Dies ist die einfachste und direkteste Methode, um Vertrauen aufzubauen.

Empfohlen für:

- Umgebungen, in denen keine externe PKI erforderlich ist
- Test- und Entwicklungsumgebungen, in denen Einfachheit bevorzugt wird

Gehe zu: [Erstellen Sie ein einzelnes selbstsigniertes Zertifikat](#)

Option B: Zertifikatskette mit zwischengeschalteter CA

Erstellen Sie eine Zertifikatskette mithilfe einer Zwischenzertifizierungsstelle. Eine Zwischenzertifikatskette bietet mehr Sicherheit, Skalierbarkeit und Flexibilität, da Stammzertifizierungsstellen (CAs) offline bleiben können, während die Zertifikatsausstellung an eine Zwischenzertifizierungsstelle delegiert wird CAs, wodurch das Risiko einer Beeinträchtigung der Stammzertifizierungsstelle verringert wird.

Empfohlen für:

- Umgebungen, in denen eine externe PKI erforderlich ist
- Integration mit AWS Private Certificate Authority (PCA)

Beispiel für eine AWS PCA-Integration: Sie können AWS Private Certificate Authority verwenden, um Ihre CA-Zwischenzertifikate zu erstellen und zu verwalten. Dies ermöglicht ein automatisiertes Lebenszyklusmanagement von Zertifikaten, einschließlich Verlängerung und Widerruf, und bietet gleichzeitig die Sicherheitsvorteile, die sich aus dem Offline-Betrieb Ihrer Stammzertifizierungsstelle ergeben. Weitere Informationen zu AWS PCA finden Sie im [AWS Private Certificate Authority User Guide](#).

Gehen Sie zu: [Erstellen Sie eine ICA-Kette \(Intermediate Certificate Authority\)](#)

Erstellen Sie ein einzelnes selbstsigniertes Zertifikat

Die vertrauenswürdige Hardware, die Sie verwenden, um den privaten Schlüssel für Ihre Produktions-Cluster zu erstellen, sollte auch ein Software-Tool bereitstellen, um unter Verwendung dieses Schlüssels ein selbstsigniertes Zertifikat zu generieren. Das folgende Beispiel verwendet OpenSSL und den privaten Schlüssel, den Sie im vorherigen Schritt erstellt haben, um ein selbstsigniertes Root-CA-Signaturzertifikat zu erstellen. Das Zertifikat gilt für 10 Jahre (3652 Tage). Lesen Sie die Anweisungen auf dem Bildschirm und befolgen Sie die Eingabeaufforderungen.

```
$ 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 []:
```

Mit diesem Befehl wird ein Zertifikat mit dem Namen `customerRootCA.crt` erstellt. Legen Sie dieses Zertifikat auf jedem Host ab, von dem aus Sie eine Verbindung zu Ihrem AWS CloudHSM Cluster herstellen möchten. Wenn Sie der Datei einen anderen Namen geben oder unter einem anderen Pfad als dem Root-Verzeichnis speichern, müssen Sie Ihre Client-Konfigurationsdatei entsprechend anpassen. Verwenden Sie das Zertifikat und den privaten Schlüssel, die Sie gerade erstellt haben, um die Cluster-Zertifikatssignierungsanforderung (Certificate Signing Request, CSR) im nächsten Schritt zu signieren.

Signieren Sie die Cluster-CSR mit Ihrer selbstsignierten Root-CA

Die vertrauenswürdige Hardware, die Sie verwenden, um Ihren privaten Schlüssel für Ihre Produktions-Cluster zu erstellen, sollte auch ein Tool bereitstellen, um die CSR mit diesem Schlüssel zu signieren. In dem folgenden Beispiel wird OpenSSL zum Signieren der Cluster-CSR verwendet. Der folgende Beispielbefehl signiert die CSR mit der selbstsignierten `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:
```

Mit diesem Befehl wird eine Datei mit dem Namen **<cluster ID>_CustomerHsmCertificate.crt** erstellt. Verwenden Sie diese Datei als signiertes Zertifikat, wenn Sie den Cluster initialisieren.

Überprüfen Sie das signierte Zertifikat anhand der Stammzertifizierungsstelle (optional):

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

Nachdem Sie das signierte HSM-Zertifikat mit Ihrer selbstsignierten Root-CA erstellt haben, gehen Sie zu. [Schritt 4. Initialisieren des Clusters](#)

Erstellen Sie eine ICA-Kette (Intermediate Certificate Authority)

Die folgenden Beispiele führen Sie durch die Erstellung einer Zertifikatskette der Länge 2, bestehend aus einer Stammzertifizierungsstelle (CA) und einer Zwischenzertifizierungsstelle. Sie erstellen zunächst ein selbstsigniertes Stammzertifizierungsstellenzertifikat und dann eine Zwischenzertifizierungsstelle, die von der Stammzertifizierungsstelle signiert ist. Schließlich verwenden Sie die Zwischenzertifizierungsstelle, um die CSR Ihres Clusters zu signieren und so eine vollständige Vertrauenskette von Ihrem HSM-Zertifikat zurück zur Stammzertifizierungsstelle zu erstellen. Dieser Ansatz bietet mehr Sicherheit, da die Stammzertifizierungsstelle offline bleibt, während die Zwischenzertifizierungsstelle für day-to-day Zertifikatsoperationen verwendet wird.

Important

Um Ihren Cluster mit einer Zertifikatskette zu initialisieren, muss Ihre Kette die folgenden Anforderungen erfüllen:

- Die Kette muss geordnet werden, beginnend mit der Zwischenzertifizierungsstelle, die die Cluster-CSR signiert. In dieser Reihenfolge sollte die erste ICA einen Emittenten haben, der dem Betreff der nächsten ICA in der Kette entspricht, usw.
- Nur die Stammzertifizierungsstelle sollte selbstsigniert sein, was bedeutet, dass Emittent und Betreff identisch sein sollten.
- Die Kette darf aus nicht mehr als 4 Zertifikaten bestehen (einschließlich der Root-CA am Ende), und die Gesamtgröße der Kette darf 16 KB (Kilobyte) nicht überschreiten.
- Alle Zertifizierungsstellen (CAs) sollten den [RFC 5280](#)-Richtlinien entsprechen.

Dieser Abschnitt enthält Beispiele für die Erstellung einer zwischengeschalteten Zertifizierungsstellenkette mit zwei verschiedenen Ansätzen: OpenSSL für die lokale Zertifikatsgenerierung und AWS Private Certificate Authority (PCA) für verwaltete Zertifikatsdienste. Wählen Sie den Ansatz, der am besten zu Ihrer Umgebung und Ihren Sicherheitsanforderungen passt.

Note

Die folgenden Beispiele sind allgemeine Anwendungsfälle und sind beide vereinfacht, wobei die einfachste Konfiguration verwendet wird. Sehen Sie sich für Produktionsumgebungen die zusätzlichen Konfigurationsoptionen und Sicherheitsanforderungen an, die für Ihren Anwendungsfall spezifisch sind.

OpenSSL

Erstellen Sie eine OpenSSL-Konfigurationsdatei mit gängigen v3-Erweiterungen für CA:

```
$ 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
```

Generieren Sie eine selbstsignierte Root-CA mit OpenSSL:

```
$ 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 []:
```

Generieren Sie einen zwischenzeitlichen CA-Schlüssel:

```
$ 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:
```

Erstellen Sie die Intermediate CA Certificate Signing Request (CSR):

```
$ 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 []:

```

Erstellen Sie mit der selbstsignierten Root-CA das Intermediate-CA-Zertifikat:

```

$ 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=

```

Kombinieren Sie die Zertifikate zu einer Kettendatei:

```

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

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

```

```
-----END CERTIFICATE-----
```

Signieren Sie die Cluster-CSR mit Ihrer Zwischen-CA:

```
$ 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

Erstellen und aktivieren Sie eine Root-CA mit AWS Private Certificate Authority:

```
$ # 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

```

Erstellen und aktivieren Sie eine untergeordnete Zertifizierungsstelle (auch als Zwischenzertifizierungsstelle bezeichnet):

```

$ # 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

```

Kombinieren Sie die Zertifikate zu einer Kettendatei:

```

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

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

```

Signieren Sie die Cluster-CSR mit AWS PCA:

```

$ 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>"
```

Laden Sie das signierte Cluster-Zertifikat herunter:

```
$ 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
```

Mit diesem Befehl wird eine Datei mit dem Namen `<cluster ID>_CustomerHsmCertificate.crt` erstellt. Verwenden Sie diese Datei als signiertes Zertifikat, wenn Sie den Cluster initialisieren.

Überprüfen Sie das signierte Zertifikat anhand der Zertifikatskette (optional):

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

Nachdem Sie das signierte HSM-Zertifikat mit Ihrer Zwischenzertifizierungsstelle erstellt haben, gehen Sie zu [Schritt 4. Initialisieren des Clusters](#).

Schritt 4. Initialisieren des Clusters

Verwenden Sie das signierte HSM-Zertifikat und Ihr Signaturzertifikat, um den Cluster zu initialisieren. Sie können die [AWS CloudHSM Konsole AWS CLI](#), die oder die AWS CloudHSM API verwenden.

Console

Initialisieren eines Clusters (Konsole)

1. Öffnen Sie die AWS CloudHSM Konsole zu <https://console.aws.amazon.com/cloudhsm/Hause>.

2. Wählen Sie die Optionsschaltfläche neben der Cluster-ID mit dem HSM, das Sie überprüfen möchten.
3. Wählen Sie Aktionen aus. Wählen Sie aus dem Drop-down-Menü die Option Initialisieren.
4. Wenn Sie den [vorherigen Schritt](#) zur Erstellung eines HSM nicht abgeschlossen haben, wählen Sie eine Availability Zone (AZ) für das HSM, das Sie erstellen. Wählen Sie dann Erstellen aus.
5. Wählen Sie auf der Seite Herunterladen der Zertifikat-Signierungsanforderung Weiter aus. Wenn die Schaltfläche Weiter nicht verfügbar ist, wählen Sie zuerst einen der CSR- oder Zertifikat-Links. Wählen Sie anschließend Weiter.
6. Wählen Sie auf der Seite Zertifikatsignierungsanforderung (CSR) signieren Weiter.
7. Gehen Sie auf der Seite Die Zertifikate hochladen wie folgt vor:
 - a. Wählen Sie neben Cluster-Zertifikat Datei hochladen. Suchen Sie anschließend das HSM-Zertifikat, das Sie zuvor signiert haben, und wählen Sie es aus. Wenn Sie die Schritte im vorigen Abschnitt ausgeführt haben, wählen Sie die Datei `<cluster ID>_CustomerHsmCertificate.crt`.
 - b. Wählen Sie neben Zertifikat ausstellen Datei hochladen. Wählen Sie dann Ihr Signaturzertifikat auf der Grundlage des von Ihnen gewählten Ansatzes aus:
 - Wenn Sie Option A (einzelnes selbstsigniertes Zertifikat) gewählt haben: Wählen Sie die Datei mit dem Namen `<customerRootCA>.crt`
 - Wenn Sie Option B (Zertifikatskette) gewählt haben: Wählen Sie die Datei mit dem Namen aus `<chainCA>.crt`
 - c. Wählen Sie Hochladen und initialisieren.

AWS CLI

So initialisieren Sie einen Cluster ([AWS CLI](#))

- Führen Sie über die Eingabeaufforderung den folgenden [initialize-cluster](#)-Befehl aus. Stellen Sie Folgendes bereit:
 - Die ID des Clusters, den Sie zuvor erstellt haben.
 - Das HSM-Zertifikat, das Sie zuvor signiert haben. Wenn Sie die Schritte im vorigen Abschnitt ausgeführt haben, wurde es in der Datei `<cluster ID>_CustomerHsmCertificate.crt` gespeichert.

- Ihr Signaturzertifikat basiert auf dem von Ihnen gewählten Ansatz:
 - Wenn Sie Option A (einzelnes selbstsigniertes Zertifikat) gewählt haben: Verwenden Sie die Datei mit dem Namen `<customerRootCA>.cert`
 - Wenn Sie Option B (Zertifikatskette) gewählt haben: Verwenden Sie die Datei mit dem Namen `<chainCA>.cert`

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

AWS CloudHSM API

Um einen Cluster (AWS CloudHSM API) zu initialisieren

- Senden Sie eine [InitializeCluster](#)-Anforderung mit folgendem Inhalt:
 - Die ID des Clusters, den Sie zuvor erstellt haben.
 - Das HSM-Zertifikat, das Sie zuvor signiert haben. Wenn Sie die Schritte im vorigen Abschnitt ausgeführt haben, wurde es in der Datei `<cluster ID>_CustomerHsmCertificate.cert` gespeichert.
 - Ihr Signaturzertifikat basiert auf dem von Ihnen gewählten Ansatz:
 - Wenn Sie Option A (einzelnes selbstsigniertes Zertifikat) gewählt haben: Verwenden Sie die Datei mit dem Namen `<customerRootCA>.cert`
 - Wenn Sie Option B (Zertifikatskette) gewählt haben: Verwenden Sie die Datei mit dem Namen `<chainCA>.cert`

CloudHSM-CLI installieren und konfigurieren

Um mit dem HSM in Ihrem AWS CloudHSM Cluster zu interagieren, benötigen Sie die CloudHSM-CLI.

Connect zu Ihrer Client-Instance her und führen Sie die folgenden Befehle aus, um die AWS CloudHSM Befehlszeilentools herunterzuladen und zu installieren. Weitere Informationen finden Sie unter [Starten Sie eine Amazon EC2 EC2-Client-Instance für die Interaktion mit AWS CloudHSM](#).

Amazon Linux 2023

Amazon Linux 2023 auf x86_64-Architektur:

```
$ 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
```

Amazon Linux 2023 zur ARM64 Architektur:

```
$ 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

Amazon Linux 2 auf x86_64-Architektur:

```
$ 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
```

Amazon Linux 2 zur ARM64 Architektur:

```
$ 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+)

RHEL 10 auf der x86_64-Architektur:

```
$ 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
```

RHEL 10 zur Architektur: ARM64

```
$ 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+)

RHEL 9 auf der x86_64-Architektur:

```
$ 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
```

RHEL 9 zur Architektur: ARM64

```
$ 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+)

RHEL 8 auf x86_64-Architektur:

```
$ 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
```

RHEL 8 zur ARM64 Architektur:

```
$ 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 24.04 LTS

Ubuntu 24.04 LTS auf der x86_64-Architektur:

```
$ 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
```

Ubuntu 24.04 LTS zur Architektur: ARM64

```
$ 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

Ubuntu 22.04 LTS auf der x86_64-Architektur:

```
$ 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
```

Ubuntu 22.04 LTS zur Architektur: ARM64

```
$ 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

Öffnen Sie Windows Server 2022 auf einer x86_64-Architektur PowerShell als Administrator und führen Sie den folgenden Befehl aus:

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

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

Windows Server 2019

Öffnen Sie Windows Server 2019 auf einer x86_64-Architektur PowerShell als Administrator und führen Sie den folgenden Befehl aus:

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

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

Windows Server 2016

Öffnen Sie Windows Server 2016 auf einer x86_64-Architektur PowerShell als Administrator und führen Sie den folgenden Befehl aus:

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

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


Verwenden Sie die folgenden Befehle, um CloudHSM-CLI zu konfigurieren.

Um eine EC2 Linux-Instanz für Client SDK 5 zu booten

- Verwenden Sie das Configure-Tool, um die IP-Adresse der HSM(s) in Ihrem Cluster anzugeben.

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

Um eine EC2 Windows-Instanz für das Client SDK 5 zu booten

- Verwenden Sie das Configure-Tool, um die IP-Adresse der HSM(s) in Ihrem Cluster anzugeben.

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

Aktiviere den Cluster in AWS CloudHSM

Wenn Sie einen AWS CloudHSM Cluster aktivieren, ändert sich der Status des Clusters von initialisiert zu aktiv. Sie können dann die [Benutzer des Hardware-Sicherheitsmoduls \(HSM\) verwalten](#) und [das HSM verwenden](#).

Important

Bevor Sie den Cluster aktivieren können, müssen Sie zunächst das ausstellende Zertifikat auf jeder EC2 Instance, die eine Verbindung zum Cluster herstellt, in den Standardspeicherort für die Plattform kopieren (Sie erstellen das ausstellende Zertifikat, wenn Sie den Cluster initialisieren). Verwenden Sie die entsprechende Zertifikatsdatei, die auf dem Ansatz basiert, den Sie bei der Cluster-Initialisierung gewählt haben:

- Wenn Sie Option A (einzelnes selbstsigniertes Zertifikat) gewählt haben: Kopieren `customerRootCA.crt`
- Wenn Sie Option B (Zertifikatskette) gewählt haben: Kopieren `chainCA.crt`

Linux-Speicherort:

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

Windows-Standort:

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

Nachdem Sie die Zertifikatsdatei kopiert haben, bearbeiten Sie die `/opt/cloudhsm/etc/cloudhsm-cli.cfg` Datei, um sicherzustellen, dass der Name der Zertifikatsdatei mit dem Namen des CA-Zertifikats übereinstimmt, das Sie kopiert haben.

Nachdem Sie das ausstellende Zertifikat platziert haben, installieren Sie die CloudHSM-CLI und führen Sie den [cluster activate](#)-Befehl auf Ihrem ersten HSM aus. Sie werden feststellen, dass das Administratorkonto auf dem ersten HSM in Ihrem Cluster die Rolle [unactivated-admin](#) hat. Dies ist eine temporäre Rolle, die nur vor der Clusteraktivierung existiert. Wenn Sie Ihren Cluster aktivieren, ändert sich die Rolle „unactivated-admin“ in „admin“.

Aktivieren eines Clusters

1. Stellen Sie eine Verbindung zu der Client-Instance her, die Sie zuvor gestartet haben. Weitere Informationen finden Sie unter [Starten Sie eine Amazon EC2 EC2-Client-Instance für die Interaktion mit AWS CloudHSM](#). Sie können eine Linux-Instance oder Windows Server starten.
2. Führen Sie die CloudHSM-CLI im interaktiven Modus aus.

Linux

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

Windows

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

3. (Optional) Verwenden Sie den Befehl `user list`, um die vorhandenen Benutzer anzuzeigen.

```
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. Verwenden Sie den `cluster activate`-Befehl, um das anfängliche Admin-Passwort festzulegen.

```

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

```

Wir empfehlen, das neue Passwort auf einem Passwort-Arbeitsblatt zu notieren. Verlieren Sie das Arbeitsblatt nicht. Wir empfehlen, eine Kopie des Passwort-Arbeitsblatts auszudrucken, Ihre wichtigen HSM-Passwörter darauf zu notieren und die Kopie anschließend an einem sicheren Ort abzugeben. Wir empfehlen, mindestens eine Kopie dieses Arbeitsblatts an einem sicheren Ort außerhalb des Standorts aufzubewahren.

5. (Optional) Verwenden Sie den Befehl `user list`, um sicherzustellen, dass sich der Typ des Benutzers in [Admin/CO](#) geändert hat.

```

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. Verwenden Sie den quit-Befehl, um das CloudHSM-CLI-Tool zu stoppen.

```
aws-cloudhsm > quit
```

Weitere Informationen zur Arbeit mit der CloudHSM-CLI oder der CMU finden Sie unter [Grundlegendes zu HSM-Benutzern](#) und [Grundlegendes zur HSM-Benutzerverwaltung mit CMU](#).

Richten Sie gegenseitiges TLS zwischen Client und AWS CloudHSM (empfohlen) ein

In den folgenden Themen werden die Schritte beschrieben, die Sie ausführen müssen, um die gegenseitige Transport Layer Security (mTLS) zwischen Client und zu aktivieren. AWS CloudHSM

Überlegungen

- Derzeit ist diese Funktion ausschließlich auf hsm2m.medium verfügbar. Weitere Informationen zum HSM-Typ finden Sie unter. [AWS CloudHSM Cluster-Modi](#)

- mTLS wird für AWS CloudHSM Schlüsselspeicher, die mit verwendet werden, nicht unterstützt.
AWS Key Management Service

Themen

- [Schritt 1. Erstellen und registrieren Sie einen Vertrauensanker auf dem HSM](#)
- [Schritt 2. Aktivieren Sie mTLS für AWS CloudHSM](#)
- [Schritt 3. Stellen Sie die mTLS-Durchsetzung ein für AWS CloudHSM](#)

Schritt 1. Erstellen und registrieren Sie einen Vertrauensanker auf dem HSM

Vor der Aktivierung von mTLS muss ein Vertrauensanker erstellt und auf dem HSM registriert werden. Dies ist ein zweistufiger Prozess:

Themen

- [Erstellen Sie einen privaten Schlüssel und ein selbstsigniertes Stammzertifikat](#)
- [Registrieren Sie den Vertrauensanker auf dem HSM](#)

Erstellen Sie einen privaten Schlüssel und ein selbstsigniertes Stammzertifikat

Note

Für einen Produktionscluster sollte der Schlüssel, den Sie erstellen wollen, auf sichere Weise mit einer vertrauenswürdigen Zufallsquelle erstellt werden. Wir empfehlen, dass Sie ein sicheres standortexternes und Offline-HSM oder etwas Äquivalentes verwenden. Speichern Sie den Schlüssel sicher.

Für die Entwicklung und das Testen können Sie jedes praktische Tool (wie OpenSSL) verwenden, um den Schlüssel zu erstellen und ein Stammzertifikat selbst zu signieren. Sie benötigen den Schlüssel und das Stammzertifikat, um das Client-Zertifikat in der Datei [enable mTLS](#) for zu signieren. AWS CloudHSM

Die folgenden Beispiele zeigen, wie Sie mit [OpenSSL](#) einen privaten Schlüssel und ein selbstsigniertes Stammzertifikat erstellen.

Example– Erstellen eines privaten Schlüssels mit OpenSSL

Verwenden Sie den folgenden Befehl, um einen 4096-Bit-RSA-Schlüssel zu erstellen, der mit dem AES-256-Algorithmus verschlüsselt ist. Um dieses Beispiel zu verwenden, `<mtls_ca_root_1.key>` ersetzen Sie es durch den Namen der Datei, in der Sie den Schlüssel speichern möchten.

```
$ 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— Erstellen Sie ein selbstsigniertes Root-Zertifikat mit OpenSSL

Verwenden Sie den folgenden Befehl, um ein selbstsigniertes Stammzertifikat zu erstellen, das nach `mtls_ca_root_1.crt` dem privaten Schlüssel benannt ist, den Sie gerade erstellt haben. Das Zertifikat ist 25 Jahre (9130 Tage) gültig. Lesen Sie die Anweisungen auf dem Bildschirm und befolgen Sie die Eingabeaufforderungen.

```
$ 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 []:
```

Registrieren Sie den Vertrauensanker auf dem HSM

Nach der Erstellung eines selbstsignierten Stammzertifikats muss der Administrator es als Vertrauensanker im Cluster registrieren. AWS CloudHSM

Um einen Vertrauensanker beim HSM zu registrieren

1. Verwenden Sie den folgenden Befehl, um die CloudHSM-CLI zu starten:

Linux

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

Windows

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

2. Melden Sie sich mit der CloudHSM-CLI als Administrator an.

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

3. Verwenden Sie den [Registrieren Sie einen Vertrauensanker mit CloudHSM CLI](#) Befehl, um den Vertrauensanker zu registrieren. Weitere Informationen finden Sie im folgenden Beispiel. Alternativ können Sie auch den Befehl `help cluster mtls register-trust-anchor` ausführen.

Example— Registriert einen Vertrauensanker beim AWS CloudHSM Cluster

Das folgende Beispiel zeigt, wie der `cluster mtls register-trust-anchor` Befehl in der CloudHSM-CLI verwendet wird, um einen Vertrauensanker auf dem HSM zu registrieren. Der Admin muss bei dem HSM angemeldet sein, um diesen Befehl zu verwenden. Ersetzen Sie diese Werte durch Ihre eigenen Werte:

```
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 unterstützt die Registrierung von Zwischenzertifikaten als Vertrauensanker. In solchen Fällen muss die gesamte PEM-kodierte Zertifikatskettendatei im HSM registriert werden, wobei die Zertifikate in hierarchischer Reihenfolge angeordnet sind. AWS CloudHSM unterstützt eine Zertifikatskette von 6980 Byte.

Nachdem Sie den Vertrauensanker erfolgreich registriert haben, können Sie den `cluster mtls list-trust-anchors` Befehl ausführen, um die aktuell registrierten Vertrauensanker zu überprüfen, wie unten gezeigt:

```
aws-cloudhsm > cluster mtls list-trust-anchors
{
  "error_code": 0,
  "data": {
    "trust_anchors": [
      {
        "certificate-reference": "0x01",
        "certificate": "<PEM Encoded Certificate>",
        "cluster-coverage": "full"
      }
    ]
  }
}
```


Note

Die maximale Anzahl von Vertrauensankern, die auf hsm2m.medium registriert werden können, beträgt zwei (2).

Schritt 2. Aktivieren Sie mTLS für AWS CloudHSM

Um die mTLS für zu aktivieren AWS CloudHSM, müssen Sie einen privaten Schlüssel und ein Client-Zertifikat erstellen, das mit dem Stammzertifikat signiert ist, das wir unter [Erstellen und Registrieren eines Vertrauensankers auf dem HSM](#) generiert haben, und dann eines der Konfigurationstools des Client SDK 5 verwenden, um den privaten Schlüsselpfad und den Pfad der Client-Zertifikatskette einzurichten.

Themen

- [Erstellen Sie einen privaten Schlüssel und eine Client-Zertifikatskette](#)
- [Konfigurieren Sie mTLS für Client SDK 5](#)

Erstellen Sie einen privaten Schlüssel und eine Client-Zertifikatskette

Example– Erstellen eines privaten Schlüssels mit OpenSSL

Verwenden Sie den folgenden Befehl, um einen 4096-Bit-RSA-Schlüssel zu erstellen. Um dieses Beispiel zu verwenden, `<ssl-client.key>` ersetzen Sie es durch den Namen der Datei, in der Sie den Schlüssel speichern möchten.

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

Example— Generieren Sie eine Certificate Signing Request (CSR) mit OpenSSL

Verwenden Sie den folgenden Befehl, um eine Zertifikatssignieranforderung (CSR) aus dem privaten Schlüssel zu generieren, den Sie gerade erstellt haben. Lesen Sie die Anweisungen auf dem Bildschirm und befolgen Sie die Eingabeaufforderungen.

```
$ 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— Signieren Sie die CSR mit dem Stammzertifikat

Verwenden Sie den folgenden Befehl, um die CSR mit dem Stammzertifikat zu signieren, das wir unter [Erstellen und registrieren Sie einen Vertrauensanker auf dem HSM erstellt und registriert haben, und erstellen Sie ein Client-Zertifikat mit dem](#) Namen `ssl-client.crt`. Das Zertifikat ist 5 Jahre (1826 Tage) gültig.

```
$ 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— Erstellen Sie eine Client-Zertifikatskette

Verwenden Sie den folgenden Befehl, um das Client-Zertifikat und das Stammzertifikat, das wir unter [Erstellen und registrieren Sie einen Vertrauensanker auf dem HSM](#) erstellt und registriert haben, zu kombinieren und eine Client-Zertifikatskette mit dem Namen `ssl-client.pem`, die im nächsten Schritt zur Konfiguration verwendet wird.

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

Note

Wenn Sie Zwischenzertifikate unter [Erstellen registriert haben und einen Vertrauensanker auf dem HSM als Vertrauensanker registrieren](#), stellen Sie sicher, dass Sie das Client-Zertifikat mit der gesamten Zertifikatskette kombinieren, um eine Client-Zertifikatskette zu erstellen.

Konfigurieren Sie mTLS für Client SDK 5

Verwenden Sie eines der Konfigurationstools des Client SDK 5, um das gegenseitige TLS zu aktivieren, indem Sie den richtigen Client-Schlüsselpfad und den richtigen Pfad für die Client-Zertifikatskette angeben. Weitere Informationen zum Konfigurationstool für Client SDK 5 finden Sie unter [???](#).

PKCS #11 library

So verwenden Sie ein benutzerdefiniertes Zertifikat und einen Schlüssel für die gegenseitige Authentifizierung zwischen TLS Client und HSM mit Client SDK 5 unter Linux

1. Kopieren Sie Schlüssel und Zertifikat in das entsprechende Verzeichnis.

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

2. Verwenden Sie das Configure-Tool, um `ssl-client.pem` und `ssl-client.key` anzugeben.

```
$ 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>
```

So verwenden Sie ein benutzerdefiniertes Zertifikat und einen Schlüssel für die gegenseitige Authentifizierung zwischen TLS Client und HSM mit Client SDK 5 unter Windows

1. Kopieren Sie Schlüssel und Zertifikat in das entsprechende Verzeichnis.

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

2. Verwenden Sie mit einem PowerShell Interpreter das Konfigurationstool, um und anzugeben. `ssl-client.pem` `ssl-client.key`

```
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

Um ein benutzerdefiniertes Zertifikat und einen Schlüssel für die gegenseitige Authentifizierung zwischen TLS Client und HSM mit Client SDK 5 unter Linux zu verwenden

1. Kopieren Sie Schlüssel und Zertifikat in das entsprechende Verzeichnis.

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

2. Verwenden Sie das Configure-Tool, um `ssl-client.pem` und `ssl-client.key` anzugeben.

```
$ 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)

So verwenden Sie ein benutzerdefiniertes Zertifikat und einen Schlüssel für die gegenseitige Authentifizierung zwischen TLS Client und HSM mit Client SDK 5 unter Windows

1. Kopieren Sie Schlüssel und Zertifikat in das entsprechende Verzeichnis.

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

2. Verwenden Sie mit einem PowerShell Interpreter das Konfigurationstool, um und anzugeben. `ssl-client.pem` `ssl-client.key`

```
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

Um ein benutzerdefiniertes Zertifikat und einen Schlüssel für die gegenseitige Authentifizierung zwischen TLS Client und HSM mit Client SDK 5 unter Linux zu verwenden

1. Kopieren Sie Schlüssel und Zertifikat in das entsprechende Verzeichnis.

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

2. Verwenden Sie das Configure-Tool, um `ssl-client.pem` und `ssl-client.key` anzugeben.

```
$ 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>
```

So verwenden Sie ein benutzerdefiniertes Zertifikat und einen Schlüssel für die gegenseitige Authentifizierung zwischen TLS Client und HSM mit Client SDK 5 unter Windows

1. Kopieren Sie Schlüssel und Zertifikat in das entsprechende Verzeichnis.

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

2. Verwenden Sie mit einem PowerShell Interpreter das Konfigurationstool, um und anzugeben. `ssl-client.pem` `ssl-client.key`

```
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

Um ein benutzerdefiniertes Zertifikat und einen Schlüssel für die gegenseitige Authentifizierung zwischen TLS Client und HSM mit Client SDK 5 unter Linux zu verwenden

1. Kopieren Sie Schlüssel und Zertifikat in das entsprechende Verzeichnis.

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

2. Verwenden Sie das Configure-Tool, um `ssl-client.pem` und `ssl-client.key` anzugeben.

```
$ 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>
```

So verwenden Sie ein benutzerdefiniertes Zertifikat und einen Schlüssel für die gegenseitige Authentifizierung zwischen TLS Client und HSM mit Client SDK 5 unter Windows

1. Kopieren Sie Schlüssel und Zertifikat in das entsprechende Verzeichnis.

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

2. Verwenden Sie mit einem PowerShell Interpreter das Konfigurationstool, um und anzugeben. `ssl-client.pem` `ssl-client.key`

```
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>
```

Schritt 3. Stellen Sie die mTLS-Durchsetzung ein für AWS CloudHSM

Nach der Konfiguration mit einem der Konfigurationstools des Client SDK 5 AWS CloudHSM erfolgt die Verbindung zwischen Client und Client über gegenseitiges TLS im Cluster. Wenn Sie jedoch den privaten Schlüsselpfad und den Pfad der Client-Zertifikatskette aus der Konfigurationsdatei

entfernen, wird die Verbindung wieder in reguläres TLS umgewandelt. Sie können die CloudHSM-CLI verwenden, um die mTLS-Erzwingung im Cluster festzulegen, indem Sie die folgenden Schritte ausführen:

1. Verwenden Sie den folgenden Befehl, um die CloudHSM-CLI zu starten:

Linux

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

Windows

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

2. Melden Sie sich mit der CloudHSM-CLI als Administrator an.

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

Note

1. Stellen Sie sicher, dass Sie die CloudHSM-CLI konfiguriert haben, und starten Sie die CloudHSM-CLI unter einer mTLS-Verbindung.
2. Sie müssen als Standard-Admin-Benutzer mit dem Benutzernamen admin angemeldet sein, bevor Sie die mTLS-Durchsetzung einrichten können.

3. Verwenden Sie den [Legen Sie die mTLS-Durchsetzungsstufe mit der CloudHSM CLI fest](#) Befehl, um die Durchsetzung festzulegen. Weitere Informationen finden Sie im folgenden Beispiel. Alternativ können Sie auch den Befehl `help cluster mtlb set-enforcement` ausführen.

Example— Stellen Sie die mTLS-Durchsetzung mit dem AWS CloudHSM Cluster ein

Das folgende Beispiel zeigt, wie Sie den `cluster mtls set-enforcement` Befehl in der CloudHSM-CLI verwenden, um die mTLS-Erzwingung mit dem HSM festzulegen. Um diesen Befehl verwenden zu können, muss der Administrator mit dem Benutzernamen `admin` beim HSM angemeldet sein.

```
aws-cloudhsm > cluster mtls set-enforcement --level cluster
{
  "error_code": 0,
  "data": {
    "message": "Mtls enforcement level set to Cluster successfully"
  }
}
```

 Warning

Nachdem Sie die Verwendung von mTLS im Cluster erzwungen haben, werden alle bestehenden Verbindungen, die keine MTLS-Verbindungen sind, gelöscht und Sie können sich nur mit mTLS-Zertifikaten mit dem Cluster verbinden.

Schlüssel erstellen und verwenden in AWS CloudHSM

Bevor Sie Schlüssel in Ihrem neuen Cluster erstellen und verwenden können, erstellen Sie einen HSM-Benutzer (Hardware Security Module) mit der AWS CloudHSM CLI. Weitere Informationen finden Sie unter [Grundlegendes zu HSM-Benutzerverwaltungsaufgaben, Erste Schritte mit der AWS CloudHSM CloudHSM-Befehlszeilenschnittstelle \(CLI\)](#) und [So verwalten Sie HSM-Benutzer](#).

 Note

Wenn Sie Client-SDK 3 verwenden, verwenden Sie [CloudHSM Management Utility \(CMU\)](#) anstelle von CloudHSM-CLI.

Nachdem Sie HSM-Benutzer erstellt haben, können Sie sich mit einer der folgenden Optionen beim HSM anmelden und Schlüssel verwalten:

- Verwenden Sie das [Schlüsselverwaltungsprogramm, ein Befehlszeilentool](#)
- Erstellen Sie eine C-Anwendung mit der [PKCS #11-Bibliothek](#)
- Erstellen Sie eine Java-Anwendung mit dem [JCE-Anbieter](#)
- Verwenden Sie die [OpenSSL Dynamic Engine direkt von der Befehlszeile aus](#)
- Verwenden Sie die OpenSSL Dynamic Engine für TLS-Offload mit [NGINX- und Apache-Webservern](#)
- Verwenden Sie den Key Storage Provider (KSP) für AWS CloudHSM mit der [Microsoft Windows Server Certificate Authority \(CA\)](#)
- Verwenden Sie den Key Storage Provider (KSP) für AWS CloudHSM mit dem [Microsoft Sign Tool](#)
- Verwenden Sie den Key Storage Provider (KSP) für das TLS-Offload mit dem [Internet Information Server \(IIS\)](#) -Webserver

Bewährte Methoden für AWS CloudHSM

Führen Sie die bewährten Methoden in diesem Thema durch, um sie effektiv zu nutzen AWS CloudHSM.

Inhalt

- [AWS CloudHSM Bewährte Methoden für die Clusterverwaltung](#)
- [AWS CloudHSM Bewährte Methoden für die Benutzerverwaltung](#)
- [AWS CloudHSM Bewährte Methoden für die Schlüsselmanagement](#)
- [AWS CloudHSM Bewährte Methoden für die Anwendungsintegration](#)
- [AWS CloudHSM Überwachung bewährter Verfahren](#)

AWS CloudHSM Bewährte Methoden für die Clusterverwaltung

Folgen Sie den bewährten Methoden in diesem Abschnitt, wenn Sie Ihren AWS CloudHSM Cluster erstellen, darauf zugreifen und ihn verwalten.

Skalieren Sie Ihren Cluster, um den Spitzenverkehr zu bewältigen

Verschiedene Faktoren können den maximalen Durchsatz beeinflussen, den Ihr Cluster verarbeiten kann, darunter die Größe der Client-Instanz, die Clustergröße, die Netzwerktopographie und die kryptografischen Operationen, die Sie für Ihren Anwendungsfall benötigen.

Als Ausgangspunkt finden Sie im Thema [AWS CloudHSM Informationen zur Leistung](#) Leistungsschätzungen zu gängigen Clustergrößen und -konfigurationen. Wir empfehlen Ihnen, Ihren Cluster mit der zu erwartenden Spitzenlast zu testen, um festzustellen, ob Ihre aktuelle Architektur robust ist und die richtige Skalierung aufweist.

Richten Sie Ihren Cluster auf Hochverfügbarkeit aus

Fügen Sie Redundanz hinzu, um die Wartung zu berücksichtigen: AWS kann Ihr HSM bei geplanten Wartungsarbeiten oder wenn ein Problem erkannt wird, ersetzen. In der Regel sollte Ihre Clustergröße mindestens +1 Redundanz aufweisen. Wenn Sie HSMs beispielsweise zwei benötigen, damit Ihr Service in Spitzenzeiten betrieben werden kann, ist Ihre ideale Clustergröße dann drei. Wenn Sie die Best Practices in Bezug auf die Verfügbarkeit befolgen, sollten diese HSM-Ersetzungen

Ihren Service nicht beeinträchtigen. Laufende Operationen am ausgetauschten HSM können jedoch fehlschlagen und müssen erneut versucht werden.

Verteilen Sie Ihr HSMs System auf viele Availability Zones: Überlegen Sie, wie Ihr Service während eines Ausfalls in der Availability Zone funktionieren kann. AWS empfiehlt, dass Sie Ihren Service HSMs auf so viele Availability Zones wie möglich verteilen. Bei einem Cluster mit drei sollten Sie HSMs sich HSMs auf drei Availability Zones verteilen. Abhängig von Ihrem System benötigen Sie möglicherweise zusätzliche Redundanz.

Halten Sie mindestens drei Schlüssel bereit HSMs , um die Haltbarkeit neu generierter Schlüssel zu gewährleisten

Für Anwendungen, die die Haltbarkeit neu generierter Schlüssel erfordern, empfehlen wir, mindestens drei Schlüssel auf verschiedene Availability Zones in einer Region zu HSMs verteilen.

Sicherer Zugriff auf Ihren Cluster

Verwenden Sie private Subnetze, um den Zugriff auf Ihre Instance zu beschränken: Starten Sie Ihre Instances HSMs und Client-Instances in den privaten Subnetzen Ihrer VPC. Dadurch wird der Zugriff HSMs von außen auf Ihre Daten eingeschränkt.

Verwenden Sie VPC-Endpunkte für den Zugriff APIs: Die AWS CloudHSM Datenebene wurde so konzipiert, dass sie funktioniert, ohne dass Zugriff auf das Internet oder AWS erforderlich ist. APIs Wenn Ihre Client-Instance Zugriff auf die AWS CloudHSM API benötigt, können Sie VPC-Endpunkte verwenden, um auf die API zuzugreifen, ohne dass Sie einen Internetzugang auf Ihrer Client-Instance benötigen. Weitere Informationen finden Sie unter [AWS CloudHSM und VPC-Endpunkte](#).

Senken Sie die Kosten, indem Sie sie an Ihre Bedürfnisse anpassen

Es fallen keine Vorabkosten für die Nutzung AWS CloudHSM an. Sie zahlen für jedes HSM, das Sie starten, eine Stundengebühr, bis Sie das HSM kündigen. Wenn Ihr Service keine kontinuierliche Nutzung von erfordert, können Sie die Kosten senken AWS CloudHSM, indem Sie Ihre Daten auf Null herunterskalieren (löschen) HSMs , wenn sie nicht benötigt werden. Wenn sie wieder benötigt HSMs werden, können Sie Ihre Daten HSMs aus einem Backup wiederherstellen. Wenn Sie beispielsweise aufgrund einer Arbeitslast Code einmal im Monat signieren müssen, und zwar am letzten Tag des Monats, können Sie Ihren Cluster vorher hochskalieren, ihn herunterskalieren, indem Sie Ihren Code HSMs nach Abschluss der Arbeit löschen, und dann Ihren Cluster wiederherstellen, um am Ende des nächsten Monats erneut Signierungsvorgänge durchzuführen.

AWS CloudHSM erstellt automatisch regelmäßige Backups der HSMs Dateien im Cluster. Wenn Sie zu einem späteren Zeitpunkt ein neues HSM hinzufügen, AWS CloudHSM wird das letzte Backup auf dem neuen HSM wiederhergestellt, sodass Sie es an derselben Stelle wieder verwenden können, an der Sie es verlassen haben. [Informationen zur Berechnung Ihrer AWS CloudHSM Architektorkosten finden Sie unter AWS CloudHSM Preise.](#)

Zugehörige Ressourcen:

- [Allgemeiner Überblick über Backups](#)
- [Richtlinie zur Aufbewahrung von Backup](#)
- [AWS CloudHSM Cluster-Backups zwischen AWS Regionen kopieren](#)

AWS CloudHSM Bewährte Methoden für die Benutzerverwaltung

Folgen Sie den bewährten Methoden in diesem Abschnitt, um Benutzer in Ihrem AWS CloudHSM Cluster effektiv zu verwalten. HSM-Benutzer unterscheiden sich von IAM-Benutzern. IAM-Benutzer und Entitäten, die über eine identitätsbasierte Richtlinie mit den entsprechenden Berechtigungen verfügen, können diese erstellen, HSMs indem sie über die AWS-API mit Ressourcen interagieren. Nachdem das HSM erstellt wurde, müssen Sie HSM-Benutzeranmeldedaten verwenden, um Vorgänge auf dem HSM zu authentifizieren. Eine ausführliche Anleitung für HSM-Benutzer finden Sie unter. [HSM-Benutzer in AWS CloudHSM](#)

Schützen Sie die Anmeldeinformationen Ihrer HSM-Benutzer

Es ist unerlässlich, die Anmeldeinformationen Ihrer HSM-Benutzer sicher zu schützen, da HSM-Benutzer die Entitäten sind, die auf Ihr HSM zugreifen und kryptografische und Verwaltungsvorgänge auf Ihrem HSM ausführen können. AWS CloudHSM hat keinen Zugriff auf Ihre HSM-Benutzeranmeldedaten und kann Ihnen nicht weiterhelfen, wenn Sie den Zugriff darauf verlieren.

Haben Sie mindestens zwei Administratoren, um eine Aussperrung zu verhindern

Um zu verhindern, dass Sie aus Ihrem Cluster ausgesperrt werden, empfehlen wir, dass Sie mindestens zwei Administratoren haben, falls ein Administratorkennwort verloren geht. In diesem Fall können Sie den anderen Administrator verwenden, um das Passwort zurückzusetzen.

 Note

Admins im Client SDK 5 sind synonym mit Crypto Officers (COs) im Client SDK 3.

Aktivieren Sie das Quorum für alle Benutzerverwaltungsvorgänge

Mit Quorum können Sie eine Mindestanzahl von Administratoren festlegen, die einen Benutzerverwaltungsvorgang genehmigen müssen, bevor dieser Vorgang durchgeführt werden kann. Aufgrund der Rechte, die Administratoren haben, empfehlen wir, Quorum für alle Benutzerverwaltungsvorgänge zu aktivieren. Dies kann die potenziellen Auswirkungen einschränken, wenn eines Ihrer Administrator Kennwörter kompromittiert wird. Weitere Informationen finden Sie unter [Quorum verwalten](#).

Erstellen Sie mehrere Krypto-Benutzer, von denen jeder über eingeschränkte Berechtigungen verfügt

Durch die Trennung der Verantwortlichkeiten der Krypto-Benutzer hat kein Benutzer die vollständige Kontrolle über das gesamte System. Aus diesem Grund empfehlen wir Ihnen, mehrere Krypto-Benutzer zu erstellen und deren Berechtigungen einzuschränken. In der Regel erfolgt dies, indem verschiedenen Krypto-Benutzern deutlich unterschiedliche Verantwortlichkeiten und Aktionen zugewiesen werden (z. B. ein Krypto-Benutzer, der für die Generierung und gemeinsame Nutzung von Schlüsseln mit anderen Krypto-Benutzern verantwortlich ist, die sie dann in Ihrer Anwendung verwenden).

Zugehörige Ressourcen:

- [Einen Schlüssel mit CloudHSM CLI teilen](#)
- [Freigabe eines Schlüssels mithilfe der CloudHSM-CLI rückgängig machen](#)

AWS CloudHSM Bewährte Methoden für die Schlüsselverwaltung

Folgen Sie den bewährten Methoden in diesem Abschnitt bei der Verwaltung von Schlüsseln in AWS CloudHSM.

Wählen Sie den richtigen Schlüsseltyp

Wenn Sie einen Sitzungsschlüssel verwenden, sind Ihre Transaktionen pro Sekunde (TPS) auf ein HSM beschränkt, für das der Schlüssel vorhanden ist. Zusätzliche HSMs Komponenten in Ihrem Cluster erhöhen den Durchsatz der Anfragen für diesen Schlüssel nicht. Wenn Sie einen Token-Schlüssel für dieselbe Anwendung verwenden, erfolgt ein Lastenausgleich für Ihre Anfragen auf alle HSMs in Ihrem Cluster verfügbaren Anfragen. Weitere Informationen finden Sie unter [Wichtige Synchronisations- und Haltbarkeitseinstellungen in AWS CloudHSM](#).

Verwalten Sie wichtige Speicherlimits

HSMs beschränken Sie die maximale Anzahl von Token- und Sitzungsschlüsseln, die gleichzeitig auf einem HSM gespeichert werden können. Informationen zu Speicherbeschränkungen für Schlüssel finden Sie unter [AWS CloudHSM Kontingente](#). Wenn Ihre Anwendung mehr als das Limit benötigt, können Sie eine oder mehrere der folgenden Strategien anwenden, um Schlüssel effektiv zu verwalten:

Verwenden Sie Trusted Wrapping, um Ihre Schlüssel in einem externen Datenspeicher zu speichern: Mit Trusted Key Wrapping können Sie das Speicherlimit für Schlüssel umgehen, indem Sie alle Ihre Schlüssel in einem externen Datenspeicher speichern. Wenn Sie diesen Schlüssel verwenden müssen, können Sie ihn als Sitzungsschlüssel in das HSM entpacken, den Schlüssel für den gewünschten Vorgang verwenden und dann den Sitzungsschlüssel verwerfen. Die ursprünglichen Schlüsseldaten bleiben sicher in Ihrem Datenspeicher gespeichert, sodass Sie sie jederzeit verwenden können. Wenn Sie zu diesem Zweck vertrauenswürdige Schlüssel verwenden, wird Ihr Schutz maximiert.

Schlüssel auf Cluster verteilen: Eine weitere Strategie zur Überwindung des Speicherlimits für Schlüssel besteht darin, Ihre Schlüssel in mehreren Clustern zu speichern. Bei diesem Ansatz behalten Sie eine Zuordnung der Schlüssel bei, die in jedem Cluster gespeichert sind. Verwenden Sie diese Zuordnung, um Ihre Client-Anfragen mit dem erforderlichen Schlüssel an den Cluster weiterzuleiten. Informationen dazu, wie Sie von derselben Client-Anwendung aus eine Verbindung zu mehreren Clustern herstellen können, finden Sie in den folgenden Themen:

- [Verbindung zu mehreren AWS CloudHSM Clustern mit dem JCE-Anbieter herstellen](#)
- [Konfiguration mit mehreren Steckplätzen mit PKCS #11 -Bibliothek für AWS CloudHSM](#)

Verwaltung und Sicherung der Schlüsselpackung

Schlüssel können über das Attribut entweder als extrahierbar oder nicht extrahierbar gekennzeichnet werden. EXTRACTABLE Standardmäßig sind HSM-Schlüssel als extrahierbar gekennzeichnet.

Extrahierbare Schlüssel sind Schlüssel, die durch Key-Wrapping aus dem HSM exportiert werden dürfen. Umhüllte Schlüssel sind verschlüsselt und müssen mit demselben Wrapping-Schlüssel entpackt werden, bevor sie verwendet werden können. Nicht extrahierbare Schlüssel dürfen unter keinen Umständen aus dem HSM exportiert werden. Es gibt keine Möglichkeit, einen nicht extrahierbaren Schlüssel extrahierbar zu machen. Aus diesem Grund ist es wichtig zu überlegen, ob Ihre Schlüssel extrahierbar sein müssen oder nicht, und das entsprechende identifizierende Attribut entsprechend festzulegen.

Wenn Sie in Ihrer Anwendung das Umschließen von Schlüsseln erfordern, sollten Sie Trusted Key Wrapping verwenden, um die Fähigkeit Ihrer HSM-Benutzer auf wrap/unwrap Schlüssel zu beschränken, die von einem Administrator ausdrücklich als vertrauenswürdig markiert wurden. Weitere Informationen finden Sie in den Themen zum Einschließen vertrauenswürdiger Schlüssel.

[Schlüssel rein AWS CloudHSM](#)

Zugehörige Ressourcen

- [Funktionen „Packen“ und „Entpacken“](#)
- [Verschlüsselungsfunktionen für JCE](#)
- [Unterstützte Java-Schlüsselattribute für AWS CloudHSM Client SDK 5](#)
- [Schlüsselattribute für CloudHSM-CLI](#)

AWS CloudHSM Bewährte Methoden für die Anwendungsintegration

Folgen Sie den bewährten Methoden in diesem Abschnitt, um die Integration Ihrer Anwendung in Ihren AWS CloudHSM Cluster zu optimieren.

Führen Sie ein Bootstrap für Ihr Client-SDK durch

Bevor Ihr Client-SDK eine Verbindung zu Ihrem Cluster herstellen kann, muss es gebootet werden. Beim Bootstrapping von IP-Adressen zu Ihrem Cluster empfehlen wir, den `--cluster-id`

Parameter nach Möglichkeit zu verwenden. Diese Methode füllt Ihre Konfiguration mit allen HSM-IP-Adressen in Ihrem Cluster, ohne dass Sie jede einzelne Adresse verfolgen müssen. Auf diese Weise wird Ihre Anwendungsinitialisierung für den Fall, dass ein HSM gewartet wird oder während eines Ausfalls der Availability Zone, zusätzliche Stabilität verleiht. Weitere Details finden Sie unter [Bootstrap für das Client-SDK](#).

Authentifizieren Sie sich, um Operationen auszuführen

In AWS CloudHSM müssen Sie sich bei Ihrem Cluster authentifizieren, bevor Sie die meisten Operationen, wie z. B. kryptografische Operationen, ausführen können.

Authentifizieren Sie sich mit der CloudHSM-CLI: [Sie können sich mit der CloudHSM-CLI entweder im Einzelbefehlsmodus oder im interaktiven Modus authentifizieren](#). Verwenden Sie den [Melden Sie sich mit der CloudHSM-CLI bei einem HSM an](#) Befehl, um sich im interaktiven Modus zu authentifizieren. Um sich im Einzelbefehlsmodus zu authentifizieren, müssen Sie die Umgebungsvariablen `CLOUDHSM_ROLE` und `CLOUDHSM_PIN` festlegen. Einzelheiten dazu finden Sie unter [Einzelbefehlsmodus](#). AWS CloudHSM empfiehlt, Ihre HSM-Anmeldeinformationen sicher zu speichern, wenn sie nicht von Ihrer Anwendung verwendet werden.

Authentifizieren Sie sich mit PKCS #11: In PKCS #11 melden Sie sich mit der `C_Login`-API an, nachdem Sie eine Sitzung mit `C_` geöffnet haben. `OpenSession` Sie müssen nur einen `C_Login` pro Slot (Cluster) durchführen. Nachdem Sie sich erfolgreich angemeldet haben, können Sie weitere Sitzungen mit `C_` öffnen, `OpenSession` ohne zusätzliche Anmeldevorgänge durchführen zu müssen. Beispiele für die Authentifizierung bei PKCS #11 finden Sie unter [Codebeispiele für die PKCS #11 - Bibliothek für das AWS CloudHSM Client-SDK 5](#)

Mit JCE authentifizieren: Der AWS CloudHSM JCE Provider unterstützt sowohl implizite als auch explizite Anmeldung. Welche Methode für Sie funktioniert, hängt von Ihrem Anwendungsfall ab. Wenn möglich, empfehlen wir die Verwendung von Implicit Login, da das SDK die Authentifizierung automatisch übernimmt, wenn Ihre Anwendung vom Cluster getrennt wird und erneut authentifiziert werden muss. Wenn Sie die implizite Anmeldung verwenden, können Sie auch Anmeldeinformationen für Ihre Anwendung angeben, wenn Sie eine Integration verwenden, bei der Sie keine Kontrolle über Ihren Anwendungscode haben. Weitere Informationen zu Anmeldemethoden finden Sie unter [Schritt 2: Geben Sie Anmeldeinformationen für den JCE-Anbieter ein](#).

Authentifizieren Sie sich mit OpenSSL: Mit der OpenSSL Dynamic Engine geben Sie Anmeldeinformationen über Umgebungsvariablen an. AWS CloudHSM empfiehlt, Ihre HSM-Anmeldeinformationen sicher zu speichern, wenn sie nicht von Ihrer Anwendung verwendet werden.

Wenn möglich, sollten Sie Ihre Umgebung so konfigurieren, dass diese Umgebungsvariablen systematisch ohne manuelle Eingabe abgerufen und festgelegt werden. Einzelheiten zur Authentifizierung mit OpenSSL finden Sie unter. [Installieren Sie die OpenSSL Dynamic Engine für AWS CloudHSM Client SDK 5](#)

Authentifizieren Sie sich mit KSP: Sie können sich beim Key Storage Provider (KSP) entweder mit dem Windows-Anmeldeinformationsmanager oder mit Umgebungsvariablen authentifizieren, siehe. [Installieren Sie den Key Storage Provider \(KSP\) für AWS CloudHSM Client SDK 5](#)

Effektives Verwalten von Schlüsseln in Ihrer Anwendung

Verwenden Sie Schlüsselattribute, um zu steuern, was Schlüssel bewirken können: Verwenden Sie beim Generieren eines Schlüssels Schlüsselattribute, um eine Reihe von Berechtigungen zu definieren, die bestimmte Arten von Vorgängen für diesen Schlüssel zulassen oder verweigern. Wir empfehlen, Schlüssel mit der geringsten Anzahl an Attributen zu generieren, die zur Ausführung ihrer Aufgabe erforderlich sind. Beispielsweise sollte ein AES-Schlüssel, der für die Verschlüsselung verwendet wird, nicht auch Schlüssel aus dem HSM herausnehmen dürfen. Weitere Informationen finden Sie auf unseren Seiten mit Attributen für den folgenden Client SDKs:

- [PKCS #11-Schlüsselattribute](#)
- [JCE-Schlüsselattribute](#)

Wenn möglich, sollten Sie Schlüsselobjekte zwischenspeichern, um die Latenz zu minimieren: Bei Schlüsselsuchoperationen wird jedes HSM in Ihrem Cluster abgefragt. Dieser Vorgang ist teuer und lässt sich nicht mit der HSM-Anzahl in Ihrem Cluster skalieren.

- Mit PKCS #11 finden Sie Schlüssel mithilfe der `C_FindObjects` API.
- Mit JCE finden Sie Schlüssel mithilfe der `KeyStore`

Um eine optimale Leistung zu erzielen, AWS empfiehlt es sich, Befehle zur Schlüsselsuche (wie [Suchen Sie mithilfe von KMU nach AWS CloudHSM Schlüsseln anhand von Attributen](#) und [Schlüssel für einen Benutzer mit CloudHSM CLI auflisten](#)) nur einmal beim Start der Anwendung zu verwenden und das zurückgegebene Schlüsselobjekt im Anwendungsspeicher zwischenspeichern. Wenn Sie dieses Schlüsselobjekt zu einem späteren Zeitpunkt benötigen, sollten Sie das Objekt aus Ihrem Cache abrufen, anstatt es bei jedem Vorgang abzufragen, was zu einem erheblichen Leistungsaufwand führt.

Verwenden Sie Multithreading

AWS CloudHSM unterstützt Multithread-Anwendungen, aber es gibt einige Dinge, die Sie bei Multithread-Anwendungen beachten sollten.

Mit PKCS #11 sollten Sie die PKCS #11 -Bibliothek (Aufruf) nur einmal initialisieren. `C_Initialize` Jedem Thread sollte eine eigene Sitzung () zugewiesen werden. `C_OpenSession` Es wird nicht empfohlen, dieselbe Sitzung in mehreren Threads zu verwenden.

Bei JCE sollte der AWS CloudHSM Anbieter nur einmal initialisiert werden. Teilen Sie Instanzen von SPI-Objekten nicht über mehrere Threads hinweg. Beispielsweise sollten Cipher, Signature, Digest, Mac KeyFactory oder KeyGenerator Objekte nur im Kontext ihres eigenen Threads verwendet werden.

Gehen Sie mit Drosselungsfehlern um

Unter den folgenden Umständen können HSM-Drosselungsfehler auftreten:

- Ihr Cluster ist nicht richtig skaliert, um Spitzenverkehr zu bewältigen.
- Ihr Cluster verfügt nicht über eine +1-Redundanz bei Wartungsereignissen.
- Ausfälle in der Availability Zone führen zu einer geringeren Anzahl verfügbarer Objekte HSMs in Ihrem Cluster.

Informationen darüber, wie Sie [HSM-Drosselung](#) dieses Szenario am besten handhaben können, finden Sie unter.

Um sicherzustellen, dass Ihr Cluster ausreichend dimensioniert ist und nicht gedrosselt wird, AWS empfiehlt es sich, einen Lasttest in Ihrer Umgebung mit dem zu erwartenden Spitzenverkehr durchzuführen.

Integrieren Sie Wiederholungsversuche bei Clustervorgängen

AWS kann Ihr HSM aus Betriebs- oder Wartungsgründen ersetzen. Um Ihre Anwendung gegen solche Situationen widerstandsfähig zu machen, AWS empfiehlt es sich, für alle Operationen, die an Ihren Cluster weitergeleitet werden, eine clientseitige Wiederholungslogik zu implementieren. Nachfolgende Wiederholungen bei fehlgeschlagenen Vorgängen aufgrund von Ersatzvorgängen werden voraussichtlich erfolgreich sein.

Implementieren Sie Strategien für die Notfallwiederherstellung

Als Reaktion auf ein Ereignis kann es erforderlich sein, Ihren Datenverkehr von einem ganzen Cluster oder einer Region weg zu verlagern. In den folgenden Abschnitten werden mehrere Strategien beschrieben, um dies zu erreichen.

Verwenden Sie VPC-Peering, um von einem anderen Konto oder einer anderen Region aus auf Ihren Cluster zuzugreifen: Sie können VPC-Peering verwenden, um von einem anderen Konto oder einer anderen Region aus auf Ihren AWS CloudHSM Cluster zuzugreifen. Informationen zur Einrichtung finden Sie unter [Was ist VPC-Peering?](#) im VPC Peering Guide. Sobald Sie Ihre Peering-Verbindungen eingerichtet und Ihre Sicherheitsgruppen entsprechend konfiguriert haben, können Sie wie gewohnt mit HSM-IP-Adressen kommunizieren.

Connect zu mehreren Clustern aus derselben Anwendung her: Der JCE-Anbieter, die PKCS #11-Bibliothek und die CloudHSM-CLI im Client SDK 5 unterstützen die Verbindung zu mehreren Clustern aus derselben Anwendung. Sie können beispielsweise über zwei aktive Cluster verfügen, die sich jeweils in unterschiedlichen Regionen befinden, und Ihre Anwendung kann eine Verbindung zu beiden gleichzeitig herstellen und im Rahmen des normalen Betriebs einen Lastenausgleich zwischen den beiden herstellen. Wenn Ihre Anwendung nicht das Client SDK 5 (das neueste SDK) verwendet, können Sie von derselben Anwendung aus keine Verbindung zu mehreren Clustern herstellen. Alternativ können Sie einen anderen Cluster am Laufen halten und, falls es zu einem regionalen Ausfall kommt, Ihren Datenverkehr auf den anderen Cluster verlagern, um Ausfallzeiten zu minimieren. Einzelheiten finden Sie auf den jeweiligen Seiten:

- [Konfiguration mit mehreren Steckplätzen mit PKCS #11 -Bibliothek für AWS CloudHSM](#)
- [Verbindung zu mehreren AWS CloudHSM Clustern mit dem JCE-Anbieter herstellen](#)
- [Verbindung zu mehreren Clustern mit CloudHSM CLI herstellen](#)

Einen Cluster aus einem Backup wiederherstellen: Sie können einen neuen Cluster aus einer Sicherung eines vorhandenen Clusters erstellen. Weitere Informationen finden Sie unter [Cluster-Backups in AWS CloudHSM](#).

Gestaffeln Sie Ihre Anwendungsbereitstellung

Folgen Sie gestaffelten Bereitstellungsstrategien wie Progressive-Wave-basierten Bereitstellungen, Einzelbereitstellungen und fortlaufenden Bereitstellungen für die Bereitstellung und Neustarts Ihrer Client-Anwendungen. Dieser Ansatz minimiert die potenziellen Auswirkungen von Änderungen und

stellt gleichzeitig sicher, dass genügend Kapazität vorhanden ist, um den Produktionsdatenverkehr während der Bereitstellung zu bedienen.

AWS CloudHSM Überwachung bewährter Verfahren

In diesem Abschnitt werden mehrere Mechanismen beschrieben, mit denen Sie Ihren Cluster und Ihre Anwendung überwachen können. Weitere Informationen zur Überwachung finden Sie unter [Überwachung AWS CloudHSM](#).

Überwachen Sie die Client-Protokolle

Jedes Client-SDK schreibt Protokolle, die Sie überwachen können. Informationen zur Client-Protokollierung finden Sie unter [Arbeiten mit AWS CloudHSM Client-SDK-Protokollen](#).

Auf Plattformen, die so konzipiert sind, dass sie kurzlebig sind, wie Amazon ECS und AWS Lambda, kann das Sammeln von Client-Protokollen aus einer Datei schwierig sein. In diesen Situationen empfiehlt es sich, die Client-SDK-Protokollierung so zu konfigurieren, dass Protokolle in die Konsole geschrieben werden. Die meisten Dienste sammeln diese Ausgabe automatisch und veröffentlichen sie in CloudWatch Amazon-Protokollen, damit Sie sie behalten und einsehen können.

Wenn Sie zusätzlich zum AWS CloudHSM Client-SDK eine Drittanbieter-Integration verwenden, sollten Sie sicherstellen, dass Sie das Softwarepaket so konfigurieren, dass auch die Ausgabe in der Konsole protokolliert wird. Die Ausgabe des AWS CloudHSM Client-SDK kann von diesem Paket erfasst und andernfalls in eine eigene Protokolldatei geschrieben werden.

Informationen [AWS CloudHSM Konfigurationstool für das Client-SDK 5](#) zur Konfiguration der Protokollierungsoptionen in Ihrer Anwendung finden Sie im.

Überwachen Sie die Überwachungsprotokolle

AWS CloudHSM veröffentlicht Auditprotokolle auf Ihrem CloudWatch Amazon-Konto. Die Auditprotokolle stammen aus dem HSM und verfolgen bestimmte Vorgänge zu Prüfungszwecken.

Sie können Auditprotokolle verwenden, um alle Verwaltungsbefehle zu verfolgen, die auf Ihrem HSM aufgerufen werden. Sie können beispielsweise einen Alarm auslösen, wenn Sie feststellen, dass ein unerwarteter Verwaltungsvorgang ausgeführt wird.

Weitere Details finden Sie unter [So funktioniert die HSM-Auditprotokollierung](#).

Überwachen AWS CloudTrail

AWS CloudHSM ist in einen Dienst integriert AWS CloudTrail, der eine Aufzeichnung der Aktionen bereitstellt, die von einem Benutzer, einer Rolle oder einem AWS Dienst in ausgeführt wurden AWS CloudHSM. AWS CloudTrail erfasst alle API-Aufrufe AWS CloudHSM als Ereignisse. Zu den erfassten Aufrufen gehören Aufrufe von der AWS CloudHSM Konsole und Codeaufrufen für die AWS CloudHSM API-Operationen.

Sie können AWS CloudTrail damit jeden API-Aufruf überprüfen, der an die AWS CloudHSM Steuerungsebene gesendet wird, um sicherzustellen, dass in Ihrem Konto keine unerwünschten Aktivitäten stattfinden.

Details dazu finden Sie unter [Arbeiten mit AWS CloudTrail und AWS CloudHSM](#).

Überwachen Sie CloudWatch Amazon-Metriken

Sie können CloudWatch Amazon-Metriken verwenden, um Ihren AWS CloudHSM Cluster in Echtzeit zu überwachen. Die Metriken können nach Region, Cluster-ID oder HSM-ID und Cluster-ID gruppiert werden.

Mithilfe von CloudWatch Amazon-Metriken können Sie CloudWatch Amazon-Alarme so konfigurieren, dass Sie vor potenziellen Problemen gewarnt werden, die sich auf Ihren Service auswirken könnten. Wir empfehlen, Alarme zu konfigurieren, um Folgendes zu überwachen:

- Nähern Sie sich Ihrem Tasten-Limit auf einem HSM
- Wir nähern uns dem Limit für die Anzahl der HSM-Sitzungen auf einem HSM
- Annäherung an das Limit für die HSM-Benutzeranzahl auf einem HSM
- Unterschiede in der HSM-Benutzer- oder Schlüsselanzahl zur Identifizierung von Synchronisationsproblemen
- Es ist nicht gesund HSMs , Ihren Cluster hochzuskalieren, bis AWS CloudHSM das Problem behoben ist

Weitere Details finden Sie unter [Arbeiten mit Amazon CloudWatch Logs und AWS CloudHSM Audit Logs](#).

Cluster in AWS CloudHSM

Ein Cluster ist eine Sammlung einzelner Hardware-Sicherheitsmodule (HSM), die synchron AWS CloudHSM bleiben. Wenn Sie eine Aufgabe oder einen Vorgang auf einem HSM in einem Cluster ausführen, werden die anderen HSMs in diesem Cluster automatisch auf dem neuesten Stand gehalten.

Sie können Ihre AWS CloudHSM Cluster über die [AWS CloudHSM Konsole](#) oder eines der [AWS SDKs Befehlszeilentools](#) verwalten. Weitere Informationen finden Sie unter den folgenden Themen.

Informationen zum Erstellen eines Clusters finden Sie unter [Erste Schritte](#).

Die folgenden Themen enthalten weitere Informationen zu Clustern.

Themen

- [AWS CloudHSM Cluster-Architektur](#)
- [AWS CloudHSM Cluster-Synchronisierung](#)
- [AWS CloudHSM Hochverfügbarkeit und Lastenausgleich im Cluster](#)
- [AWS CloudHSM Cluster-Modi](#)
- [HSM gibt ein AWS CloudHSM](#)
- [Connect das Client-SDK mit dem AWS CloudHSM Cluster](#)
- [Skalierung HSMs in einem AWS CloudHSM Cluster](#)
- [Löschen eines AWS CloudHSM Clusters](#)
- [AWS CloudHSM Cluster aus Backups erstellen](#)
- [Migration vom Typ Cluster-HSM](#)

AWS CloudHSM Cluster-Architektur

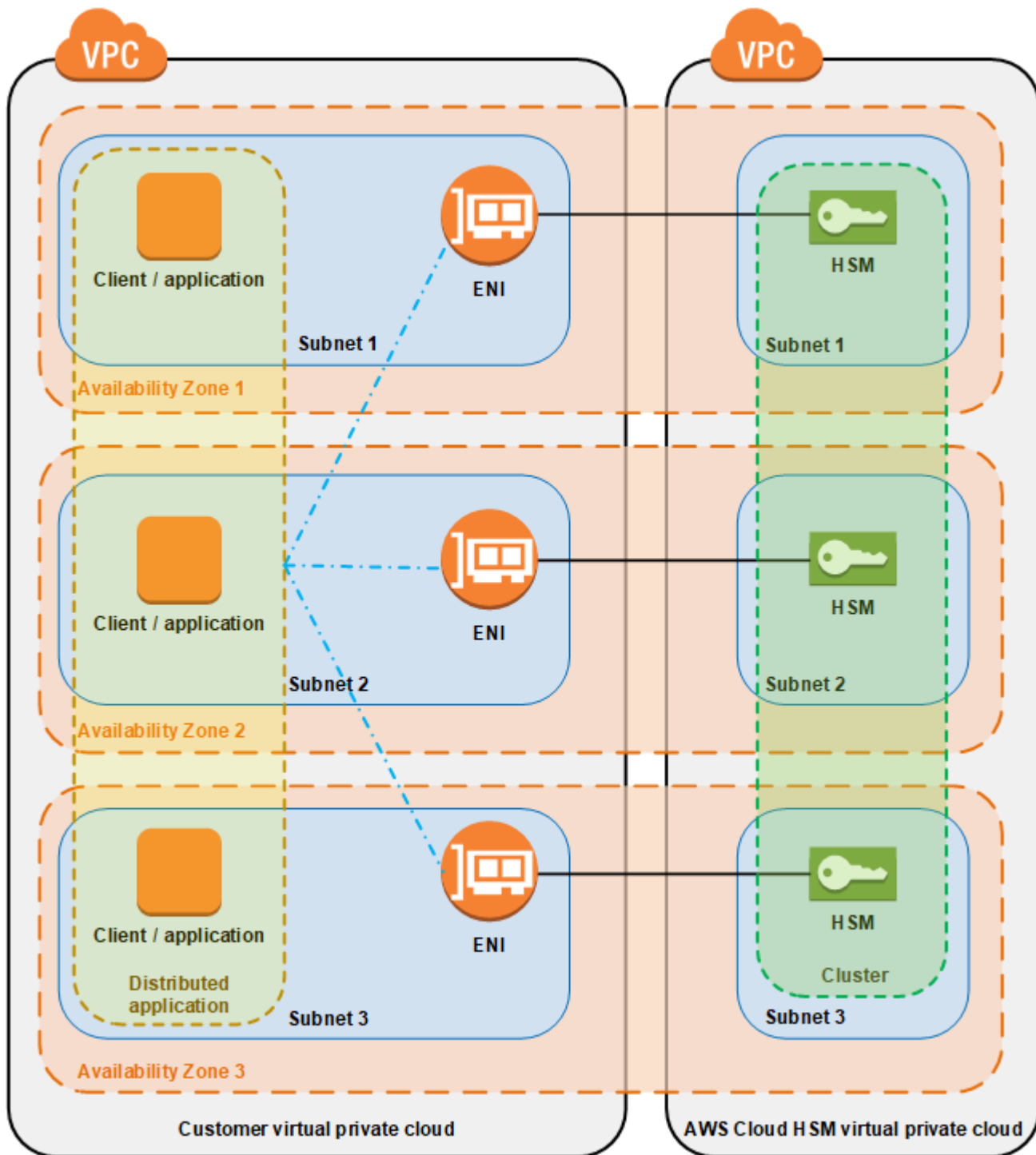
Wenn Sie einen Cluster erstellen, geben Sie eine Amazon Virtual Private Cloud (VPC) in Ihrem AWS Konto und ein oder mehrere Subnetze in dieser VPC an. Wir empfehlen, dass Sie in jeder Availability Zone (AZ) in der von Ihnen ausgewählten Region ein Subnetz erstellen. AWS Sie können private Subnetze erstellen, wenn Sie eine VPC erstellen. Weitere Informationen hierzu finden Sie unter [Erstellen Sie eine virtuelle private Cloud \(VPC\) für AWS CloudHSM](#).

Jedes Mal, wenn Sie ein HSM-Cluster erstellen, geben Sie den Cluster und die Availability Zone für das HSM. Indem Sie sie HSMs in verschiedene Availability Zones einteilen, erreichen Sie Redundanz und hohe Verfügbarkeit für den Fall, dass eine Availability Zone nicht verfügbar ist.

Wenn Sie ein HSM erstellen, AWS CloudHSM fügt es ein elastic network interface (ENI) in das angegebene Subnetz in Ihrem AWS Konto ein. Die Elastic Network-Schnittstelle ist die Schnittstelle für die Interaktion mit dem HSM. Das HSM befindet sich in einer separaten VPC in einem AWS Konto, das Eigentum von ist. AWS CloudHSM Das HSM und die entsprechende Netzwerkschnittstelle befinden sich in derselben Availability Zone.

Um mit dem HSMs in einem Cluster zu interagieren, benötigen Sie die AWS CloudHSM Client-Software. In der Regel installieren Sie den Client auf Amazon EC2 EC2-Instances, sogenannten Client-Instances, die sich in derselben VPC wie das HSM befinden ENIs, wie in der folgenden Abbildung dargestellt. Dies ist jedoch technisch nicht erforderlich. Sie können den Client auf jedem kompatiblen Computer installieren, sofern er eine Verbindung zum HSM herstellen kann. ENIs Der Client kommuniziert mit der Person HSMs in Ihrem Cluster über ihre. ENIs

Die folgende Abbildung stellt einen AWS CloudHSM Cluster mit drei Clustern dar HSMs, die sich jeweils in einer anderen Availability Zone in der VPC befinden.



AWS CloudHSM Cluster-Synchronisierung

AWS CloudHSM hält in einem AWS CloudHSM Cluster die Schlüssel der einzelnen Person HSMs synchron. Sie müssen nichts tun, um die Schlüssel auf Ihrem zu synchronisieren HSMs. Um die Benutzer und Richtlinien auf jedem HSM synchron zu halten, aktualisieren Sie die AWS CloudHSM

Client-Konfigurationsdatei, bevor Sie [HSM-Benutzer verwalten](#). Weitere Informationen finden Sie unter [Aufrechterhalten der Synchronität von HSM-Benutzern](#).

Wenn Sie einem Cluster ein neues HSM hinzufügen, AWS CloudHSM erstellt eine Sicherungskopie aller Schlüssel, Benutzer und Richtlinien auf einem vorhandenen HSM. Anschließend stellt es diese Sicherung auf dem neuen HSM wieder her. Dadurch bleiben die beiden synchron HSMs .

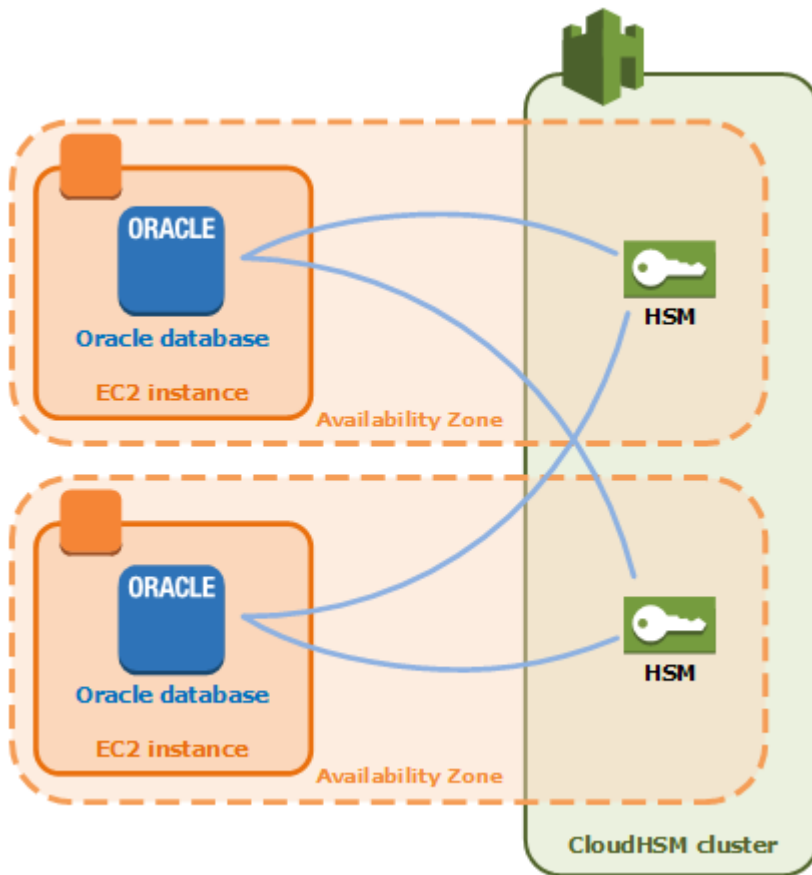
Wenn die HSMs in einem Cluster nicht mehr synchron sind, AWS CloudHSM werden sie automatisch neu synchronisiert. Um dies zu aktivieren, werden die Anmeldeinformationen des [Appliance-Benutzers AWS CloudHSM](#) verwendet. Dieser Benutzer existiert auf allen von HSMs bereitgestellten Geräten AWS CloudHSM und verfügt über eingeschränkte Berechtigungen. Er kann einen Hash von Objekten auf dem HSM erhalten und maskierte (verschlüsselte) Objekte extrahieren und einfügen. AWS kann Ihre Benutzer oder Schlüssel weder anzeigen oder ändern, noch mit diesen Schlüsseln kryptografischen Operationen durchführen.

AWS CloudHSM Hochverfügbarkeit und Lastenausgleich im Cluster

Wenn Sie einen AWS CloudHSM Cluster mit mehr als einem HSM erstellen, erhalten Sie automatisch einen Lastenausgleich. Lastenausgleich bedeutet, dass der [AWS CloudHSM Client](#) kryptografische Operationen auf der Grundlage der Kapazität HSMs der einzelnen HSM für zusätzliche Verarbeitung auf den gesamten Cluster verteilt.

Wenn Sie die HSMs in verschiedenen AWS Availability Zones erstellen, erhalten Sie automatisch Hochverfügbarkeit. Hohe Verfügbarkeit bedeutet, dass Sie die Zuverlässigkeit verbessern, da keines der HSMs für sich einen Single Point of Failure darstellt. Wir empfehlen, dass Sie HSMs in jedem Cluster mindestens zwei haben, wobei sich jedes HSM in verschiedenen Availability Zones innerhalb einer AWS Region befindet.

Die folgende Abbildung zeigt z. B. eine Oracle-Datenbankanwendung, die auf zwei verschiedene Availability Zones verteilt ist. Die Datenbank-Instances speichern ihre Masterschlüssel in einem Cluster, der in jeder Availability Zone ein HSM enthält. AWS CloudHSM synchronisiert die Schlüssel automatisch mit beiden, HSMs sodass sie sofort zugänglich und redundant sind.



AWS CloudHSM Cluster-Modi

AWS CloudHSM bietet Cluster in zwei Modi: FIPS und Nicht-FIPS. Im FIPS-Modus können nur nach dem Federal Information Processing Standard (FIPS) validierte Schlüssel und Algorithmen verwendet werden. Der Nicht-FIPS-Modus bietet alle Schlüssel und Algorithmen, die von unterstützt werden AWS CloudHSM, unabhängig von der FIPS-Zulassung.

Lesen Sie die Details auf dieser Seite, bevor Sie entscheiden, welcher Clustermodus und welcher HSM-Typ für Ihre Anforderungen geeignet sind.

Note

Alle Cluster, die vor dem 10. Juni 2024 erstellt wurden, befinden sich im FIPS-Modus und haben den HSM-Typ `hsm1.medium`.

Verwenden Sie den Befehl `describe-clusters`, um den Modus und den HSM-Typ Ihres Clusters zu sehen.

In der folgenden Tabelle sind die Hauptunterschiede zwischen den einzelnen Clustermodi aufgeführt:

Unterscheidungsmerkmal	FIPS-Modus	Nicht-FIPS-Modus
Kompatibilität mit dem HSM-Typ	Verfügbar mit hsm1.medium und hsm2m.medium.	Verfügbar mit hsm2m.medium.
Backup-Kompatibilität	Kann nur zum Sichern von Wiederherstellungsclustern im FIPS-Modus verwendet werden.	Kann nur zum Sichern und Wiederherstellen von Clustern im Nicht-FIPS-Modus verwendet werden.
Auswahl der Tasten	Unterstützt das Generieren und Verwenden von Schlüsseln mit Mechanismen, die FIPS-zugelassen ¹ sind.	Unterstützt das Generieren und Verwenden von Schlüsseln mit allen FIPS-validierten Mechanismen sowie mit anderen nicht validierten Mechanismen.
Algorithmen	Unterstützt AWS CloudHSM Algorithmen, die FIPS-zugelassen ¹ sind.	Unterstützt AWS CloudHSM Algorithmen, die sowohl FIPS-zugelassen als auch nicht FIPS-zugelassen sind.

[1] Weitere Informationen finden Sie unter [Benachrichtigungen über veraltete Versionen](#).

Beachten Sie vor der Auswahl eines Clustermodus, dass der Modus eines Clusters (FIPS oder Nicht-FIPS) nach der Erstellung nicht mehr geändert werden kann. Stellen Sie daher sicher, dass Sie den richtigen Modus für Ihre Bedürfnisse auswählen.

HSM gibt ein AWS CloudHSM

AWS CloudHSM bietet auch zwei Typen von Hardware-Sicherheitsmodulen (HSM) an: hsm1.medium und hsm2m.medium. Lesen Sie die Details auf dieser Seite, bevor Sie entscheiden, welcher HSM-Typ für Ihre Anforderungen geeignet ist.

AWS CloudHSM Bietet zusätzlich zu den Cluster-Modi zwei HSM-Typen: hsm1.medium und hsm2m.medium. Jeder HSM-Typ verwendet unterschiedliche Hardware, und jeder Cluster kann nur

einen HSM-Typ enthalten. In der folgenden Tabelle sind die wichtigsten Unterschiede zwischen den beiden aufgeführt:

Unterscheidungsmerkmal	hsm1.medium	hsm2m. mittel
Kompatibilität mit dem Clustermodus	Verfügbar für Cluster im FIPS-Modus.	Verfügbar für Cluster im FIPS- oder Nicht-FIPS-Modus.
Kompatibilität mit Netzwerktypen	Nicht verfügbar	Verfügbar für Cluster im FIPS- oder Nicht-FIPS-Modus.
Backup-Kompatibilität	Kann zur Sicherung und Wiederherstellung auf den Clustern hsm1.medium und hsm2m.medium im FIPS-Modus verwendet werden.	Kann nur zum Sichern und Wiederherstellen von hsm2m.medium-Clustern verwendet werden.
Wichtige Kapazität	3.300 pro Cluster.	Insgesamt 16.666 Schlüssel , wobei asymmetrische Schlüssel maximal 3.333 pro Cluster haben.
Client SDKs	Unterstützt alle Clients. SDKs	Unterstützt alle Clients SDKs.
Client-SDK-Versionen	Kompatibel mit SDK-Version 3.1.0 und höher.	Kompatibel mit der Client-SDK-Version 5.9.0 und höher.
Verfügbarkeit in der Region	CloudHSM unterstützt die Erstellung neuer Cluster in keiner AWS Region mehr. Weitere Informationen finden Sie unter Benachrichtigungen über veraltete Versionen .	Verfügbar in AWS-Regionen dem CloudHSM verfügbar ist .
Leistung	Informationen zur Leistung der einzelnen HSM-Typen finden Sie unter. AWS CloudHSM Informationen zur Leistung	

Unterscheidungsmerkmal	hsm1.medium	hsm2m. mittel
Zertifizierung	FIPS 140-2-, PCI DSS-, PCI-PIN- und SOC2 PCI-3DS-konform.	FIPS 140-3-, PCI DSS-, PCI-PIN- und PCI-3DS-konform. SOC2

Connect das Client-SDK mit dem AWS CloudHSM Cluster

Um mit Client-SDK 5 oder Client-SDK 3 eine Verbindung zum Cluster herzustellen, müssen Sie zunächst zwei Dinge tun:

- Verfügen Sie über ein ausstellendes Zertifikat auf der EC2-Instance
- Das Client-SDK per Bootstrap auf den Cluster übertragen

Platzieren Sie das ausstellende Zertifikat auf jeder EC2-Instance

Sie erstellen das ausstellende Zertifikat, wenn Sie den Cluster initialisieren. Kopieren Sie das ausstellende Zertifikat an den Standardspeicherort für die Plattform auf jeder EC2-Instance, die eine Verbindung zum Cluster herstellt.

Linux

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

Windows

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

Geben Sie den Speicherort des ausstellenden Zertifikats an

Beim Client-SDK 5 verwenden Sie das Configure-Tool, um den Speicherort des ausstellenden Zertifikats anzugeben.

PKCS #11 library

Um das ausstellende Zertifikat für das Client-SDK 5 auf Linux zu platzieren

- Verwenden Sie das Configure-Tool, um einen Speicherort für das ausstellende Zertifikat anzugeben.

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

Um das ausstellende Zertifikat unter Windows für Client-SDK 5 zu platzieren

- Verwenden Sie das Configure-Tool, um einen Speicherort für das ausstellende Zertifikat anzugeben.

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

OpenSSL Dynamic Engine

Um das ausstellende Zertifikat für das Client-SDK 5 auf Linux zu platzieren

- Verwenden Sie das Configure-Tool, um einen Speicherort für das ausstellende Zertifikat anzugeben.

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

OpenSSL Dynamic Engine Provider

Um das ausstellende Zertifikat für das Client-SDK 5 auf Linux zu platzieren

- Verwenden Sie das Configure-Tool, um einen Speicherort für das ausstellende Zertifikat anzugeben.

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

Key Storage Provider (KSP)

Um das ausstellende Zertifikat unter Windows für Client-SDK 5 zu platzieren

- Verwenden Sie das Configure-Tool, um einen Speicherort für das ausstellende Zertifikat anzugeben.

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

JCE provider

Um das ausstellende Zertifikat für das Client-SDK 5 auf Linux zu platzieren

- Verwenden Sie das Configure-Tool, um einen Speicherort für das ausstellende Zertifikat anzugeben.

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

Um das ausstellende Zertifikat unter Windows für Client-SDK 5 zu platzieren

- Verwenden Sie das Configure-Tool, um einen Speicherort für das ausstellende Zertifikat anzugeben.

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

CloudHSM CLI

Um das ausstellende Zertifikat für das Client-SDK 5 auf Linux zu platzieren

- Verwenden Sie das Configure-Tool, um einen Speicherort für das ausstellende Zertifikat anzugeben.

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

Um das ausstellende Zertifikat unter Windows für Client-SDK 5 zu platzieren

- Verwenden Sie das Configure-Tool, um einen Speicherort für das ausstellende Zertifikat anzugeben.

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

Weitere Informationen finden Sie unter [Configure Tool](#).

Weitere Informationen zur Initialisierung des Clusters oder zum Erstellen und Signieren des Zertifikats finden Sie unter [Cluster initialisieren](#).

Bootstrap für das Client-SDK

Der Bootstrap-Vorgang ist je nach der Version des Client-SDK, die Sie verwenden, unterschiedlich. Sie benötigen jedoch die IP-Adresse eines der Hardware-Sicherheitsmodule (HSM) im Cluster. Sie können die IP-Adresse eines beliebigen HSM verwenden, das an Ihren Cluster angeschlossen ist. Nachdem das Client-SDK eine Verbindung hergestellt hat, ruft es die IP-Adressen aller weiteren Benutzer ab HSMs und führt Lastenausgleich und clientseitige Schlüsselsynchronisierung durch.

Um eine IP-Adresse für den Cluster zu erhalten

Um eine IP-Adresse für ein HSM (Konsole) zu erhalten

1. Öffnen Sie die AWS CloudHSM Konsole zu <https://console.aws.amazon.com/cloudhsm/Hause>.
2. Um die AWS-Region zu ändern, verwenden Sie die Regionsauswahl in der oberen rechten Ecke der Seite.
3. Um die Cluster-Detailseite zu öffnen, wählen Sie in der Cluster-Tabelle die Cluster-ID aus.
4. Um die IP-Adresse zu erhalten, gehen Sie zur HSMs Registerkarte. Wählen Sie für IPv4 Cluster eine Adresse aus, die unter IPv4 ENI-Adresse aufgeführt ist. Verwenden Sie für Dual-Stack-Cluster entweder die ENI IPv4 - oder die IPv6 ENI-Adresse.

Um eine IP-Adresse für ein HSM zu erhalten (AWS CLI)

- Rufen Sie die IP-Adresse eines HSM ab, indem Sie den [describe-clusters](#) Befehl von der AWS CLI verwenden. In der Ausgabe des Befehls entspricht die IP-Adresse von den Werten von `EniIp` und `EniIPv6` (falls es HSMs sich um einen Dual-Stack-Cluster handelt).


```
$ 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",
...
      }
    ]
  }
}
```

Weitere Informationen zum Bootstrapping finden Sie unter [Configure Tool](#).

So bootstrappen Sie das Client-SDK 5

PKCS #11 library

So bootstrappen Sie eine Linux-EC2-Instance für Client-SDK 5

- Verwenden Sie das Konfigurationstool, um die IP-Adresse eines HSM in Ihrem Cluster anzugeben.

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

So bootstrappen Sie eine Windows-EC2-Instance für Client-SDK 5

- Verwenden Sie das Konfigurationstool, um die IP-Adresse eines HSM in Ihrem Cluster anzugeben.

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

OpenSSL Dynamic Engine

So bootstrappen Sie eine Linux-EC2-Instance für Client-SDK 5

- Verwenden Sie das Konfigurationstool, um die IP-Adresse eines HSM in Ihrem Cluster anzugeben.

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

OpenSSL Dynamic Engine Provider

So bootstrappen Sie eine Linux-EC2-Instance für Client-SDK 5

- Verwenden Sie das Konfigurationstool, um die IP-Adresse eines HSM in Ihrem Cluster anzugeben.

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

Key Storage Provider (KSP)

So bootstrappen Sie eine Windows-EC2-Instance für Client-SDK 5

- Verwenden Sie das Konfigurationstool, um die IP-Adresse eines HSM in Ihrem Cluster anzugeben.

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

JCE provider

So bootstrappen Sie eine Linux-EC2-Instance für Client-SDK 5

- Verwenden Sie das Konfigurationstool, um die IP-Adresse eines HSM in Ihrem Cluster anzugeben.

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

So bootstrappen Sie eine Windows-EC2-Instance für Client-SDK 5

- Verwenden Sie das Konfigurationstool, um die IP-Adresse eines HSM in Ihrem Cluster anzugeben.

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

CloudHSM CLI

So bootstrappen Sie eine Linux-EC2-Instance für Client-SDK 5

- Verwenden Sie das Configure-Tool, um die IP-Adresse der HSM(s) in Ihrem Cluster anzugeben.

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

So bootstrappen Sie eine Windows-EC2-Instance für Client-SDK 5

- Verwenden Sie das Configure-Tool, um die IP-Adresse der HSM(s) in Ihrem Cluster anzugeben.

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

Note

Sie können den `--cluster-id`-Parameter anstelle von `-a <HSM_IP_ADDRESSES>` verwenden. Informationen zu den Anforderungen für die Verwendung von `--cluster-id` finden Sie unter [AWS CloudHSM Konfigurationstool für das Client-SDK 5](#).

So bootstrappen Sie das Client-SDK 3

So bootstrappen Sie eine Linux-EC2-Instance für Client-SDK 3

- Verwenden Sie `configure` diese Option, um die IP-Adresse eines HSM in Ihrem Cluster anzugeben.

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

So bootstrappen Sie eine Windows-EC2-Instance für Client-SDK 3

- Wird verwendet `configure`, um die IP-Adresse eines HSM in Ihrem Cluster anzugeben.

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

Weitere Informationen zur `-`-Konfiguration finden Sie unter [???](#).

Skalierung HSMs in einem AWS CloudHSM Cluster

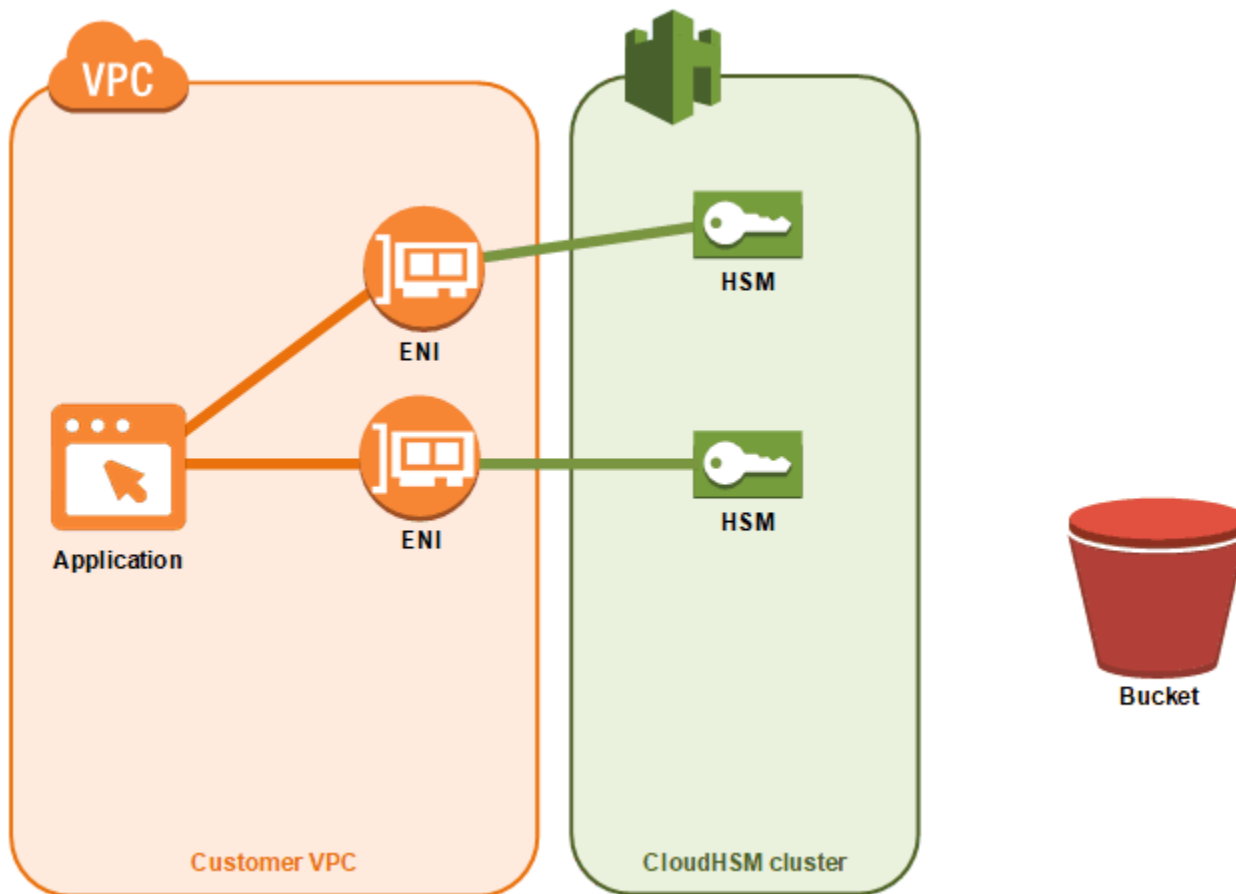
Um Ihren AWS CloudHSM Cluster nach oben oder unten zu skalieren, können Sie ihn HSMs mithilfe der [AWS CloudHSM Konsole](#) oder eines der [Befehlszeilentools hinzufügen AWS SDKs oder](#) entfernen. Wir empfehlen, Ihren Cluster einem Belastungstest zu unterziehen, um die zu erwartende Spitzenlast zu ermitteln, und dann ein weiteres HSM hinzuzufügen, um eine hohe Verfügbarkeit zu gewährleisten.

Themen

- [Hinzufügen eines HSM zu einem Cluster AWS CloudHSM](#)
- [Ein HSM aus einem Cluster entfernen AWS CloudHSM](#)

Hinzufügen eines HSM zu einem Cluster AWS CloudHSM

Die folgende Abbildung zeigt die Ereignisse, die beim Hinzufügen eines HSM in einem Cluster auftreten.



1. Sie fügen einem Cluster ein neues HSM hinzu. In den folgenden Verfahren wird erläutert, wie Sie diesen Vorgang über die [AWS CloudHSM -Konsole](#), die [AWS Command Line Interface \(AWS CLI\)](#) und die [AWS CloudHSM -API](#) ausführen.

Dies ist die einzige Aktion, die Sie ausführen. Die restlichen Ereignisse werden automatisch durchgeführt.

2. AWS CloudHSM erstellt eine Sicherungskopie eines vorhandenen HSM im Cluster. Weitere Informationen finden Sie unter [Sicherungen](#).
3. AWS CloudHSM stellt das Backup auf dem neuen HSM wieder her. Auf diese Weise wird sichergestellt, dass das HSM mit den anderen im Cluster synchron ist.
4. Die HSMs im Cluster vorhandenen Geräte benachrichtigen den AWS CloudHSM Client, dass es ein neues HSM im Cluster gibt.
5. Der Client stellt eine Verbindung mit dem neuen HSM her.

So fügen Sie ein HSM hinzu (Konsole)

1. Öffnen Sie die AWS CloudHSM Konsole zu <https://console.aws.amazon.com/cloudhsm/Hause>.
2. Wählen Sie einen Cluster für das HSM aus, das Sie hinzufügen möchten.
3. Wählen Sie auf der HSMsRegisterkarte Create HSM aus.
4. Wählen Sie eine Availability Zone (AZ) für das HSM, das Sie erstellen. Wählen Sie die Option Erstellen aus.

So fügen Sie ein HSM hinzu (AWS CLI)

- Geben Sie an der Eingabeaufforderung den Befehl [create-hsm](#) samt einer Cluster-ID sowie einer Availability Zone für das HSM ein, das Sie erstellen. Wenn Sie die Cluster-ID des bevorzugten Clusters nicht kennen, verwenden Sie den [describe-clusters](#)-Befehl. Geben Sie die Availability Zone in der Form us-east-2a, us-east-2b, usw. an.

```
$ 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"  
  }  
}
```

Um ein HSM (AWS CloudHSM API) hinzuzufügen

- Senden Sie eine [CreateHsm](#)-Anforderung und geben Sie die Cluster-ID und eine Availability Zone für das HSM an, das Sie erstellen.

Ein HSM aus einem Cluster entfernen AWS CloudHSM

Sie können ein HSM mithilfe der [AWS CloudHSM Konsole](#) [AWS CLI](#), der oder der AWS CloudHSM API entfernen.

So entfernen Sie ein HSM (Konsole)

1. Öffnen Sie die AWS CloudHSM Konsole zu <https://console.aws.amazon.com/cloudhsm/Hause>.
2. Wählen Sie den Cluster mit dem HSM aus, das Sie entfernen möchten.
3. Wählen Sie auf der HSMsRegisterkarte das HSM aus, das Sie entfernen möchten. Wählen Sie dann HSM löschen aus.
4. Bestätigen Sie, dass Sie das HSM löschen möchten. Wählen Sie dann Löschen.

So entfernen Sie ein HSM (AWS CLI)

- Geben Sie in der Eingabeaufforderung den Befehl [delete-hsm](#) ein. Geben Sie die ID des Clusters mit dem HSM, das Sie löschen möchten, und eine der folgenden HSM-IDs ein:
 - Die HSM-ID (`--hsm-id`)
 - Die HSM-IP-Adresse (`--eni-ip`)
 - Die Elastic-Network-Schnittstellen-ID des HSM (`--eni-id`)

Wenn Sie die Werte für diese IDs nicht kennen, geben Sie den Befehl [describe-clusters](#) ein.

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

Um ein HSM (AWS CloudHSM API) zu entfernen

- Senden Sie eine [DeleteHsm](#)-Anforderung und geben Sie die Cluster-ID und eine ID Zone für das HSM an, das Sie löschen möchten.

Löschen eines AWS CloudHSM Clusters

Bevor Sie einen Cluster löschen können, müssen Sie alle Cluster HSMs aus dem Cluster entfernen. Weitere Informationen finden Sie unter [Ein HSM aus einem Cluster entfernen AWS CloudHSM](#).

Nachdem Sie alle entfernt haben HSMs, können Sie einen Cluster mithilfe der [AWS CloudHSM Konsole](#), der [AWS Command Line Interface \(AWS CLI\)](#) oder der AWS CloudHSM API löschen.

So löschen Sie einen Cluster (Konsole)

1. Öffnen Sie die AWS CloudHSM Konsole zu <https://console.aws.amazon.com/cloudhsm/Hause>.
2. Wählen Sie den Cluster aus, den Sie löschen möchten. Wählen Sie dann Cluster löschen aus.
3. Bestätigen Sie, dass Sie den Cluster löschen möchten, und wählen Sie dann Löschen aus.

Löschen eines Clusters (AWS CLI)

- Geben Sie an der Eingabeaufforderung den Befehl [delete-cluster](#) ein und übergeben Sie die ID des Clusters, das Sie löschen möchten. Wenn Sie die Cluster-ID nicht kennen, geben Sie den Befehl [describe-clusters](#) ein.

```
$ 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"
  }
}
```



```
}  
}
```

So löschen Sie einen AWS CloudHSM -Cluster (API)

- Senden Sie eine [DeleteCluster](#)-Anforderung, wobei Sie die ID des Clusters angeben, den Sie löschen möchten.

AWS CloudHSM Cluster aus Backups erstellen

Gehen Sie wie in diesem Thema beschrieben vor, um einen AWS CloudHSM Cluster aus einer Sicherung wiederherzustellen. Ihr Cluster enthält die gleichen Benutzer, das gleiche Schlüsselmateriale, die gleichen Zertifikate, die gleiche Konfiguration und die gleichen Richtlinien wie bei der Sicherung. Weitere Informationen zur Verwaltung von Sicherungen finden Sie unter [Sicherungen von Clustern](#).

Cluster aus Sicherungen erstellen (Konsole)

1. Öffnen Sie die AWS CloudHSM Konsole zu <https://console.aws.amazon.com/cloudhsm/Hause>.
2. Wählen Sie Cluster erstellen.
3. Führen Sie im Abschnitt Cluster-Konfiguration Folgendes aus:
 - a. Wählen Sie für die VPC eine VPC für den Cluster aus, den Sie erstellen möchten.
 - b. Wählen Sie für die AZ(s) ein privates Subnetz für jede Availability Zone aus, die Sie dem Cluster hinzufügen möchten.
 - c. Wählen Sie unter Netzwerktyp das IP-Protokoll aus, das HSMs Sie für Verbindungen verwenden möchten.
4. Gehen Sie im Abschnitt Cluster source (Cluster-Quelle) wie folgt vor:
 - a. Wählen Sie Cluster von einer früheren Sicherung wiederherstellen aus.
 - b. Wählen Sie die Sicherung aus, die Sie wiederherstellen möchten.
5. Wählen Sie Weiter: Prüfen aus.
6. Überprüfen Sie Ihre Cluster-Konfiguration, dann wählen Sie Cluster erstellen aus.
7. Geben Sie an, wie lange der Dienst Sicherungen aufbewahren soll.

Akzeptieren Sie den standardmäßigen Aufbewahrungszeitraum von 90 Tagen oder geben Sie einen neuen Wert zwischen 7 und 379 Tagen ein. Der Dienst löscht automatisch Sicherungen in diesem Cluster, die älter sind als der hier angegebene Wert. Sie können dies später ändern. Weitere Informationen finden Sie unter [Konfigurieren Sie die Aufbewahrung von Backups](#).

8. Wählen Sie Weiter.
9. (Optional) Geben Sie einen Tag-Schlüssel und einen optionalen Tag-Wert ein. Wählen Sie Tag hinzufügen, wenn Sie mehr als ein Tag zum Cluster hinzufügen möchten.
10. Wählen Sie Überprüfen.
11. Überprüfen Sie Ihre Cluster-Konfiguration und wählen Sie dann Cluster erstellen aus.

Tip

Um in diesem Cluster ein HSM zu erstellen, das dieselben Benutzer, dasselbe Schlüsselmaterial, dieselben Zertifikate, dieselbe Konfiguration und dieselben Richtlinien enthält wie das Backup, das Sie wiederhergestellt haben, [fügen Sie dem Cluster ein HSM](#) hinzu.

Cluster aus Sicherungen erstellen (AWS CLI)

Geben Sie den [describe-backups](#)-Befehl ein, um die Sicherungs-ID zu ermitteln.

- Geben Sie in der Eingabeaufforderung den Befehl [create-cluster](#) ein. Geben Sie den HSM-Instanztyp, das Subnetz IDs der Subnetze, die Sie erstellen möchten HSMs, und die Backup-ID des Backups an, das Sie wiederherstellen.

```
$ 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"
  }
}

```

Erstellen Sie Cluster aus Backups (API)AWS CloudHSM

Im folgenden Thema erfahren Sie, wie Sie mithilfe der API Cluster aus Sicherungen erstellen.

- [CreateCluster](#)

Migration vom Typ Cluster-HSM

AWS CloudHSM bietet die Möglichkeit, den HSM-Typ eines vorhandenen Clusters zu ändern. Prüfen Sie anhand der Tabelle auf dieser Seite, ob die Änderung des HSM-Typs zulässig ist.

Weitere Informationen zu den HSMs unterstützten Typen und ihren Funktionen finden Sie [HSM gibt ein AWS CloudHSM](#) unter.

Note

Sie können den FIPS-Modus eines Clusters während dieses Vorgangs nicht ändern.

Aus	Bis	Comment
hsm1.medium	hsm2m. mittel	Erlaubt
hsm2m.medium	hsm1. mittel	Befriedigend. Sie können innerhalb von 24 Stunden nach Beginn einer Migration ein Rollback von hsm2m.medium auf hsm1.medium durchführen.

Topics

- [Migration von hsm1.medium zu hsm2m.medium](#)

Migration von hsm1.medium zu hsm2m.medium

Sie können Ihren Cluster von hsm1.medium auf hsm2m.medium migrieren. AWS CloudHSM In diesem Thema werden die Voraussetzungen, der Migrationsprozess und die Rollback-Verfahren beschrieben.

Bevor Sie mit der Migration beginnen, stellen Sie sicher, dass Ihre Anwendung den Empfehlungen unter entspricht. [Richten Sie Ihren Cluster auf Hochverfügbarkeit aus](#) Dies trägt dazu bei, Ausfallzeiten während des Prozesses zu vermeiden.

Note

Automatische Migrationen zu hsm2m.medium beginnen am 20. Januar 2026.

Überblick über den Migrationsprozess von hsm1.medium zu hsm2m.medium

Sie können die Migration mit der AWS CloudHSM Konsole, der oder der API starten. AWS CLI AWS CloudHSM Unabhängig davon, wo Sie sie initiieren, verwendet die AWS CloudHSM Clustermigration den `modify-cluster` API-Endpunkt. Alternativ migriert AWS CloudHSM den Cluster automatisch in Ihrem Namen. Sobald die Migration beginnt, wechselt Ihr gesamter Cluster in einen eingeschränkten Schreibmodus. Weitere Informationen finden Sie unter [Cluster-Modus mit eingeschränktem Schreibzugriff](#).

Um die Auswirkungen zu minimieren, AWS CloudHSM wechseln Sie nacheinander HSMs von hsm1.medium zu hsm2m.medium. Das Ersatzprodukt HSMs behält dieselben IP-Adressen bei, sodass während oder nach der Migration keine Konfigurationsänderungen erforderlich sind.

So funktioniert die Migration:

1. AWS CloudHSM Erstellt vor der Migration des ersten HSM ein vollständiges Backup des gesamten Clusters.
2. AWS CloudHSM Erstellt mithilfe dieser Sicherung ein neues HSM des angeforderten Typs (hsm2m.medium), um das erste HSM zu ersetzen.
3. AWS CloudHSM Erstellt vor der Migration jedes nachfolgenden HSM eine neue vollständige Sicherung des gesamten Clusters.
4. AWS CloudHSM wiederholt die Schritte 3 und 4 für jedes HSM im Cluster, wobei jeweils ein HSM migriert wird.
5. Jede einzelne HSM-Migration dauert ungefähr 30 Minuten.

AWS CloudHSM überwacht den Zustand des Clusters und führt während des gesamten Migrationsprozesses Validierungen durch. Wenn eine AWS CloudHSM Zunahme von Fehlern festgestellt wird oder eine Validierungsprüfung fehlschlägt, wird die Migration automatisch gestoppt und der Cluster wird auf seinen ursprünglichen HSM-Typ zurückgesetzt. Sie können nach dem Start der Migration auch bis zu 24 Stunden lang manuell ein Rollback durchführen. Vor dem Rollback finden Sie weitere Informationen unter [Überlegungen zum HSM-Rollback](#).

Voraussetzungen für die Migration zu hsm2m.medium

Ihr vorhandener AWS CloudHSM Cluster muss diese Anforderungen erfüllen, um zu hsm2m.medium zu migrieren. Wenn bei den Validierungsprüfungen eine Bedingung nicht erfüllt ist, AWS CloudHSM wird der Cluster automatisch auf seinen ursprünglichen HSM-Typ zurückgesetzt.

Eine Liste der bekannten Migrationsprobleme finden Sie unter [???](#)

- In den letzten 7 Tagen:
 - Alle Client-Verbindungen haben SDK 5.9 oder höher verwendet.
 - Wenn ECDSA Verify durchgeführt wurde, haben alle Client-Verbindungen SDK 5.13 oder höher verwendet.
 - AWS CloudHSM Instanzen haben nur unterstützte (und keine der veralteten) Funktionen verwendet. Einzelheiten finden Sie unter [Benachrichtigungen über veraltete Versionen](#).

- Sie müssen in den letzten 7 Tagen ein SDK verwendet haben, um eine Verbindung mit mindestens einem HSM im Cluster herzustellen.
- Der Cluster befindet sich im Status AKTIV.
- Der Cluster hat 27 HSMs oder weniger.
- Die Fehlerrate für HSM-Operationen steigt während der Migration nicht an.

 Note

Die vorherige Einschränkung, die Kunden mit Tokenschlüssel-Workloads an der Migration hinderte, wurde aufgehoben.

Cluster-Modus mit eingeschränktem Schreibzugriff

Wenn Ihr Cluster mit der Migration beginnt, wechselt er in einen eingeschränkten Schreibmodus. Operationen, die den HSM-Status ändern können, werden abgelehnt. Alle Lesevorgänge bleiben davon unberührt.

Während der Migration erhält Ihre Anwendung beim Versuch der folgenden Operationen einen Fehler vom HSM:

- Generierung und Löschung von Token-Schlüsseln (Sitzungsschlüssel-Workloads werden weiterhin ausgeführt).
- Alle Benutzererstellungen, Löschungen oder Änderungen.
- Quorum-Operationen.
- Änderung von Schlüsseln innerhalb des HSM, z. B. Änderung von Schlüsselattributen.
- mTLS-Registrierung.

AWS CloudHSM versetzt Ihren Cluster außerdem in einen `MODIFY_IN_PROGRESS` Status während der Migration. Während dieser Zeit können Sie dem Cluster weder etwas hinzufügen noch HSMs aus ihm entfernen.

Die Migration wird gestartet

Durch den Cluster-Migrationsprozess werden einzelne Personen HSMs in Ihrem Cluster nacheinander ersetzt. Die Dauer hängt von der Anzahl der Mitglieder HSMs in Ihrem Cluster ab. Im

Durchschnitt dauert dieser Vorgang etwa 30 Minuten pro HSM. Sie können den Fortschritt verfolgen, indem Sie den HSM-Typ der einzelnen Personen HSMs im Cluster überwachen, um zu sehen, wie viele Personen auf den neuen Typ migriert wurden.

Console

Um den HSM-Typ (Konsole) zu ändern

1. Öffnen Sie die AWS CloudHSM Konsole zu <https://console.aws.amazon.com/cloudhsm/Hause>.
2. Wählen Sie das Optionsfeld neben der ID des Clusters aus, den Sie ändern möchten
3. Wählen Sie Modify HSM Type im Menü Aktionen den gewünschten HSM-Typ aus

Durch dieses Verfahren wird Ihr Cluster in den MODIFY_IN_PROGRESS Status versetzt. Nach der Migration kehrt Ihr Cluster in den ACTIVE Status zurück.

AWS CLI

Um den HSM-Typ zu ändern () [AWS CLI](#)

- Führen Sie über die Eingabeaufforderung den folgenden [modify-cluster](#)-Befehl aus. Geben Sie die Cluster-ID und den gewünschten HSM-Typ an.

```
$ 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  
  }  
}
```

Dieses Verfahren versetzt Ihren Cluster in den MODIFY_IN_PROGRESS Status. Nach der Migration kehrt Ihr Cluster in den ACTIVE Status zurück.

AWS CloudHSM API

Um den HSM-Typ (AWS CloudHSM API) zu ändern

- Senden Sie eine [ModifyCluster](#)-Anforderung. Geben Sie die Cluster-ID und den gewünschten HSM-Typ für den Cluster an.

Durch dieses Verfahren wird Ihr Cluster in den MODIFY_IN_PROGRESS Status versetzt. Nach der Migration kehrt Ihr Cluster in den ACTIVE Status zurück.

Die Migration rückgängig machen

AWS CloudHSM überwacht erhöhte Fehlerraten und führt während der gesamten Migration kontinuierliche Validierungsprüfungen durch. Wenn AWS CloudHSM eine Verschlechterung der Servicequalität oder Validierungsfehler festgestellt werden, wird automatisch ein Rollback zum ursprünglichen HSM-Typ Ihres Clusters eingeleitet. Während eines Rollbacks gilt für jedes HSM im Cluster Folgendes:

- AWS CloudHSM verwendet das Backup, das zu Beginn der HSM-Migration erstellt wurde.
- Es ersetzt jeweils ein HSM, bis alle wieder auf den ursprünglichen Typ zurückgesetzt HSMs sind.
- Ihr Cluster bleibt während des gesamten Vorgangs im eingeschränkten Schreibmodus.

Sie können die Migration innerhalb von 24 Stunden nach dem Start rückgängig machen. Um die Frist für das Rollback zu überprüfen:

1. Führen Sie den Befehl [describe-clusters](#) aus.
2. Suchen Sie nach dem Wert `HsmTypeRollbackExpiration`. Dieser Zeitstempel ist Ihre Rollback-Frist.

Wenn Sie sich für ein Rollback entscheiden, tun Sie dies vor Ablauf dieser Frist. Das Rollback verwendet das neueste Backup Ihres ursprünglichen HSM-Typs.

Warning

Seien Sie vorsichtig, wenn Sie nach Abschluss einer Migration ein Rollback durchführen. Wenn Sie eine Migration abschließen und anschließend neue Schlüssel oder Benutzer erstellen, kann ein Rollback zu Datenverlust führen. AWS CloudHSM Unter [Daten nach einem Rollback synchronisieren](#) erfahren Sie, wie Sie den Datenverlust nach einem Rollback minimieren können.

Console

So führen Sie ein Rollback Ihres HSM-Typs (Konsole) durch

1. Öffnen Sie die AWS CloudHSM Konsole zu <https://console.aws.amazon.com/cloudhsm/Hause>.
2. Wählen Sie die ID des Clusters aus, für den Sie ein Rollback durchführen möchten.
3. Wählen Sie `Modify HSM Type` im Menü Aktionen den ursprünglichen HSM-Typ aus

Durch dieses Verfahren wird Ihr Cluster in den `ROLLBACK_IN_PROGRESS` Status versetzt. Nach dem Rollback kehrt Ihr Cluster in den `ACTIVE` Status zurück.

AWS CLI

Um Ihr HSM zurückzusetzen, geben Sie () ein [AWS CLI](#)

- Führen Sie über die Eingabeaufforderung den folgenden [modify-cluster](#)-Befehl aus. Geben Sie die Cluster-ID und den ursprünglichen HSM-Typ an.

```
$ 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
  }
}
```

Durch dieses Verfahren wird Ihr Cluster in den ROLLBACK_IN_PROGRESS Status versetzt. Nach dem Rollback kehrt Ihr Cluster in den ACTIVE Status zurück.

AWS CloudHSM API

Um Ihren HSM-Typ (AWS CloudHSM API) zurückzusetzen

- Senden Sie eine [ModifyCluster](#)-Anforderung. Geben Sie die Cluster-ID und den ursprünglichen HSM-Typ für den Cluster an.

Durch dieses Verfahren wird Ihr Cluster in den `ROLLBACK_IN_PROGRESS` Status versetzt. Nach dem Rollback kehrt Ihr Cluster in den `ACTIVE` Status zurück.

Daten werden nach einem Rollback synchronisiert

Sie HSMs befinden sich während der Migration im eingeschränkten Schreibmodus, wodurch Änderungen am HSM-Status verhindert werden. Wenn Sie während dieser Zeit (während der Cluster aktiv ist `MODIFY_IN_PROGRESS`) ein Rollback durchführen, entsteht ein Cluster, dessen Inhalt mit dem des ursprünglichen Clusters identisch ist.

Nachdem Ihr Cluster in den `ACTIVE` Status zurückgekehrt ist, wird der eingeschränkte Schreibmodus aufgehoben. Wenn Sie im `ACTIVE` Status einen Schlüssel oder Benutzer erstellen und dann ein Rollback durchführen, ist dieser Schlüssel oder Benutzer nicht in Ihrem Rollback-Cluster vorhanden.

Um die Benutzersynchronisierung zu beheben, verwenden Sie die [Benutzerverwaltung](#) mit der CloudHSM-CLI, um die fehlenden Benutzer in Ihrem Rollback-Cluster neu zu erstellen. Die Benutzer müssen manuell neu erstellt werden, da der `user replicate` Befehl die Synchronisierung von Benutzern von `hsm2m.medium` mit `hsm1.medium` nicht unterstützt. Weitere [Informationen](#) finden Sie unter [Bekannte Probleme durch Benutzer replizieren](#).

Um die Schlüsselsynchronisierung zu beheben, verwenden Sie den Befehl [key replicate](#), um einen Schlüssel zwischen zwei Clustern zu replizieren. Wenn Sie CloudHSM CLI nicht installiert haben, lesen Sie die Anweisungen unter [???](#)

Um Schlüssel nach dem Rollback zu synchronisieren

Gehen Sie nach Abschluss des Rollbacks wie folgt vor. Wir werden diese Begriffe verwenden:

- „cluster-1“: Ihr Rollback-Cluster (jetzt `hsm1.medium`)
- „cluster-2“: Ein neuer temporärer `hsm2m.medium`-Cluster, den Sie erstellen werden

1. Erstellen Sie einen neuen `hsm2m.medium`-Cluster (Cluster-2) mit dem neuesten `hsm2m.medium`-Backup von Cluster-1:

```
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. Erstellen Sie ein HSM in Cluster-2:

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

3. Listet die Schlüssel in Cluster-2 auf, die repliziert werden müssen:

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

4. Replizieren Sie jeden Schlüssel von Cluster-2 nach Cluster-1:

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

5. Wiederholen Sie Schritt 4 für jeden Schlüssel, der kopiert werden muss.
6. Löschen Sie das HSM in Cluster-2:

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

7. Cluster-2 löschen:

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

HSM-Benutzer in AWS CloudHSM

Bevor Sie Ihren AWS CloudHSM Cluster für die Kryptoverarbeitung verwenden können, müssen Sie Benutzer und [Schlüssel](#) auf den Hardware-Sicherheitsmodulen (HSM) in Ihrem Cluster erstellen.

Note

HSM-Benutzer unterscheiden sich von IAM-Benutzern. IAM-Benutzer, die über die richtigen Anmeldeinformationen verfügen, können HSMs durch Interaktion mit Ressourcen über die AWS-API etwas erstellen. Nachdem das HSM erstellt wurde, müssen Sie HSM-Benutzeranmeldedaten verwenden, um Vorgänge auf dem HSM zu authentifizieren.

In AWS CloudHSM müssen Sie die Befehlszeilentools [CloudHSM CLI](#) oder [CloudHSM Management Utility \(CMU\)](#) verwenden, um die Benutzer auf Ihrem HSM zu erstellen und zu verwalten. CloudHSM-CLI ist für die Verwendung mit [der neuesten SDK-Versionsserie](#) konzipiert, während die CMU für die Verwendung mit [der vorherigen SDK-Versionsserie](#) konzipiert ist.

Weitere Informationen zur Verwaltung von HSM-Benutzern finden Sie in den folgenden Themen. AWS CloudHSM Außerdem erfahren Sie, wie Sie die Quorum-Authentifizierung verwenden.

Themen

- [HSM-Benutzerverwaltung mit CloudHSM CLI](#)
- [HSM-Benutzerverwaltung mit CloudHSM Management Utility \(CMU\)](#)

HSM-Benutzerverwaltung mit CloudHSM CLI

[Um Benutzer des Hardware Security Module \(HSM\) zu verwalten AWS CloudHSM, müssen Sie sich mit dem Benutzernamen und dem Passwort eines Administrators beim HSM anmelden.](#) Nur Administratoren können Benutzer verwalten. Das HSM enthält einen Standard-Admin namens admin. Sie haben das Passwort für admin festgelegt, wenn Sie das [Cluster aktiviert haben](#).

Dieses Thema enthält step-by-step Anleitungen und Einzelheiten zur Verwaltung von HSM-Benutzern mit der CloudHSM-CLI.

Themen

- [Voraussetzungen für die Benutzerverwaltung in CloudHSM CLI](#)
- [HSM-Benutzertypen für CloudHSM CLI](#)
- [Tabelle mit HSM-Benutzerberechtigungen für CloudHSM CLI](#)
- [Erstellen Sie mit der CloudHSM-CLI einen HSM-Benutzeradministrator](#)
- [Erstellen Sie einen HSM-Kryptobenutzer mit der CloudHSM-CLI](#)
- [Alle HSM-Benutzer im Cluster mithilfe der CloudHSM-CLI auflisten](#)
- [Ändern Sie HSM-Benutzerkennwörter mit der CloudHSM-CLI](#)
- [HSM-Benutzer mithilfe der CloudHSM-CLI löschen](#)
- [MFA für HSM-Benutzer mithilfe der CloudHSM-CLI verwalten](#)
- [Verwaltung der Quorum-Authentifizierung \(M-of-N-Zugriffskontrolle\) mithilfe der CloudHSM-CLI](#)

Voraussetzungen für die Benutzerverwaltung in CloudHSM CLI

Bevor Sie CloudHSM CLI verwenden, um Benutzer von Hardware-Sicherheitsmodulen (HSM) zu verwalten AWS CloudHSM, müssen Sie diese Voraussetzungen erfüllen. In den folgenden Themen werden die ersten Schritte mit der CloudHSM-CLI beschrieben.

Themen

- [Rufen Sie die IP-Adresse eines HSM ab in AWS CloudHSM](#)
- [Laden Sie die CloudHSM CLI herunter](#)

Rufen Sie die IP-Adresse eines HSM ab in AWS CloudHSM

Um die CloudHSM CLI zu verwenden, müssen Sie das Configure-Tool verwenden, um die lokale Konfiguration zu aktualisieren. Anweisungen zum Ausführen des Configure-Tools mit der CloudHSM CLI finden Sie unter [Erste Schritte mit der AWS CloudHSM Befehlszeilenschnittstelle \(CLI\)](#). Für den `-a`-Parameter müssen Sie die IP-Adresse eines HSM in Ihrem Cluster hinzufügen. Wenn Sie mehrere haben HSMs, können Sie eine beliebige IP-Adresse verwenden. Dadurch wird sichergestellt, dass CloudHSM CLI alle Änderungen, die Sie vornehmen, auf den gesamten Cluster übertragen kann. Denken Sie daran, dass die CloudHSM-CLI ihre lokale Datei verwendet, um Clusterinformationen zu verfolgen. Wenn sich der Cluster seit der letzten Verwendung von CloudHSM CLI von einem bestimmten Host aus geändert hat, müssen Sie diese Änderungen zur lokalen Konfigurationsdatei hinzufügen, die auf diesem Host gespeichert ist. Entfernen Sie niemals ein HSM, während Sie die CloudHSM-CLI verwenden.

Um eine IP-Adresse für ein HSM (Konsole) zu erhalten

1. Öffnen Sie die AWS CloudHSM Konsole zu <https://console.aws.amazon.com/cloudhsm/Hause>.
2. Um die AWS-Region zu ändern, verwenden Sie die Regionsauswahl in der oberen rechten Ecke der Seite.
3. Um die Cluster-Detailseite zu öffnen, wählen Sie in der Cluster-Tabelle die Cluster-ID aus.
4. Um die IP-Adresse zu erhalten, gehen Sie zur HSMs Registerkarte. Wählen Sie für IPv4 Cluster eine Adresse aus, die unter IPv4 ENI-Adresse aufgeführt ist. Verwenden Sie für Dual-Stack-Cluster entweder die ENI IPv4 - oder die IPv6 ENI-Adresse.

Um eine IP-Adresse für ein HSM zu erhalten (AWS CLI)

- Rufen Sie die IP-Adresse eines HSM ab, indem Sie den [describe-clusters](#) Befehl von der AWS CLI verwenden. In der Ausgabe des Befehls entspricht die IP-Adresse von den Werten von `EniIp` und `EniIpV6` (falls es HSMs sich um einen Dual-Stack-Cluster handelt).

```
$ 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",
...
      }
    ]
  }
}
```

Laden Sie die CloudHSM CLI herunter

Die neueste Version von CloudHSM CLI ist für HSM-Benutzerverwaltungsaufgaben für Client-SDK 5 verfügbar. Um CloudHSM-CLI herunterzuladen und zu installieren, folgen Sie den Anweisungen unter [CloudHSM CLI installieren und konfigurieren](#).

HSM-Benutzertypen für CloudHSM CLI

Für die meisten Operationen, die Sie auf dem Hardware Security Module (HSM) ausführen, sind die Anmeldeinformationen eines AWS CloudHSM HSM-Benutzers erforderlich. Das HSM authentifiziert jeden HSM-Benutzer, und jeder HSM-Benutzer hat einen Typ, der bestimmt, welche Operationen Sie als dieser Benutzer auf dem HSM ausführen können.

Note

HSM-Benutzer unterscheiden sich von IAM-Benutzern. IAM-Benutzer, die über die richtigen Anmeldeinformationen verfügen, können HSMs durch Interaktion mit Ressourcen über die AWS-API etwas erstellen. Nachdem das HSM erstellt wurde, müssen Sie HSM-Benutzeranmeldedaten verwenden, um Vorgänge auf dem HSM zu authentifizieren.

Benutzertypen

- [Nicht aktivierter Admin](#)
- [Admin.](#)
- [Crypto-Benutzer \(Crypto User, CU\)](#)
- [Appliance-Benutzer \(Appliance User, AU\)](#)

Nicht aktivierter Admin

In der CloudHSM CLI ist der nicht aktivierte Administrator ein temporärer Benutzer, der nur auf dem ersten HSM in einem AWS CloudHSM -Cluster existiert, der noch nie aktiviert wurde. Um [einen Cluster zu aktivieren](#), führen Sie den `cluster activate`-Befehl in der CloudHSM CLI aus. Nach der Ausführung dieses Befehls werden nicht aktivierte Administratoren aufgefordert, das Passwort zu ändern. Nach dem Ändern des Passworts wird der inaktivierte Administrator zum Administrator.

Admin.

In der CloudHSM CLI kann der Administrator Benutzerverwaltungsvorgänge durchführen. Beispielsweise können Sie Benutzer erstellen und löschen und Benutzerpasswörter ändern. Weitere Informationen über Admins finden Sie in der [Tabelle mit HSM-Benutzerberechtigungen für CloudHSM CLI](#).

Crypto-Benutzer (Crypto User, CU)

Ein Crypto-Benutzer (CU) kann die folgenden Schlüsselverwaltungs- und kryptografischen Vorgänge ausführen.

- Schlüsselverwaltung – Erstellen, Löschen, Freigeben, Importieren und Exportieren von kryptographischen Schlüsseln.
- Kryptografische Operationen – Verwenden Sie kryptographische Schlüssel für die Verschlüsselung, Entschlüsselung, Signatur, Verifizierung und mehr.

Weitere Informationen hierzu finden Sie unter [Tabelle mit HSM-Benutzerberechtigungen für CloudHSM CLI](#).

Appliance-Benutzer (Appliance User, AU)

Der Appliance-Benutzer (AU) kann Kloning- und Synchronisierungsvorgänge auf Ihrem Cluster durchführen. HSMs AWS CloudHSM verwendet die AU, um die HSMs in einem AWS CloudHSM Cluster zu synchronisieren. Die AU ist auf allen Geräten vorhanden, die von HSMs bereitgestellt werden AWS CloudHSM, und verfügt über eingeschränkte Berechtigungen. Weitere Informationen hierzu finden Sie unter [Tabelle mit HSM-Benutzerberechtigungen für CloudHSM CLI](#).

AWS kann keine Operationen an Ihrem ausführen HSMs . AWS kann Ihre Benutzer oder Schlüssel nicht anzeigen oder ändern und mit diesen Schlüsseln keine kryptografischen Operationen ausführen.

Tabelle mit HSM-Benutzerberechtigungen für CloudHSM CLI

In der folgenden Tabelle sind die HSM-Operationen (Hardware Security Module) aufgeführt, sortiert nach dem Typ des HSM-Benutzers oder der HSM-Sitzung, in der der Vorgang ausgeführt werden kann. AWS CloudHSM

	Admin.	Crypto-Benutzer (Crypto User, CU)	Appliance -Benutzer (Appliance User, AU)	Nicht authentif izierte Sitzungen
Grundlege nde Cluster-I nformationen erhalten ¹	 Ja	 Ja	 Ja	 Ja
Das eigene Passwort ändern	 Ja	 Ja	 Ja	Nicht zutreffend
Das Passwort eines Benutzers ändern	 Ja	 Nein	 Nein	 Nein
Benutzer hinzufügen oder entfernen	 Ja	 Nein	 Nein	 Nein
Den Synchroni sierungsstatus erhalten ²	 Ja	 Ja	 Ja	 Nein
Maskierte Objekte extrahier en und einfügen ³	 Ja	 Ja	 Ja	 Nein

	Admin.	Crypto-Benutzer (Crypto User, CU)	Appliance -Benutzer (Appliance User, AU)	Nicht authentif izierte Sitzungen
Funktionen zur Schlüssel verwaltung ⁴	 Nein	 Ja	 Nein	 Nein
Verschlüsseln/ Entschlüsseln	 Nein	 Ja	 Nein	 Nein
Signieren/ Überprüfen	 Nein	 Ja	 Nein	 Nein
Generieren Sie Digests und HMACs	 Nein	 Ja	 Nein	 Nein

- [1] Zu den grundlegenden Clusterinformationen gehören die Anzahl der HSMs Mitglieder im Cluster und die IP-Adresse, das Modell, die Seriennummer, die Geräte-ID, die Firmware-ID usw. jedes HSM.
- [2] Der Benutzer kann verschiedene Digests (Hashes) erhalten, die den Schlüsseln auf dem HSM entsprechen. Eine Anwendung kann diese Gruppen von Digests vergleichen, um den Synchronisationsstatus innerhalb eines HSMs Clusters zu ermitteln.
- [3] Maskierte Objekte sind Schlüssel, die verschlüsselt werden, bevor sie das HSM verlassen. Sie können außerhalb des HSM nicht entschlüsselt werden. Sie werden erst entschlüsselt, nachdem sie in ein HSM eingefügt wurden, das sich in demselben Cluster befindet wie das HSM, aus dem sie extrahiert wurden. Eine Anwendung kann maskierte Objekte extrahieren und einfügen, um sie HSMs in einem Cluster zu synchronisieren.

- [4] Die Funktionen zur Schlüsselverwaltung umfassen unter anderem das Erstellen, Löschen, Verpacken, Entpacken und Ändern der Schlüsselattribute.

Erstellen Sie mit der CloudHSM-CLI einen HSM-Benutzeradministrator

Gehen Sie wie folgt vor, um mithilfe der CloudHSM-CLI einen Administratorbenutzer für das Hardware Security Module (HSM) zu erstellen.

1. Verwenden Sie den folgenden Befehl, um den interaktiven CloudHSM-CLI-Modus zu starten.

Linux

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

Windows

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

2. Verwenden Sie den login-Befehl und melden Sie sich beim Cluster als Administrator an.

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

3. Sie werden vom System aufgefordert, Ihr Passwort einzugeben. Geben Sie das Passwort ein und die Ausgabe zeigt, dass der Befehl erfolgreich war.

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

4. Geben Sie den folgenden Befehl ein, um einen Admin zu erstellen.

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

5. Geben Sie das Passwort für den neuen Benutzer ein.

6. Geben Sie das Passwort erneut ein, um zu bestätigen, dass das eingegebene Passwort korrekt ist.

Erstellen Sie einen HSM-Kryptobenzutzer mit der CloudHSM-CLI

Gehen Sie wie folgt vor, um mithilfe der CloudHSM-CLI einen Hardware-Sicherheitsmodul (HSM) - Kryptobenzutzer (CU) zu erstellen.

1. Verwenden Sie den folgenden Befehl, um den interaktiven CloudHSM-CLI-Modus zu starten.

Linux

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

Windows

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

2. Verwenden Sie den login-Befehl und melden Sie sich beim Cluster als Administrator an.

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

3. Sie werden vom System aufgefordert, Ihr Passwort einzugeben. Geben Sie das Passwort ein und die Ausgabe zeigt, dass der Befehl erfolgreich war.

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

4. Geben Sie den folgenden Befehl ein, um einen Crypto-Benutzer zu erstellen.

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

5. Geben Sie das Passwort für den neuen Crypto-Benutzer ein.

6. Geben Sie das Passwort erneut ein, um zu bestätigen, dass das eingegebene Passwort korrekt ist.

Alle HSM-Benutzer im Cluster mithilfe der CloudHSM-CLI auflisten

Verwenden Sie den `user list` Befehl in der CloudHSM-CLI, um alle Benutzer im AWS CloudHSM Cluster aufzulisten. Sie müssen sich nicht anmelden, um `user list` auszuführen. Alle Benutzertypen können Benutzer auflisten.

Gehen Sie wie folgt vor, um alle Benutzer im Cluster aufzulisten

1. Verwenden Sie den folgenden Befehl, um den interaktiven CloudHSM-CLI-Modus zu starten.

Linux

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

Windows

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

2. Geben Sie den folgenden Befehl ein, um alle Benutzer im Cluster aufzulisten:

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

Für weitere Informationen über `user list` finden Sie in der [Benutzerliste](#).

Ändern Sie HSM-Benutzerkennwörter mit der CloudHSM-CLI

Verwenden Sie den `user change-password` Befehl in der CloudHSM-CLI, um das Passwort eines HSM-Benutzers (Hardware Security Module) zu ändern.

Bei Benutzertypen und Passwörtern wird zwischen Groß- und Kleinschreibung unterschieden, bei Benutzernamen jedoch nicht.

Admin, Crypto-Benutzer (CU) und Appliance-Benutzer (AU) können ihr eigenes Passwort ändern. Um das Passwort eines anderen Benutzers zu ändern, müssen Sie sich als Administrator anmelden. Sie können das Passwort eines Benutzers, der gerade angemeldet ist, nicht ändern.

So ändern Sie Ihr eigenes Passwort

1. Verwenden Sie den folgenden Befehl, um den interaktiven CloudHSM-CLI-Modus zu starten.

Linux

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

Windows

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

2. Verwenden Sie den login-Befehl und melden Sie sich als Benutzer mit dem Passwort an, das Sie ändern möchten.

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

3. Geben Sie das Passwort des Benutzers ein.

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

4. Geben Sie den user change-password-Befehl ein.

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

5. Geben Sie das neue Passwort ein.
6. Geben Sie das neue Passwort erneut ein.

So ändern Sie das Passwort eines anderen Benutzers

1. Verwenden Sie den folgenden Befehl, um den interaktiven CloudHSM-CLI-Modus zu starten.

Linux

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

Windows

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

2. Melden Sie sich mit der CloudHSM-CLI als Administrator an.

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

3. Geben Sie den user change-password-Befehl zusammen mit dem Benutzernamen des Benutzers ein, dessen Passwort Sie ändern möchten.

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

4. Geben Sie das neue Passwort ein.
5. Geben Sie das neue Passwort erneut ein.

Weitere Informationen über user change-password finden Sie unter [user change-password](#).

HSM-Benutzer mithilfe der CloudHSM-CLI löschen

Verwenden Sie user delete in der CloudHSM-CLI, um einen HSM-Benutzer (Hardware Security Module) zu löschen. Sie müssen sich als Administrator anmelden, um einen anderen Benutzer zu löschen.

Tip

Sie können keine Crypto-Benutzer (CU) löschen, die Schlüssel besitzen.

So löschen Sie einen Benutzer

1. Verwenden Sie den folgenden Befehl, um den interaktiven CloudHSM-CLI-Modus zu starten.

Linux

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

Windows

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

2. Verwenden Sie den login-Befehl und melden Sie sich beim Cluster als Administrator an.

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

3. Sie werden vom System aufgefordert, Ihr Passwort einzugeben. Geben Sie das Passwort ein und die Ausgabe zeigt, dass der Befehl erfolgreich war.

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

4. Verwenden Sie den Befehl user delete, um den Benutzer zu löschen.

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

Für weitere Informationen über user delete finden Sie unter [deleteUser](#).

MFA für HSM-Benutzer mithilfe der CloudHSM-CLI verwalten

Um die Sicherheit zu erhöhen, können Sie die Multi-Faktor-Authentifizierung (MFA) für Benutzer konfigurieren, um den AWS CloudHSM Cluster zu schützen.

Wenn Sie sich mit einem MFA-fähigen HSM-Benutzerkonto (Hardware Security Module) bei einem Cluster anmelden, geben Sie der CloudHSM-CLI Ihr Passwort — den ersten Faktor, was Sie wissen — und die CloudHSM-CLI stellt Ihnen ein Token zur Verfügung und fordert Sie auf, das Token signieren zu lassen.

Um den zweiten Faktor bereitzustellen – was Sie haben – signieren Sie das Token mit einem privaten Schlüssel aus einem Schlüsselpaar, das Sie bereits erstellt und dem HSM-Benutzer zugeordnet haben. Um auf den Cluster zuzugreifen, stellen Sie der CloudHSM CLI das signierte Token zur Verfügung.

Weitere Informationen zur Einrichtung von MFA für einen Benutzer finden Sie unter [MFA für CloudHSM-CLI einrichten](#)

Die folgenden Themen enthalten weitere Informationen zur Arbeit mit der Quorumauthentifizierung in AWS CloudHSM

Themen

- [Quorum-Authentifizierung und MFA in AWS CloudHSM Clustern mit CloudHSM CLI](#)
- [Anforderungen an das MFA-Schlüsselpaar für die AWS CloudHSM Verwendung von CloudHSM CLI](#)
- [MFA für CloudHSM-CLI einrichten](#)
- [Benutzer mit aktiviertem MFA für CloudHSM CLI erstellen](#)
- [Melden Sie Benutzer mit aktiviertem MFA für CloudHSM CLI an](#)
- [Schlüssel für Benutzer mit aktiviertem MFA für CloudHSM CLI rotieren](#)
- [Einen öffentlichen MFA-Schlüssel mithilfe der CloudHSM-CLI deregistrieren](#)
- [Token-Dateireferenz für MFA mit CloudHSM CLI](#)

Quorum-Authentifizierung und MFA in AWS CloudHSM Clustern mit CloudHSM CLI

Der AWS CloudHSM Cluster verwendet denselben Schlüssel für die Quorum-Authentifizierung und für die Multi-Faktor-Authentifizierung (MFA). Das bedeutet, dass ein Benutzer mit aktiviertem MFA effektiv für MoFN oder Quorum Access Control registriert ist. Beachten Sie die folgenden Punkte, um MFA und die Quorum-Authentifizierung erfolgreich für denselben HSM-Benutzer zu verwenden:

- Wenn Sie heute die Quorum-Authentifizierung für einen Benutzer verwenden, sollten Sie dasselbe Schlüsselpaar verwenden, das Sie für den Quorumbenutzer erstellt haben, um MFA für den Benutzer zu aktivieren.

- Wenn Sie die MFA-Anforderung für einen Nicht-MFA-Benutzer hinzufügen, der kein Quorum-Authentifizierungsbutzer ist, registrieren Sie diesen Benutzer als Quorum (MoFN) registrierten Benutzer mit MFA-Authentifizierung.
- Wenn Sie die MFA-Anforderung aufheben oder das Passwort für einen MFA-Benutzer ändern, der auch ein registrierter Quorum-Authentifizierungsbutzer ist, entfernen Sie auch die Registrierung des Benutzers als Quorum-Benutzer (MoFN).
- Wenn Sie die MFA-Anforderung aufheben oder das Passwort für einen MFA-Benutzer ändern, der auch ein Quorum-Authentifizierungsbutzer ist, Sie aber trotzdem möchten, dass dieser Benutzer an der Quorum-Authentifizierung teilnimmt, müssen Sie diesen Benutzer erneut als Quorum-Benutzer (MofN) registrieren.

Weitere Informationen über die Quorum-Authentifizierung finden Sie unter [Quorum-Authentifizierung verwalten \(M von N\)](#).

Anforderungen an das MFA-Schlüsselpaar für die AWS CloudHSM Verwendung von CloudHSM CLI

Um die Multi-Faktor-Authentifizierung (MFA) für einen Benutzer eines Hardware-Sicherheitsmoduls (HSM) zu aktivieren AWS CloudHSM, können Sie ein neues key pair erstellen oder einen vorhandenen Schlüssel verwenden, der die folgenden Anforderungen erfüllt:

- Schlüsseltyp: Asymmetrisch
- Schlüsselnutzung: Signieren und verifizieren
- Schlüsselspezifikation: RSA_2048
- Der Signaturalgorithmus umfasst: SHA256with RSAEncryption

Note

Wenn Sie die Quorumauthentifizierung verwenden oder planen, die Quorumauthentifizierung zu verwenden, finden Sie weitere Informationen unter [Quorum-Authentifizierung und MFA in AWS CloudHSM Clustern mit CloudHSM CLI](#).

Sie können die CloudHSM-CLI und das Schlüsselpaar verwenden, um einen neuen Admin-Benutzer mit aktiviertem MFA zu erstellen.

MFA für CloudHSM-CLI einrichten

Gehen Sie wie folgt vor, um die Multi-Faktor-Authentifizierung (MFA) für CloudHSM CLI einzurichten.

1. Um MFA mithilfe der Token-Sign-Strategie einzurichten, müssen Sie zunächst einen privaten 2048-Bit-RSA-Schlüssel und den zugehörigen öffentlichen Schlüssel generieren.

```
$ 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. Verwenden Sie den folgenden Befehl, um die CLI im interaktiven Modus zu starten.

Linux

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

Windows

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

3. Melden Sie sich mit der CloudHSM CLI bei Ihrem Benutzerkonto an.

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

4. Führen Sie als Nächstes den Befehl aus, um Ihre MFA-Strategie zu ändern. Sie müssen den Parameter `--token` angeben. Dieser Parameter gibt eine Datei an, in die unsignierte Token geschrieben werden.

```
aws-cloudhsm > user change-mfa token-sign --token unsigned-tokens.json --
username <username> --role crypto-user --change-quorum
Enter password:
Confirm password:
```

5. Sie haben jetzt eine Datei mit unsignierten Token, die signiert werden müssen: `unsigned-tokens.json`. Die Anzahl der Token in dieser Datei hängt von der Anzahl der Token in Ihrem Cluster ab. HSMs Jedes Token steht für ein HSM. Diese Datei ist im JSON-Format und enthält Token, die signiert werden müssen, um nachzuweisen, dass Sie über einen privaten Schlüssel verfügen.

```
$ cat unsigned-tokens.json
{
  "version": "2.0",
  "tokens": [
    {
      {
        "unsigned": "Vtf/9Q0FY45v/E1osvpEMr59JsnP/hLDm4It002vqL8=",
        "signed": ""
      },
      {
        "unsigned": "wVbC0/5IKwjyZK2NBpdFLyI7BiayZ24YcdUdlcxLwZ4=",
        "signed": ""
      },
      {
        "unsigned": "z6aW9RzErJBL5KqFG5h8lhTVt9oLbxppjod0Ebysydw=",
        "signed": ""
      }
    ]
  }
}
```

6. Der nächste Schritt besteht darin, diese Token mit dem in Schritt 1 erstellten privaten Schlüssel zu signieren. Platzieren Sie die Signaturen wieder in der Datei. Zuerst müssen Sie die Base64-codierten Token extrahieren und dekodieren.

```
$ echo "Vtf/9Q0FY45v/E1osvpEMr59JsnP/hLDm4It002vqL8=" > 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. Jetzt haben Sie binäre Token, die Sie mit dem in Schritt 1 erstellten privaten RSA-Schlüssel signieren können.

```
$ 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. Jetzt haben Sie binäre Signaturen der Token. Sie müssen sie mit Base64 codieren und sie dann wieder in Ihre Tokendatei einfügen.

```
$ 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. Schließlich können Sie die Base64-Werte kopieren und wieder in Ihre Token-Datei einfügen:

```
{
  "version": "2.0",
  "tokens": [
    {
      "unsigned": "1jqwx9bJ0UUQLiNb7mxXS1uBJsEXh0B9nj05BqnPsE=",
      "signed": "e1w3fZeCKIY50C4zPeg9Rt90M1Q1q3W1Jh6Yw7xXm4nF6e9ETLE39+9M
+rUqDWMRZjaBfaMbg5d9yDkz5p13U7ch2t1F9LoYabsWutkT014KRq/rcYMvFsU9n/Ey/
TK0PVaxLN42X+pebV4juwMhN4mK4CzdFAJgM+UGB0j4yB9recp0BB9K8QFSpJZALSEdDgUc/
mS1eDq3rU0int6+4NKuLQjpR
```

```
+LSEIWRZ6g6+MND2vXGskxHjadCQ09L7Tz8VcWjKDbxJcBiGKvkqyozl9zrGo8fA3WHBmwiAgS61Merx77ZGY4PFR37
YMSC14prCN15DtMRv2xA1SGSb4w=="
  },
  {
    "unsigned": "LMMFc34ASpvnNPFzBbMbr9FProS/Zu2P8zF/xzk5hVQ=",
    "signed": "HBImKnHmw+6R2TpFEpfiAg4+hu2pFNwn43ClhKPkn2higbEhUD0JVi
+4MerSyvU/NN79iWVxDvJ9Ito+jpiRQjTfTGEoIteyuAr1v/Bzh+Hjmr0530QpZaJ/VXGIgApD0myuu/
ZGNKQTCSSkL7+V81FG7yR1Nm22jUeGa735zvm/E+cenvZdy0VVx6A7WeWrl3JEKKBweHbi+7BwbaW
+PTdCuIRd4Ug76Sy+cFhsvcG1k7cMwDh8MgXzIZ2m1f/hdy2j8qAx0RTLlmwyU0YvPY0vUhc
+s83hx36QpGwGcD7RA0bPT50rTx7PHd0N1CL+Wwy91We8yIOFBS6nxo1R7w=="
  },
  {
    "unsigned": "dzeHbwhiVXQqcUGj563z51/7sLUdxjL93Sb0UyZRjH8=",
    "signed": "VgQPvrTsvG1jVBFxHnsduq16x8ZrnxfcYVYGf/
N7gEzI4At3GDs2EVZWTRdvS0uGHdkFYp1apHgJZ7PDVmGcTkIXVD2lFYppcgN1SzkY1ftr5E0jqS9ZjYEggGuB4g//
MxaBaRbJai/6BlcE92NIdBusTtreIm3yTpjIXNAVoeRSnkfuw7wZcL96QoklNb1WUuSHw
+psUyeIVtIwFMHEfFoRC0t
+VhmnlnFnkjGPb9W3Aprw2dRRvFM3R2ZTDvMCi0YDzUCd43GftGq2LfxH3qSD51oFHglHQV0Y0jyVzz1Avub5HQdt0Q
  }
]
}
```

10. Jetzt, da Ihre Token-Datei alle erforderlichen Signaturen enthält, können Sie fortfahren. Geben Sie den Namen der Datei ein, die die signierten Token enthält, und drücken Sie die Eingabetaste. Geben Sie abschließend den Pfad Ihres öffentlichen Schlüssels ein.

```
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"
  }
}
```

Jetzt haben Sie Ihren Benutzer mit MFA eingerichtet.

```
{
  "username": "<username>",
  "role": "crypto-user",
  "locked": "false",
```

```

    "mfa": [
      {
        "strategy": "token-sign",
        "status": "enabled"
      }
    ],
    "cluster-coverage": "full"
  },

```

Benutzer mit aktiviertem MFA für CloudHSM CLI erstellen

Gehen Sie wie folgt vor, um AWS CloudHSM Benutzer mit aktivierter Multi-Faktor-Authentifizierung (MFA) zu erstellen.

1. Verwenden Sie die CloudHSM CLI, um sich als Administrator beim HSM anzumelden.
2. Verwenden Sie den [user create](#)-Befehl, um einen Benutzer Ihrer Wahl zu erstellen. Folgen Sie dann den Schritten unter [MFA für CloudHSM-CLI einrichten](#), um MFA für den Benutzer einzurichten.

Melden Sie Benutzer mit aktiviertem MFA für CloudHSM CLI an

Gehen Sie wie folgt vor, um AWS CloudHSM Benutzer mit aktivierter Multi-Faktor-Authentifizierung (MFA) anzumelden.

1. Verwenden Sie den [login mfa-token-sign](#)-Befehl in der CloudHSM-CLI, um den Anmeldevorgang mit MFA für einen Benutzer zu starten, der MFA aktiviert hat.

```

aws-cloudhsm > login --username <username> --role <role> mfa-token-sign --
token <unsigned-tokens.json>
Enter password:

```

2. Geben Sie Ihr Passwort ein. Sie werden dann aufgefordert, den Pfad zur Tokendatei einzugeben, die Tokenpaare enthält, wobei signierte unsigned/signed Token diejenigen sind, die mithilfe Ihres privaten Schlüssels generiert wurden.

```

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. Während Sie aufgefordert werden, den Pfad der signierten Token-Datei einzugeben, können Sie die Datei mit dem unsignierten Token in einem separaten Terminal überprüfen. Identifizieren Sie die Datei mit unsignierten Token, die signiert werden müssen: **<unsigned-tokens.json>**. Die Anzahl der Token in dieser Datei hängt von der Anzahl der Token HSMs in Ihrem Cluster ab. Jedes Token steht für ein HSM. Diese Datei ist im JSON-Format und enthält Token, die signiert werden müssen, um nachzuweisen, dass Sie über einen privaten Schlüssel verfügen.

```
$ cat <unsigned-tokens.json>
{
  "version": "2.0",
  "tokens": [
    {
      "unsigned": "Vtf/9Q0FY45v/E1osvpEMr59JsnP/hLDm4It002vqL8=",
      "signed": ""
    },
    {
      "unsigned": "wVbC0/5IKwjyZK2NBpdFLyI7BiayZ24YcdUdlcxLwZ4=",
      "signed": ""
    },
    {
      "unsigned": "z6aW9RzErJBL5KqFG5h8lhTVt9oLbxppjod0Ebysydw=",
      "signed": ""
    }
  ]
}
```

4. Signieren Sie die unsignierten Token mit dem in Schritt 2 erstellten privaten Schlüssel. Zuerst müssen Sie die Base64-codierte Token extrahieren und dekodieren.

```
$ echo "Vtf/9Q0FY45v/E1osvpEMr59JsnP/hLDm4It002vqL8=" > 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. Sie haben jetzt binäre Token. Signieren Sie sie mit dem privaten RSA-Schlüssel, den Sie zuvor in [Schritt 1 der MFA-Einrichtung](#) erstellt haben.

```
$ 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. Sie haben jetzt binäre Signaturen der Token. Kodieren Sie sie mit Base64 und platzieren Sie sie wieder in Ihrer Tokendatei.

```

$ 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. Kopieren Sie schließlich die base64-Werte und fügen Sie sie wieder in Ihre Token-Datei ein:

```

{
  "version": "2.0",
  "tokens": [
    {
      "unsigned": "1jqwx9bJ0UUQLiNb7mxXS1uBJSExh0B9nj05BqnPsE=",
      "signed": "eiw3fZeCKIY50C4zPeg9Rt90M1Q1q3W1Jh6Yw7xXm4nF6e9ETLE39+9M
+rUqDWMRZjaBfaMbg5d9yDkz5p13U7ch2t1F9LoYabsWutkT014KRq/rcYMvFsU9n/Ey/
TK0PVaxLN42X+pebV4juwMhN4mK4CzdFAJgM+UGB0j4yB9recp0BB9K8QFSpJZALSEdDgUc/
mS1eDq3rU0int6+4NKuLQjpR
+LSEIWRZ6g6+MND2vXGskxHjadCQ09L7Tz8VcWjKDbxJcBiGKvkqyozl9zrGo8fA3WHBmwiAgS61Merx77ZGY4PFR37
YMSC14prCN15DtMRv2xA1SGSb4w=="
    },
    {
      "unsigned": "LMMFc34ASpNvNPFzBbMbr9FProS/Zu2P8zF/xzk5hVQ=",
      "signed": "HBImKnHmw+6R2TpFEpfiAg4+hu2pFNwn43ClhKPkn2higbEhUD0JVi
+4MerSyvU/NN79iWvxDvJ9Ito+jpiRQjTfTGEoIteyuAr1v/Bzh+Hjmr0530QpZaJ/VXGIgApD0myuu/
ZGNKQTCskkL7+V81FG7yR1Nm22jUeGa735zvm/E+cenvZdy0VVx6A7WeWrl3JEKKBweHbi+7BwbaW

```

```
+PTdCuIRd4Ug76Sy+cFhsvcG1k7cMwDh8MgXzIZ2m1f/hdy2j8qAx0RTL1mwyU0YvPY0vUhc
+s83hx36QpGwGcD7RA0bPT50rTx7PHd0N1CL+Wwy91We8yI0FBS6nxo1R7w=="
  },
  {
    "unsigned": "dzeHbwhiVXQqcUGj563z51/7sLUdxjL93Sb0UyZRjH8=",
    "signed": "VgQPvrTsvGljVBFxHnsduq16x8ZrnxfcYVYGf/
N7gEzI4At3GDs2EVZWTRdvS0uGHdkFYp1apHgJZ7PDVmGcTkIXVD2lFYppcgN1SzkY1ftr5E0jqS9ZjYEqgGuB4g//
MxaBaRbJai/6BlcE92NIdBusTtreIm3yTpjIXNAVoeRSnkfuw7wZcL96Qok1Nb1WUuSHw
+psUyeIVtIwFMHEfFoRC0t
+VhmnlnFnkjGPb9W3Aprw2dRRvFM3R2ZTDvMCi0YDzUCd43GftGq2LfxH3qSD51oFHg1HQV0Y0jyVzz1Avub5HQdt0Q
  }
]
}
```

8. Jetzt, da Ihre Token-Datei alle erforderlichen Signaturen enthält, können Sie fortfahren. Geben Sie den Namen der Datei ein, die die signierten Token enthält, und drücken Sie die Eingabetaste. Sie sollten sich jetzt erfolgreich anmelden.

```
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>"
  }
}
```

Schlüssel für Benutzer mit aktiviertem MFA für CloudHSM CLI rotieren

Gehen Sie wie folgt vor, um Schlüssel für AWS CloudHSM Benutzer mit aktivierter Multi-Faktor-Authentifizierung (MFA) zu rotieren.

<result>

Sie haben die generierte Tokendatei im JSON-Format mit Ihrem privaten Schlüssel signiert und einen neuen öffentlichen MFA-Schlüssel registriert.

</result>

1. Verwenden Sie die CloudHSM-CLI, um sich als beliebiger Administrator oder als der spezifische Benutzer, für den MFA aktiviert ist, am HSM anzumelden (weitere Informationen finden Sie unter [Benutzer mit aktiviertem MFA anmelden](#)).
2. Führen Sie als Nächstes den Befehl aus, um Ihre MFA-Strategie zu ändern. Sie müssen den Parameter `--token` angeben. Dieser Parameter gibt eine Datei an, in die unsigned Token geschrieben werden.

```
aws-cloudhsm > user change-mfa token-sign --token unsigned-tokens.json --  
username <username> --role crypto-user --change-quorum  
Enter password:  
Confirm password:
```

3. Identifizieren Sie die Datei mit unsigned Token, die signiert werden müssen: `unsigned-tokens.json`. Die Anzahl der Token in dieser Datei hängt von der Anzahl der Token HSMs in Ihrem Cluster ab. Jedes Token steht für ein HSM. Diese Datei ist im JSON-Format und enthält Token, die signiert werden müssen, um nachzuweisen, dass Sie über einen privaten Schlüssel verfügen. Dies ist der neue private Schlüssel aus dem neuen public/private RSA-Schlüsselpaar, das Sie für die Rotation des aktuell registrierten öffentlichen Schlüssels verwenden möchten.

```
$ cat unsigned-tokens.json  
{  
  "version": "2.0",  
  "tokens": [  
    {  
      "unsigned": "Vtf/9Q0FY45v/E1osvpEMr59JsnP/hLDm4It002vqL8=",  
      "signed": ""  
    },  
    {  
      "unsigned": "wVbC0/5IKwjyZK2NBpdFLyI7BiayZ24YcdUdlcxLwZ4=",  
      "signed": ""  
    },  
    {  
      "unsigned": "z6aW9RzErJBL5KqFG5h8lhTVt9oLbxppjod0Ebysydw=",  
      "signed": ""  
    }  
  ]  
}
```

4. Signieren Sie diese Token mit dem privaten Schlüssel, den Sie zuvor bei der Einrichtung erstellt haben. Zuerst müssen wir die Base64-codierten Token extrahieren und dekodieren.

```
$ echo "Vtf/9Q0FY45v/E1osvpEMr59JsnP/hLDm4It002vqL8=" > 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. Sie haben jetzt binäre Token. Signieren Sie sie mit dem privaten RSA-Schlüssel, den Sie zuvor bei der Installation erstellt haben.

```
$ 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. Sie haben jetzt binäre Signaturen der Token. Kodieren Sie sie mit Base64 und platzieren Sie sie wieder in Ihrer Tokendatei.

```
$ 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. Kopieren Sie schließlich die base64-Werte und fügen Sie sie wieder in Ihre Token-Datei ein:

```
{
  "version": "2.0",
  "tokens": [
    {
```

```

    "unsigned": "1jqwx9bJ0UUQLiNb7mxXS1uBJsEXh0B9nj05BqnPsE=",
    "signed": "eiw3fZeCKIY50C4zPeg9Rt90M1Q1q3W1Jh6Yw7xXm4nF6e9ETLE39+9M
+rUqDWMRZjaBfaMbg5d9yDkz5p13U7ch2t1F9LoYabsWutkT014KRq/rcYMvFsU9n/Ey/
TK0PVaxLN42X+pebV4juwMhN4mK4CzdFAJgM+UGB0j4yB9recp0BB9K8QFSpJZALSEdDgUc/
mS1eDq3rU0int6+4NKuLQjpR
+LSEIWRZ6g6+MND2vXGskxHjadCQ09L7Tz8VcWjKDbxJcBiGKvkqyozl9zrGo8fA3WHBmwiAgS61Merx77ZGY4PFR37
YMSC14prCN15DtMRv2xA1SGSb4w=="
  },
  {
    "unsigned": "LMMFc34ASpNvNPFzBbMbr9FProS/Zu2P8zF/xzk5hVQ=",
    "signed": "HBImKnHmw+6R2TpFEpfiAg4+hu2pFNwn43ClhKPkn2higbEHUD0JVi
+4MerSyvU/NN79iWVxDvJ9Ito+jpiRQjTfTGEoIteyuAr1v/Bzh+Hjmr0530QpZaJ/VXGIgApD0myuu/
ZGNKQTCskkL7+V81FG7yR1Nm22jUeGa735zvm/E+cenvZdy0VVx6A7WeWr13JEKKBweHbi+7BwbaW
+PTdCuIRd4Ug76Sy+cFhsvcG1k7cMwDh8MgXzIZ2m1f/hdy2j8qAx0RTLlmwyU0YvPY0vUhc
+s83hx36QpGwGcD7RA0bPT50rTx7PHd0N1CL+Wwy91We8yIOFBS6nxo1R7w=="
  },
  {
    "unsigned": "dzeHbwhiVXQqcUGj563z51/7sLUdxjL93Sb0UyZRjH8=",
    "signed": "VgQPvrTsvG1jVBFxHnsduq16x8ZrxnxfcYVYGf/
N7gEzI4At3GDs2EVZWTRdvS0uGHdkFYp1apHgJZ7PDVmcGtKIXVD21FYppcgN1SzkY1ftr5E0jqS9ZjYEggGuB4g//
MxaBaRbJai/6B1cE92NidBusTtreIm3yTpjIXNAVoeRSnkfuw7wZcL96Qok1Nb1WUuSHw
+psUyeIVtIwFMHEffoRC0t
+VhmnlnFnkjGPb9W3Aprw2dRRvFM3R2ZTDvMCi0YDzUCd43GftGq2LfxH3qSD51oFHg1HQV0Y0jyVzz1Avub5HQdt00
  }
]
}

```

8. Jetzt, da Ihre Token-Datei alle erforderlichen Signaturen enthält, können Sie fortfahren. Geben Sie den Namen der Datei ein, die die signierten Token enthält, und drücken Sie die Eingabetaste. Geben Sie abschließend den Pfad Ihres neuen öffentlichen Schlüssels ein. Jetzt sehen Sie Folgendes als Teil der Ausgabe der [Benutzerliste](#).

```

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"
  }
}

```

Jetzt haben wir unseren Benutzer mit MFA eingerichtet.

```
{
  "username": "<username>",
  "role": "crypto-user",
  "locked": "false",
  "mfa": [
    {
      "strategy": "token-sign",
      "status": "enabled"
    }
  ],
  "cluster-coverage": "full"
},
```

Einen öffentlichen MFA-Schlüssel mithilfe der CloudHSM-CLI deregistrieren

Gehen Sie wie folgt vor, um einen öffentlichen Schlüssel mit Multi-Faktor-Authentifizierung (MFA) für AWS CloudHSM Admin-Benutzer abzumelden, wenn der öffentliche MFA-Schlüssel registriert ist.

1. Verwenden Sie die CloudHSM CLI, um sich als Administrator mit aktiviertem MFA beim HSM anzumelden.
2. Verwenden Sie den `user change-mfa token-sign`-Befehl, um MFA für einen Benutzer zu entfernen.

```
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"
  }
}
```

Token-Dateireferenz für MFA mit CloudHSM CLI

Die Tokendatei, die entweder bei der Registrierung eines öffentlichen Schlüssels für Multi-Faktor-Authentifizierung (MFA) oder beim Versuch, sich mit MFA bei der CloudHSM-CLI anzumelden, generiert wird, besteht aus folgenden Elementen:

- Tokens: Ein Array mit Base64-codierten unsigned/signed Tokenpaaren in Form von JSON-Objektliteralen.
- Unsigniert: Ein Base64-codiertes und gehashtes Token. SHA256
- Signiert: Ein Base64-codiertes signiertes Token (Signatur) des unsignierten Tokens unter Verwendung des privaten RSA-2048-Bit-Schlüssels.

```
{
  "version": "2.0",
  "tokens": [
    {
      "unsigned": "1jqwx9bJ0UUQLiNb7mxXS1uBJsEXh0B9nj05BqnPsE=",
      "signed": "eiw3fZeCKIY50C4zPeg9Rt90M1Q1q3W1Jh6Yw7xXm4nF6e9ETLE39+9M
+rUqDWMRZjaBfaMbg5d9yDkz5p13U7ch2t1F9LoYabsWutkT014KRq/rcYMvFsU9n/Ey/TK0PVaxLN42X
+pebV4juwMhN4mK4CzdFAJgM+UGB0j4yB9recp0BB9K8QFSpJZALSEdDgUc/mS1eDq3rU0int6+4NKuLQjpR
+LSEIWRZ6g6+MND2vXGskxHjadCQ09L7Tz8VcWjKDbxJcBiGKvkqyozl9zrGo8fA3WHBmwiAgS61Merx77ZGY4PFR37+j/
YMSC14prCN15DtMRv2xA1SGSb4w=="
    },
    {
      "unsigned": "LMMFc34ASpNvNPFzBbMbr9FProS/Zu2P8zF/xzk5hVQ=",
      "signed": "HBImKnHmw+6R2TpFEpfiAg4+hu2pFNwn43ClhKPkn2higbEhUD0JVi
+4MerSyvU/NN79iWVxDvJ9Ito+jpiRQjTfTGEoIteyuAr1v/Bzh+Hjmr0530QpZaJ/VXGIgApD0myuu/
ZGNKQTCskkL7+V81FG7yR1Nm22jUeGa735zvm/E+cenvZdy0VVx6A7WeWrl3JEKKBweHbi+7BwbaW
+PTdCuIRd4Ug76Sy+cFhsvcG1k7cMwDh8MgXzIZ2m1f/hdy2j8qAx0RTLlmyU0YvPY0vUhc
+s83hx36QpGwGcD7RA0bPT50rTx7PHd0N1CL+Wwy91We8yIOFBS6nxo1R7w=="
    },
    {
      "unsigned": "dzeHbwhiVXQqcUGj563z51/7sLUdxjL93Sb0UyZRjH8=",
      "signed": "VgQPvrTsvGljVBFxHnsduq16x8ZrxnxfcYVYGf/
N7gEzI4At3GDs2EVZWTRdvS0uGHdkFYp1apHgJZ7PDVmGcTkIXVD21FYppcgN1SzkY1ftr5E0jqS9ZjYEggGuB4g//
MxaBaRbJai/6BlcE92NIdBusTtreIm3yTpjIXNAVoeRSnkfuw7wZcL96Qok1Nb1WUuSHw
+psUyeIVtIwFMHEfFoRC0t
+Vhmn1nFnkjGPb9W3Aprw2dRRvFM3R2ZTDvMCi0YDzUCd43GftGq2LfxH3qSD51oFHg1HQV0Y0jyVzz1Avub5HQdt0QdErI
    }
  ]
}
```



```
}
```

Verwaltung der Quorum-Authentifizierung (M-of-N-Zugriffskontrolle) mithilfe der CloudHSM-CLI

AWS CloudHSM Cluster unterstützen die Quorumauthentifizierung, auch bekannt als M-of-N-Zugriffskontrolle. Bei dieser Funktion müssen HSM-Benutzer bei bestimmten Vorgängen zusammenarbeiten, wodurch eine zusätzliche Schutzebene entsteht.

Mit der Quorum-Authentifizierung kann kein einzelner Benutzer auf dem HSM quorumgesteuerte Operationen auf dem HSM ausführen. Stattdessen muss eine Mindestanzahl von HSM-Benutzern (mindestens 2) zusammenarbeiten, um diese Operationen durchzuführen.

Die Quorum-Authentifizierung kann die folgenden Vorgänge steuern:

- HSM-Benutzerverwaltung durch den [Administrator](#): HSM-Benutzer erstellen und löschen oder das Passwort eines anderen HSM-Benutzers ändern. Weitere Informationen finden Sie unter [Benutzerverwaltung mit aktivierter Quorum-Authentifizierung für die AWS CloudHSM Verwendung von CloudHSM CLI](#).

Wichtige Punkte zur Quorum-Authentifizierung in AWS CloudHSM

- Ein HSM-Benutzer kann sein eigenes Quorum-Token signieren, d. h. eine der erforderlichen Genehmigungen für die Quorumauthentifizierung bereitstellen.
- Sie wählen die Mindestanzahl von Quorum-Genehmiger, die zwischen zwei (2) und acht (8) liegt.
- HSMs kann bis zu 1024 Quorum-Token speichern. Wenn dieses Limit erreicht ist, löscht das HSM ein abgelaufenes Token, um ein neues zu erstellen.
- Token laufen standardmäßig zehn Minuten nach der Erstellung ab.
- Für Cluster mit aktivierter MFA wird derselbe Schlüssel für die Quorum-Authentifizierung und die Multi-Faktor-Authentifizierung (MFA) verwendet. Weitere Informationen finden Sie [unter Verwenden der CloudHSM-CLI zur Verwaltung von MFA](#).
- Jedes HSM kann ein Token pro Admin-Service und mehrere Token pro Crypto User-Service enthalten.

Die folgenden Themen stellen weitere Informationen zur Quorum-Authentifizierung in AWS CloudHSM zur Verfügung.

Themen

- [Quorum-Authentifizierungsprozess für CloudHSM CLI](#)
- [Unterstützte AWS CloudHSM Dienstnamen und Typen für die Quorum-Authentifizierung mit CloudHSM CLI](#)
- [Richten Sie die Quorum-Authentifizierung für AWS CloudHSM Administratoren mithilfe der CloudHSM-CLI ein](#)
- [Benutzerverwaltung mit aktivierter Quorum-Authentifizierung für die AWS CloudHSM Verwendung von CloudHSM CLI](#)
- [Ändern Sie den Quorum-Mindestwert für die AWS CloudHSM Verwendung von CloudHSM CLI](#)

Quorum-Authentifizierungsprozess für CloudHSM CLI

In den folgenden Schritten werden die Quorum-Authentifizierungsprozesse für CloudHSM CLI zusammengefasst. Die exakten Schritte und Tools finden Sie unter [Benutzerverwaltung mit aktivierter Quorum-Authentifizierung für die AWS CloudHSM Verwendung von CloudHSM CLI](#).

1. Jeder Benutzer des Hardware-Sicherheitsmoduls (HSM) erstellt einen asymmetrischen Schlüssel zum Signieren. Benutzer tun dies außerhalb des HSMs und achten darauf, den Schlüssel angemessen zu schützen.
2. Jeder HSM-Benutzer meldet sich beim HSM an und registriert den öffentlichen Teil seines Signierungsschlüssels (den öffentlichen Schlüssel) beim HSM.
3. Wenn ein HSM-Benutzer eine Quorum-kontrollierte Operation durchführen möchte, meldet sich derselbe Benutzer beim HSM an und erhält ein Quorum-Token.
4. Der HSM-Benutzer gibt das Quorum-Token an einen oder mehrere andere HSM-Benutzer weiter und bittet um deren Zustimmung.
5. Die anderen HSM-Benutzer genehmigen, indem sie mit ihren Schlüsseln das Quorum-Token kryptographisch signieren. Dies geschieht außerhalb des HSMs.
6. Wenn der HSM-Benutzer über die erforderliche Anzahl von Genehmigungen verfügt, meldet sich derselbe Benutzer beim HSM an und führt den quorumgesteuerten Vorgang mit dem `--approval`-Argument aus. Dabei wird die signierte Quorum-Tokendatei bereitgestellt, die alle erforderlichen Genehmigungen (Signaturen) enthält.
7. Das HSM verwendet die registrierten, öffentlichen Schlüssel jedes Unterzeichners, um die Signaturen zu verifizieren. Wenn die Signaturen gültig sind, genehmigt das HSM das Token und der quorumgesteuerte Vorgang wird ausgeführt.

Unterstützte AWS CloudHSM Dienstnamen und Typen für die Quorum-Authentifizierung mit CloudHSM CLI

Admin-Dienste: Die Quorum-Authentifizierung wird für Dienste mit Administratorrechten verwendet, z. B. für das Erstellen von Benutzern, das Löschen von Benutzern, das Ändern von Benutzerkennwörtern, das Festlegen von Quorumwerten und das Deaktivieren von Quorum- und MFA-Funktionen.

Crypto User Services: Die Quorum-Authentifizierung wird für privilegierte Dienste verwendet, die Krypto-Benutzern vorbehalten sind und mit einem bestimmten Schlüssel verknüpft sind, z. B. das Signieren mit einem Schlüssel, sharing/unsharing einem Schlüssel, wrapping/unwrapping einem Schlüssel und das Festlegen eines Schlüsselattributs. Der Quorumwert eines zugehörigen Schlüssels wird konfiguriert, wenn der Schlüssel generiert, importiert oder entpackt wird. Der Quorumwert muss gleich oder kleiner als die Anzahl der Benutzer sein, denen der Schlüssel zugeordnet ist. Dazu gehören Benutzer, mit denen der Schlüssel geteilt wird, und der Schlüsselbesitzer.

Jeder Dienstyp wird weiter in einen qualifizierenden Dienstnamen unterteilt, der eine bestimmte Gruppe von Quorum-unterstützten Dienstvorgängen enthält, die ausgeführt werden können.

Service-Name	Servicetyp	Serviceoperationen
user	Admin.	• user create • user delete • user change-password • user change-mfa
quorum	Admin.	• Quorum-Tokenzeichen set-quorum-value
Cluster 1	Admin.	• Cluster-MTLS register-trust-anchor • Cluster-MTLS deregister-trust-anchor • Durchsetzung von Cluster-M TLS-Sätzen
Schlüsselverwaltung	Crypto-Benutzer	• Schlüsselhülle

Service-Name	Servicetyp	Serviceoperationen
		• Schlüssel auspacken • key share • key unshare • key set-attribute
Verwendung des Schlüssels	Crypto-Benutzer	• Schlüsselzeichen

[1] Der Cluster-Service ist ausschließlich auf hsm2m.medium verfügbar

Richten Sie die Quorum-Authentifizierung für AWS CloudHSM Administratoren mithilfe der CloudHSM-CLI ein

In den folgenden Themen werden die Schritte beschrieben, die Sie ausführen müssen, um Ihr Hardware Security Module (HSM) so zu konfigurieren, dass AWS CloudHSM [Administratoren](#) die Quorumauthentifizierung verwenden können. Sie müssen diese Schritte nur einmal ausführen, wenn Sie die Quorum-Authentifizierung für Admins zum ersten Mal konfigurieren. Nachdem Sie diese Schritte abgeschlossen haben, fahren Sie mit [Benutzerverwaltung mit aktivierter Quorum-Authentifizierung für die AWS CloudHSM Verwendung von CloudHSM CLI](#) fort.

Themen

- [Voraussetzungen](#)
- [Schritt 1. Erstellen und Registrieren eines Schlüssels für das Signieren](#)
- [Schritt 2. Setzen des Quorum-Mindestwerts für das HSM](#)
- [Quorum-Mindestwerte](#)

Voraussetzungen

Um dieses Beispiel zu verstehen, sollten Sie mit der [CloudHSM-CLI](#) vertraut sein.

Schritt 1. Erstellen und Registrieren eines Schlüssels für das Signieren

Um die Quorum-Authentifizierung zu verwenden, muss jeder Administrator alle der folgenden Schritte ausführen:

Themen

- [Erstellen eines RSA-Schlüsselpaars](#)
- [Erstellen und signieren Sie ein Registrierungstoken](#)
- [Registrieren Sie den öffentlichen Schlüssel beim HSM](#)

Erstellen eines RSA-Schlüsselpaars

Es gibt viele verschiedene Möglichkeiten, ein Schlüsselpaar zu erstellen und zu schützen. In den folgenden Beispielen wird gezeigt, wie dies mit [OpenSSL](#) durchgeführt wird.

Example– Erstellen eines privaten Schlüssels mit OpenSSL

Das folgende Beispiel zeigt, wie OpenSSL verwendet wird, um einen 2048-Bit-RSA-Schlüssel zu erstellen. Um dieses Beispiel zu verwenden, `<admin.key>` ersetzen Sie es durch den Namen der Datei, in der Sie den Schlüssel speichern möchten.

```
$ openssl genrsa -out <admin.key>
Generating RSA private key, 2048 bit long modulus
.....+++
.+++
e is 65537 (0x10001)
```

Generieren eines öffentlichen Schlüssels mit dem privaten Schlüssel, den Sie gerade erstellt haben.

Example– Erstellen Sie einen öffentlichen Schlüssel mit OpenSSL

Das folgende Beispiel zeigt, wie Sie OpenSSL verwenden, um einen öffentlichen Schlüssel aus dem privaten Schlüssel zu erstellen, den Sie gerade erstellt haben.

```
$ openssl rsa -in admin.key -outform PEM -pubout -out admin1.pub
writing RSA key
```

Erstellen und signieren Sie ein Registrierungstoken

Sie erstellen ein Token und signieren es mit dem privaten Schlüssel, den Sie gerade im vorherigen Schritt generiert haben.

Example– Erstellen Sie ein Registrierungstoken

1. Verwenden Sie den folgenden Befehl, um die CloudHSM-CLI zu starten:

Linux

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

Windows

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

2. Erstellen Sie ein Registrierungstoken, indem Sie den Befehl [quorum token-sign generate](#) ausführen:

```
aws-cloudhsm > quorum token-sign generate --service registration --token /path/
tokenfile
{
  "error_code": 0,
  "data": {
    "path": "/path/tokenfile"
  }
}
```

3. Der Befehl [quorum token-sign generate](#) generiert ein Registrierungstoken im angegebenen Dateipfad. Untersuchen Sie die Tokendatei:

```
$ cat /path/tokenfile
{
  "version": "2.0",
  "tokens": [
    {
      "approval_data": <approval data in base64 encoding>,
      "unsigned": <unsigned token in base64 encoding>,
      "signed": ""
    }
  ]
}
```

Die Tokendatei besteht aus Folgendem:

- **approval_data:** Ein Base64-kodiertes zufälliges Datentoken, dessen Rohdaten das Maximum von 245 Byte nicht überschreiten.
- **unsigned:** Ein Base64-codiertes und SHA256 gehashtes Token der **approval_data**.

- signiert: Ein base64-kodiertes signiertes Token (Signatur) des unsignierten Tokens, unter Verwendung des zuvor mit OpenSSL generierten privaten RSA-Schlüssels mit 2048 Bit.

Sie signieren das unsignierte Token mit dem privaten Schlüssel, um nachzuweisen, dass Sie Zugriff auf den privaten Schlüssel haben. Sie benötigen die vollständig mit einer Signatur und dem öffentlichen Schlüssel aufgefüllte Registrierungstokendatei, um den Administrator als Quorumbenutzer im Cluster zu registrieren. AWS CloudHSM

Example– Signieren Sie das unsignierte Registrierungstoken

1. Dekodieren Sie das Base64-kodierte unsignierte Token und platzieren Sie es in einer Binärdatei:

```
$ echo -n '6BMUj6mUjjko6ZLCEdzG1WpR5sILhFJfqhW1ej30q1g=' | base64 -d > admin.bin
```

2. Verwenden Sie OpenSSL und den privaten Schlüssel, um das jetzt binäre unsignierte Registrierungstoken zu signieren und eine binäre Signaturdatei zu erstellen:

```
$ openssl pkeyutl -sign \  
-inkey admin.key \  
-pkeyopt digest:sha256 \  
-keyform PEM \  
-in admin.bin \  
-out admin.sig.bin
```

3. Kodieren Sie die binäre Signatur in Base64:

```
$ base64 -w0 admin.sig.bin > admin.sig.b64
```

4. Kopieren Sie die Base64-kodierte Signatur und fügen Sie sie in die Token-Datei ein:

```
{  
  "version": "2.0",  
  "tokens": [  
    {  
      "approval_data": <approval data in base64 encoding>,  
      "unsigned": <unsigned token in base64 encoding>,  
      "signed": <signed token in base64 encoding>  
    }  
  ]  
}
```

```
}
```

Registrieren Sie den öffentlichen Schlüssel beim HSM

Nach der Erstellung eines Schlüssels muss der Administrator den öffentlichen Schlüssel im AWS CloudHSM Cluster registrieren.

So registrieren Sie einen öffentlichen Schlüssel bei dem HSM

1. Verwenden Sie den folgenden Befehl, um die CloudHSM-CLI zu starten:

Linux

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

Windows

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

2. Melden Sie sich mit der CloudHSM-CLI als Administrator an.

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

3. Verwenden Sie den Befehl [Registrieren Sie die Token-Sign-Quorum-Strategie eines Benutzers mithilfe der CloudHSM-CLI](#), um den öffentlichen Schlüssel zu registrieren. Weitere Informationen finden Sie im folgenden Beispiel. Alternativ können Sie auch den Befehl `help user change-quorum token-sign register` ausführen.

Example— Registrieren Sie einen öffentlichen Schlüssel im AWS CloudHSM Cluster

Das folgende Beispiel zeigt, wie Sie den `user change-quorum token-sign register`-Befehl in CloudHSM CLI verwenden, um einen öffentlichen Schlüssel von admin im HSM zu registrieren. Der

Admin muss bei dem HSM angemeldet sein, um diesen Befehl zu verwenden. Ersetzen Sie diese Werte durch Ihre eigenen Werte:

```
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: Der Dateipfad zur PEM-Datei mit dem öffentlichen Schlüssel

Erforderlich: Ja

/path/tokenfile: Der Dateipfad mit dem Token, das mit dem privaten Schlüssel des Benutzers signiert wurde

Erforderlich: Ja

Nachdem alle Admins ihre öffentlichen Schlüssel registriert haben, zeigt die Ausgabe des user list-Befehls dies im Quorumfeld an und gibt an, welche aktivierte Quorum-Strategie verwendet wird, wie unten dargestellt:

```
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"
  }
]
}
}

```

In diesem Beispiel besteht der AWS CloudHSM Cluster aus zwei HSMs Clustern mit jeweils denselben Admins, wie in der folgenden Ausgabe des `user list` Befehls dargestellt. Weitere Informationen zum Erstellen von Benutzern finden Sie unter [Benutzerverwaltung mit CloudHSM CLI](#)

Schritt 2. Setzen des Quorum-Mindestwerts für das HSM

Um die Quorum-Authentifizierung zu verwenden, muss sich ein Administrator beim HSM anmelden und dann den Mindestwert für das Quorum festlegen. Dies ist die Mindestanzahl von Admin-Genehmigungen, die erforderlich sind, um HSM-Benutzermanagement-Vorgänge auszuführen. Jeder Administrator auf dem HSM kann den Mindestwert für das Quorum festlegen, auch Administratoren, die keinen Schlüssel zum Signieren registriert haben. Sie können den Quorum-Mindestwert jederzeit ändern. Weitere Informationen finden Sie unter [Ändern Sie den Mindestwert](#).

So legen Sie den Quorum-Mindestwert im HSM fest

1. Verwenden Sie den folgenden Befehl, um die CloudHSM-CLI zu starten:

Linux

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

Windows

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

2. Melden Sie sich mit der CloudHSM-CLI als Administrator an.

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

```

```

    "data": {
      "username": "<admin>",
      "role": "admin"
    }
  }
}

```

3. Verwenden Sie den [Einen Quorumwert mithilfe der CloudHSM-CLI aktualisieren](#)-Befehl, um den Quorum-Mindestwert festzulegen. Das `--service` Flag identifiziert den HSM-Service, für den Sie Werte festlegen. Sehen Sie sich das folgende Beispiel an oder verwenden Sie den `help quorum token-sign set-quorum-value` Befehl für weitere Informationen.

Example– Setzen des Quorum-Mindestwerts für das HSM

In diesem Beispiel wird ein Quorum-Mindestwert von zwei (2) verwendet. Sie können einen Wert zwischen zwei (2) und acht (8) wählen, bis zur Gesamtzahl der Administratoren auf dem HSM. In diesem Beispiel hat das HSM vier (4) Administratoren, sodass der maximal mögliche Wert vier (4) ist.

Um den folgenden Beispielbefehl zu verwenden, ersetzen Sie die letzte Zahl (<2>) durch den bevorzugten Quorum-Mindestwert.

```

aws-cloudhsm > quorum token-sign set-quorum-value --service user --value <2>
{
  "error_code": 0,
  "data": "Set quorum value successful"
}

```

In diesem Beispiel listet der [Quorumwerte mit CloudHSM CLI anzeigen](#) Befehl die HSM-Diensttypen, -namen und -beschreibungen auf, die im Dienst enthalten sind.

Quorum-Mindestwerte

Um den Quorum-Mindestwert für einen Dienst abzurufen, verwenden Sie den `quorum token-sign list-quorum-values`-Befehl:

```

aws-cloudhsm > quorum token-sign list-quorum-values
{
  "error_code": 0,
  "data": {
    "user": 2,
    "quorum": 1
  }
}

```

```
}
}
```

Die Ausgabe des vorangegangenen `quorum token-sign list-quorum-values`-Befehls zeigt, dass der Quorum-Mindestwert für den HSM-Benutzerdienst, der für die Benutzerverwaltungsvorgänge zuständig ist, jetzt zwei (2) beträgt. Nachdem Sie diese Schritte abgeschlossen haben, fahren Sie mit [Benutzerverwaltung mit Quorum \(M von N\)](#) fort.

Admin-Dienste: Die Quorum-Authentifizierung wird für Dienste mit Administratorrechten verwendet, z. B. für das Erstellen von Benutzern, das Löschen von Benutzern, das Ändern von Benutzerkennwörtern, das Festlegen von Quorumwerten und das Deaktivieren von Quorum- und MFA-Funktionen.

Crypto User Services: Die Quorum-Authentifizierung wird für privilegierte Dienste verwendet, die Krypto-Benutzern vorbehalten sind und mit einem bestimmten Schlüssel verknüpft sind, z. B. das Signieren mit einem Schlüssel, sharing/unsharing einem Schlüssel, wrapping/unwrapping einem Schlüssel und das Festlegen eines Schlüsselattributs. Der Quorumwert eines zugehörigen Schlüssels wird konfiguriert, wenn der Schlüssel generiert, importiert oder entpackt wird. Der Quorumwert muss gleich oder kleiner als die Anzahl der Benutzer sein, denen der Schlüssel zugeordnet ist. Dazu gehören Benutzer, mit denen der Schlüssel geteilt wird, und der Schlüsselbesitzer.

Jeder Dienstyp wird weiter in einen qualifizierenden Dienstnamen unterteilt, der eine bestimmte Gruppe von Quorum-unterstützten Dienstvorgängen enthält, die ausgeführt werden können.

Service-Name	Servicetyp	Serviceoperationen
user	Admin.	• user create • user delete • user change-password • user change-mfa
quorum	Admin.	• Quorum-Tokenzeichen set-quorum-value
Cluster 1	Admin.	• Cluster-MTLS register-trust-anchor • Cluster-MTLS deregister-trust-anchor

Service-Name	Servicetyp	Serviceoperationen
		• Durchsetzung von Cluster-M TLS-Sätzen
Schlüsselverwaltung	Crypto-Benutzer	• Schlüsselhülle • Schlüssel auspacken • key share • key unshare • key set-attribute
Verwendung des Schlüssels	Crypto-Benutzer	• Schlüsselzeichen

[1] Der Cluster-Service ist ausschließlich auf hsm2m.medium verfügbar

Benutzerverwaltung mit aktivierter Quorum-Authentifizierung für die AWS CloudHSM Verwendung von CloudHSM CLI

Ein AWS CloudHSM [Administrator](#) des Hardware Security Module (HSM) kann die Quorumauthentifizierung für die folgenden Vorgänge im Cluster konfigurieren: AWS CloudHSM

- [Einen AWS CloudHSM Benutzer mit CloudHSM CLI erstellen](#)
- [Löschen Sie einen AWS CloudHSM Benutzer mit CloudHSM CLI](#)
- [Ändern Sie das Passwort eines Benutzers mit CloudHSM CLI](#)
- [Die Kategorie „user change-mfa“ in der CloudHSM-CLI](#)

Nachdem der AWS CloudHSM Cluster für die Quorumauthentifizierung konfiguriert wurde, können Administratoren HSM-Benutzerverwaltungsvorgänge nicht mehr selbst ausführen. Das folgende Beispiel zeigt die Ausgabe, die angezeigt wird, wenn ein Admin versucht, einen neuen Benutzer auf dem HSM anzulegen. Der Befehl schlägt fehl und es wird ein Fehler angezeigt, der besagt, dass eine Quorumauthentifizierung erforderlich ist.

```
aws-cloudhsm > user create --username user1 --role crypto-user
Enter password:
Confirm password:
{
  "error_code": 1,
```

```
"data": "Quorum approval is required for this operation"
}
```

Zum Ausführen eines HSM-Benutzermanagement-Vorgangs muss ein Admin folgende Aufgaben erledigen:

Themen

- [Schritt 1. Abrufen eines Quorum-Tokens](#)
- [Schritt 2. Unterschriften von genehmigenden Administratoren einholen](#)
- [Schritt 3. Genehmigen Sie das Token auf dem AWS CloudHSM Cluster und führen Sie einen Benutzerverwaltungsvorgang aus](#)

Schritt 1. Abrufen eines Quorum-Tokens

Zunächst muss der Administrator die CloudHSM-CLI verwenden, um ein Quorum-Token anzufordern.

So fordern Sie ein Quorum-Token an

1. Verwenden Sie den folgenden Befehl, um die CloudHSM-CLI zu starten.

Linux

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

Windows

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

2. Melden Sie sich mit der CloudHSM-CLI als Administrator an.

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

3. Verwenden Sie zum Generieren eines Quorum-Tokens den Befehl `quorum token-sign generate`. Weitere Informationen finden Sie im folgenden Beispiel. Alternativ können Sie auch den Befehl `help quorum token-sign generate` ausführen.

Example– Generieren Sie ein Quorum-Token

Dieses Beispiel ruft ein Quorum-Token für den Admin mit Benutzernamen `admin` ab und speichert das Token in einer Datei namens `admin.token`. Zum Verwenden des Beispiels ersetzen Sie diese Werte durch Ihre eigenen:

- **`<admin>`**— Der Name des Administrators, der das Token erhält. Dies muss derselbe Admin sein, der am HSM angemeldet ist und diesen Befehl ausführt.
- **`<admin.token>`**— Der Name der Datei, die zum Speichern des Quorum-Tokens verwendet werden soll.

In dem folgenden Befehl gibt `user` den Namen des Dienstes an, für den Sie das Token, das Sie erzeugen, verwenden können. In diesem Fall ist das Token für die HSM-Benutzermanagement-Vorgänge (Service user).

```
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"
  }
}
```

Der `quorum token-sign generate`-Befehl generiert ein Quorum-Token für den Benutzerdienst im angegebenen Dateipfad. Die Tokendatei kann wie folgt überprüft werden:


```
$ cat </path/admin.token>
{
  "version": "2.0",
  "service": "user-management",
  "approval_data": "AAEAAwAAABgAAAAAAAAAAAJ9eFkfcP3mNzJA1fK
+0WbNhZG1pbgAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAABj5vbeAAAAAAAAAAAAAQADAAAFQAAAAAAAAAAW/
v5Euk83amq1fij0zyvD2FkbWluAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAGPm9t4AAAAAAAAAAAAABAAMAAAAUA
+b23gAAAAAAAAAA",
  "token": "0l2LZkmAHZyAc1hPhyck0oVW33aGrgG77qmDHWQ3CJ8=",
  "signatures": []
}
```

Die Tokendatei besteht aus Folgendem:

- **service:** Eine Kennung für den Quorumdienst, dem das Token zugeordnet ist.
- **approval_data:** Ein Base64-codiertes Rohdaten-Token, das vom HSM generiert wurde.
- **token:** Ein Base64-kodiertes und SHA-256-Hash-Token der **approval_data**
- **signatures:** Ein Array von Base64-codierten signierten Tokens (Signaturen) des unsignierten Tokens, wobei jede Signatur eines Genehmigers die Form eines JSON-Objektliterals hat:

```
{
  "username": "<APPROVER_USERNAME>",
  "role": "<APPROVER_ROLE>",
  "signature": "<APPROVER_RSA2048_BIT_SIGNATURE>"
}
```

Jede Signatur wird aus dem Ergebnis eines Genehmigers erstellt, der den entsprechenden privaten RSA-2048-Bit-Schlüssel verwendet, dessen öffentlicher Schlüssel im HSM registriert wurde.

Durch Ausführen des `quorum token-sign list`-Befehls kann bestätigt werden, dass das generierte Benutzerdienst-Quorum-Token auf dem CloudHSM-Cluster vorhanden ist:

```
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"
  }
]
}
}

```

Die `token-timeout-seconds`-Zeit gibt den Zeitraum in Sekunden an, in dem ein generiertes Token genehmigt werden muss, bevor es abläuft.

Schritt 2. Unterschriften von genehmigenden Administratoren einholen

Ein Administrator, der ein Quorum-Token besitzt, muss das Token von anderen Admins genehmigen lassen. Für die Genehmigung nutzen die anderen Admins Ihren Signaturschlüssel zur kryptografischen Token-Signierung. Dies geschieht außerhalb des HSMs.

Es gibt viele verschiedene Möglichkeiten, ein Token zu signieren. Im folgenden Beispiel wird gezeigt, wie dies mit [OpenSSL](#) durchgeführt wird. Wenn Sie ein anderes Signatur-Tool verwenden, müssen Sie sicherstellen, dass dieses den privaten Schlüssel (Signaturschlüssel) des Admins zur Signierung des SHA-256-Digests des Tokens verwendet.

Example– Unterschriften von genehmigenden Administratoren einholen

In diesem Beispiel benötigt der Administrator, der das Token (`admin`) besitzt, mindestens zwei (2) Genehmigungen. Das folgende Beispiel zeigt, wie zwei (2) Admins OpenSSL zur kryptografischen Signierung des Tokens einsetzen können.

1. Dekodieren Sie das Base64-kodierte unsignierte Token und platzieren Sie es in einer Binärdatei:

```
$ echo -n '012LZkmAHZyAc1hPhyck0oVW33aGrgG77qmDHWQ3CJ8=' | base64 -d > admin.bin
```

- Verwenden Sie OpenSSL und den jeweiligen privaten Schlüssel des Genehmigers (admin3), um das jetzt binäre unsignierte Quorum-Token für den Benutzerservice zu signieren und eine binäre Signaturdatei zu erstellen:

```
$ openssl pkeyutl -sign \
-inkey admin3.key \
-pkeyopt digest:sha256 \
-keyform PEM \
-in admin.bin \
-out admin.sig.bin
```

- Kodieren Sie die binäre Signatur in Base64:

```
$ base64 -w0 admin.sig.bin > admin.sig.b64
```

- Kopieren Sie abschließend die Base64-kodierte Signatur und fügen Sie sie in die Tokendatei ein. Folgen Sie dabei dem zuvor für die Genehmigersignatur angegebenen JSON-Objektliteralformat:

```
{
  "version": "2.0",
  "approval_data": "AAEAAwAAABgAAAAAAAAAAJ9eFkfcP3mNzJA1fK
+0WbNhZG1pbgAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAABj5vbeAAAAAAAAAAAAAQADAAAFQAAAAAAAAAAAAW
v5Euk83amq1fij0zyvD2FkbWluAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAGPm9t4AAAAAAAAAAAAABAAMAA
+b23gAAAAAAAAAA",
  "token": "012LZkmAHZyAc1hPhyck0oVW33aGrgG77qmDHWQ3CJ8=",
  "signatures": [
    {
      "username": "admin2",
      "role": "admin",
      "signature": "06qx7/mUaVkYYVr1PW7l8JJko+Kh3e8zBIqdk3tAiNy+1rW
+0sDtvYujhEU4a0FVLcrUFmyB/CX90QmgJLgx/pyK+ZPEH+GoJGqk9YZ7X1n0XwZRP9g7hKV
+7XCtg9TuDFtHYWDpBfz2jWiu2fXfX4/
jTs4f2xIfFPIDKcSP8fhxjQ63xEcCf1jzGha6rDQMu4xUWwdtDgft7um7EJ9dXNoHqLB7cTzphaubNaEFbFPXQ1siGm
ssktwyruGFLpXs1n0tJ0EglGhx2qbYTs+omKWZd0R15WIWEXW3IXw/
Dg5vV0brNpvG0eZK08nSMc27+cyPySc+ZbNw=="
    },
    {
      "username": "admin3",
      "role": "admin",
      "signature": "06qx7/mUaVkYYVr1PW7l8JJko+Kh3e8zBIqdk3tAiNy+1rW
+0sDtvYujhEU4a0FVLcrUFmyB/CX90QmgJLgx/pyK+ZPEH+GoJGqk9YZ7X1n0XwZRP9g7hKV
+7XCtg9TuDFtHYWDpBfz2jWiu2fXfX4/"
    }
  ]
}
```

```
jTs4f2xIfFPIDKcSP8fhxjQ63xEcCf1jzGha6rDQMu4xUWwdtDgFT7um7EJ9dXNoHqLB7cTzphaubNaEFbFPXQ1siGm
ssktwyruGFLpXs1n0tJ0EglGhx2qbYTs+omKWZd0R15WIWEXW3IXw/
Dg5vV0brNpvG0eZK08nSMc27+cyPySc+ZbNw=="
    }
  ]
}
```

Schritt 3. Genehmigen Sie das Token auf dem AWS CloudHSM Cluster und führen Sie einen Benutzerverwaltungsvorgang aus

Sobald ein Administrator über die erforderlichen Genehmigungen/Signaturen verfügt hat, wie im vorherigen Abschnitt beschrieben, kann er dieses Token zusammen mit einem der folgenden Benutzerverwaltungsvorgänge an den AWS CloudHSM -Cluster weitergeben:

- [Erstellen](#)
- [delete](#)
- [change-password](#)
- [user change-mfa](#)

Weitere Informationen zur Verwendung dieser Befehle finden Sie unter [Benutzerverwaltung mit CloudHSM CLI](#).

Während der Transaktion wird das Token innerhalb des AWS CloudHSM Clusters genehmigt und der angeforderte Benutzerverwaltungsvorgang ausgeführt. Der Erfolg des Benutzerverwaltungsvorgangs hängt sowohl von einem gültigen genehmigten Quorum-Token als auch von einem gültigen Benutzerverwaltungsvorgang ab.

Der Admin kann das Token nur für einen Vorgang nutzen. Wurde dieser erfolgreich ausgeführt, verliert das Token seine Gültigkeit. Um einen weiteren HSM-Benutzerverwaltungsvorgang durchzuführen, muss der Administrator den oben beschriebenen Vorgang wiederholen. Das heißt, der Administrator muss ein neues Quorum-Token generieren, neue Signaturen von den Genehmigenden einholen und dann das neue Token auf dem HSM mit dem angeforderten Benutzerverwaltungsvorgang genehmigen und verbrauchen.

Note

Das Quorum-Token ist nur gültig, solange Ihre aktuelle Anmeldesitzung geöffnet ist. Wenn Sie sich von CloudHSM-CLI abmelden oder wenn die Netzwerkverbindung unterbrochen

wird, ist das Token nicht mehr gültig. Ebenso kann ein autorisiertes Token nur innerhalb der CloudHSM-CLI verwendet werden. Es kann nicht zur Authentifizierung in einer anderen Anwendung verwendet werden.

Example Einen neuen Benutzer als Administrator erstellen

Im folgenden Beispiel erstellt ein eingeloggter Administrator einen neuen Benutzer auf dem HSM:

```
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"  
  }  
}
```

Der Administrator gibt dann den user list-Befehl ein, um die Erstellung des neuen Benutzers zu bestätigen:

```
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"
  }
]
}
}

```

Wenn der Administrator versucht, einen anderen HSM-Benutzerverwaltungsvorgang durchzuführen, schlägt dieser mit einem Quorum-Authentifizierungsfehler fehl:

```

aws-cloudhsm > user delete --username user1 --role crypto-user
{
  "error_code": 1,
  "data": "Quorum approval is required for this operation"
}

```

Wie unten gezeigt, zeigt der quorum token-sign list-Befehl, dass der Administrator keine genehmigten Token hat. Um eine weitere HSM-Benutzerverwaltungsoperation durchzuführen, muss der Administrator ein neues Quorum-Token generieren, neue Signaturen von den Genehmigern einholen und die gewünschte Benutzerverwaltungsoperation mit dem Argument `--approval` ausführen, um das Quorum-Token bereitzustellen, das bei der Ausführung der Benutzerverwaltungsoperation genehmigt und aufgenommen werden soll.

```

aws-cloudhsm > quorum token-sign list
{
  "error_code": 0,
  "data": {
    "tokens": []
  }
}

```

Ändern Sie den Quorum-Mindestwert für die AWS CloudHSM Verwendung von CloudHSM CLI

Nachdem Sie [den Quorum-Mindestwert für CloudHSM-Administratoren festgelegt](#) haben, müssen Sie möglicherweise den Quorum-Mindestwert anpassen. Das HSM lässt Änderungen am Quorum-Mindestwert nur zu, wenn die Anzahl der Genehmiger den aktuellen Wert erreicht oder überschreitet. Bei einem Quorum-Mindestwert von zwei (2) müssen beispielsweise mindestens zwei (2) Administratoren alle Änderungen genehmigen.

Note

Der Quorumwert des Benutzerdienstes muss immer kleiner oder gleich dem Quorumwert des Quorumdienstes sein. Informationen zu Dienstnamen finden Sie unter. [Unterstützte AWS CloudHSM Dienstnamen und Typen für die Quorum-Authentifizierung mit CloudHSM CLI](#)

Um die Zustimmung des Quorums zur Änderung des Quorum-Mindestwerts zu erhalten, benötigen Sie ein Quorum-Token für den quorum service, der den quorum token-sign set-quorum-value-Befehl verwendet. Um ein Quorum-Token für den quorum service zu generieren, der den quorum token-sign set-quorum-value-Befehl verwendet, muss der Quorumdienst höher als eins (1) sein. Das heißt, bevor Sie den Quorum-Mindestwert für den Benutzerdienst ändern können, müssen Sie möglicherweise den Quorum-Mindestwert für den Quorum-Dienst ändern.

Schritte zum Ändern des Quorum-Mindestwerts für Administratoren

1. Starten Sie den interaktiven CloudHSM-CLI-Modus.

Linux

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

Windows

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

2. Melden Sie sich mit der CloudHSM-CLI als Administrator an.

```
aws-cloudhsm > login --username <admin> --role admin
Enter password:
```



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

- Überprüfen Sie die aktuellen Quorum-Mindestwerte:

```
aws-cloudhsm > quorum token-sign list-quorum-values
```

- Wenn der Quorum-Mindestwert für den Quorumdienst niedriger ist als der Wert für den Benutzerdienst, ändern Sie den Quorumdienstwert:

```
aws-cloudhsm > quorum token-sign set-quorum-value --service quorum --value <3>
```

- [Generieren Sie ein Quorum-Token](#) für den Quorumdienst.
- [Erhalten von Genehmigungen \(Signaturen\) von anderen Admins.](#)
- [Genehmigen Sie das Token auf dem CloudHSM-Cluster und führen Sie einen Benutzerverwaltungsvorgang aus.](#)
- Ändern Sie den Quorum-Mindestwert für den Benutzerdienst:

```
aws-cloudhsm > quorum token-sign set-quorum-value
```

Example Mindestwerte für den Quorumdienst anpassen

- Überprüfen Sie die aktuellen Werte. Das Beispiel zeigt, dass der Quorum-Mindestwert für den Benutzerservice derzeit zwei (2) beträgt.

```
aws-cloudhsm > quorum token-sign list-quorum-values
{
  "error_code": 0,
  "data": {
    "user": 2,
    "quorum": 1
  }
}
```

2. Ändern Sie den Quorumdienstwert. Legen Sie den Quorum-Mindestwert für den Quorumdienst auf einen Wert fest, der dem Wert für den Benutzerdienst entspricht oder höher ist. In diesem Beispiel wird der Quorum-Mindestwert für den Quorumdienst auf zwei (2) festgelegt. Dies ist derselbe Wert, der im vorherigen Beispiel für den Benutzerservice festgelegt wurde.

```
aws-cloudhsm > quorum token-sign set-quorum-value --service quorum --value 2
{
  "error_code": 0,
  "data": "Set quorum value successful"
}
```

3. Überprüfen Sie die Änderungen. Dieses Beispiel zeigt, dass der Quorum-Mindestwert für Benutzerservice und Quorumdienst jetzt zwei (2) beträgt.

```
aws-cloudhsm > quorum token-sign list-quorum-values
{
  "error_code": 0,
  "data": {
    "user": 2,
    "quorum": 2
  }
}
```

HSM-Benutzerverwaltung mit CloudHSM Management Utility (CMU)

[Um Benutzer des Hardware Security Module \(HSM\) zu verwalten AWS CloudHSM, müssen Sie sich mit dem Benutzernamen und dem Passwort eines Cryptographic Officer \(CO\) beim HSM anmelden.](#)

COs können nur Benutzer verwalten. Das HSM enthält einen Standard-CO mit dem Namen admin. Sie haben das Passwort für admin festgelegt, wenn Sie das [Cluster aktiviert haben](#).

Dieses Thema enthält step-by-step Anleitungen und Einzelheiten zur Verwaltung von HSM-Benutzern mit dem AWS CloudHSM Management Utility (CMU).

Themen

- [Voraussetzungen für die Benutzerverwaltung in Management Utility AWS CloudHSM](#)
- [HSM-Benutzertypen für Management Utility AWS CloudHSM](#)
- [Tabelle mit HSM-Benutzerberechtigungen für Management Utility AWS CloudHSM](#)

- [Erstellen Sie HSM-Benutzer mit dem Management Utility AWS CloudHSM](#)
- [Führen Sie mithilfe AWS CloudHSM des Management Utility alle HSM-Benutzer im Cluster auf](#)
- [Ändern Sie HSM-Benutzerkennwörter mit AWS CloudHSM der Management Utility](#)
- [Löschen Sie HSM-Benutzer mithilfe des Management Utility AWS CloudHSM](#)
- [2FA für Benutzer mit dem AWS CloudHSM Management Utility verwalten](#)
- [Verwendung von CloudHSM Management Utility \(CMU\) zur Verwaltung der Quorum-Authentifizierung \(M-von-N-Zugriffskontrolle\)](#)

Voraussetzungen für die Benutzerverwaltung in Management Utility AWS CloudHSM

Bevor Sie das AWS CloudHSM Management Utility (CMU) zur Verwaltung von Benutzern des Hardware Security Module (HSM) verwenden können AWS CloudHSM, müssen Sie diese Voraussetzungen erfüllen. In den folgenden Themen werden die ersten Schritte mit der CMU beschrieben.

Sections

- [Rufen Sie die IP-Adresse eines HSM ab in AWS CloudHSM](#)
- [Verwenden von CMU mit Client-SDK 3.2.1 und früher](#)
- [Laden Sie das CloudHSM Management Utility herunter](#)

Rufen Sie die IP-Adresse eines HSM ab in AWS CloudHSM

Um CMU verwenden zu können, müssen Sie das Configure-Tool verwenden, um die lokale Konfiguration zu aktualisieren. Die CMU stellt eine eigene Verbindung zum Cluster her, und diese Verbindung ist nicht clusterfähig. Um Clusterinformationen nachzuverfolgen, verwaltet die CMU eine lokale Konfigurationsdatei. Das bedeutet, dass Sie bei jeder Verwendung der CMU zunächst die Konfigurationsdatei aktualisieren sollten, indem Sie das Befehlszeilentool [configure](#) mit dem `--cmu-` Parameter ausführen. Wenn Sie das Client-SDK 3.2.1 oder früher verwenden, müssen Sie einen anderen Parameter als `--cmu` verwenden. Weitere Informationen finden Sie unter [the section called "Verwenden von CMU mit Client-SDK 3.2.1 und früher"](#).

Für den `--cmu`-Parameter müssen Sie die IP-Adresse eines HSM in Ihrem Cluster hinzufügen. Wenn Sie mehrere haben HSMs, können Sie eine beliebige IP-Adresse verwenden. Dadurch wird sichergestellt, dass die CMU alle von Ihnen vorgenommenen Änderungen auf den gesamten

Cluster übertragen kann. Denken Sie daran, dass die CMU ihre lokale Datei verwendet, um Clusterinformationen zu verfolgen. Wenn sich der Cluster seit der letzten Verwendung der CMU von einem bestimmten Host aus geändert hat, müssen Sie diese Änderungen der lokalen Konfigurationsdatei hinzufügen, die auf diesem Host gespeichert ist. Fügen Sie niemals ein HSM hinzu oder entfernen Sie es, während Sie CMU verwenden.

Um eine IP-Adresse für ein HSM (Konsole) zu erhalten

1. Öffnen Sie die AWS CloudHSM Konsole zu <https://console.aws.amazon.com/cloudhsm/Hause>.
2. Um die AWS-Region zu ändern, verwenden Sie die Regionsauswahl in der oberen rechten Ecke der Seite.
3. Um die Cluster-Detailseite zu öffnen, wählen Sie in der Cluster-Tabelle die Cluster-ID aus.
4. Um die IP-Adresse zu erhalten, gehen Sie zur HSMs Registerkarte. Wählen Sie für IPv4 Cluster eine Adresse aus, die unter IPv4 ENI-Adresse aufgeführt ist. Verwenden Sie für Dual-Stack-Cluster entweder die ENI IPv4 - oder die IPv6 ENI-Adresse.

Um eine IP-Adresse für ein HSM zu erhalten (AWS CLI)

- Rufen Sie die IP-Adresse eines HSM ab, indem Sie den [describe-clusters](#) Befehl von der AWS CLI verwenden. In der Ausgabe des Befehls entspricht die IP-Adresse von den Werten von `EniIp` und `EniIPv6` (falls es HSMs sich um einen Dual-Stack-Cluster handelt).

```
$ 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",
...
      }
    ]
  }
}
```

Verwenden von CMU mit Client-SDK 3.2.1 und früher

Mit Client SDK 3.3.0 AWS CloudHSM wurde Unterstützung für den `--cmu` Parameter hinzugefügt, was die Aktualisierung der Konfigurationsdatei für die CMU vereinfacht. Wenn Sie eine CMU-Version von Client-SDK 3.2.1 oder früher verwenden, müssen Sie weiterhin die Parameter `-a` und `-m` verwenden, um die Konfigurationsdatei zu aktualisieren. Weitere Informationen zu diesen Parametern finden Sie unter [Configure-Tool](#).

Laden Sie das CloudHSM Management Utility herunter

Die neueste Version von CMU ist für HSM-Benutzerverwaltungsaufgaben verfügbar, unabhängig davon, ob Sie Client-SDK 5 und Client-SDK 3 verwenden.

So laden Sie CMU herunter und installieren es

- Laden Sie jq herunter und installieren Sie sie.

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. Laden Sie das [CloudHSM Management Utility](#) herunter
2. Führen Sie das CMU-Installationsprogramm (AWSCloudHSMMManagementUtil-latest.msi) mit Windows-Administratorrechten aus.

Windows Server 2012 R2

1. Laden Sie das [CloudHSM Management Utility](#) herunter
2. Führen Sie das CMU-Installationsprogramm (AWSCloudHSMMManagementUtil-latest.msi) mit Windows-Administratorrechten aus.

Windows Server 2016

1. Laden Sie das [CloudHSM Management Utility](#) herunter
2. Führen Sie das CMU-Installationsprogramm (AWSCloudHSMMManagementUtil-latest.msi) mit Windows-Administratorrechten aus.

HSM-Benutzertypen für Management Utility AWS CloudHSM

Für die meisten Operationen, die Sie auf dem Hardware Security Module (HSM) ausführen, sind die Anmeldeinformationen eines AWS CloudHSM HSM-Benutzers erforderlich. Das HSM authentifiziert jeden HSM-Benutzer, und jeder HSM-Benutzer hat einen Typ, der bestimmt, welche Operationen Sie als dieser Benutzer auf dem HSM ausführen können.

Note

HSM-Benutzer unterscheiden sich von IAM-Benutzern. IAM-Benutzer, die über die richtigen Anmeldeinformationen verfügen, können HSMs durch Interaktion mit Ressourcen über die AWS-API etwas erstellen. Nachdem das HSM erstellt wurde, müssen Sie HSM-Benutzeranmeldedaten verwenden, um Vorgänge auf dem HSM zu authentifizieren.

Benutzertypen

- [Precrypto Officer \(PRECO\)](#)
- [Crypto Officer \(CO\)](#)

- [Crypto-Benutzer \(Crypto User, CU\)](#)
- [Appliance-Benutzer \(Appliance User, AU\)](#)

Precrypto Officer (PRECO)

Sowohl im Cloud Management Utility (CMU) als auch im Key Management Utility (KMU) ist PRECO ein temporärer Benutzer, der nur auf dem ersten HSM in einem AWS CloudHSM -Cluster existiert. Das erste HSM in einem neuen Cluster enthält einen PRECO-Benutzer, der angibt, dass dieser Cluster noch nie aktiviert wurde. Um [einen Cluster zu aktivieren](#), führen Sie die cloudhsm-cli aus und führen den Befehl aus. cluster activate Melden Sie sich beim HSM an und ändern Sie das PRECO-Passwort. Wenn Sie das Passwort ändern, wird dieser Benutzer zum Crypto Officer (CO).

Crypto Officer (CO)

Sowohl im Cloud Management Utility (CMU) als auch im Key Management Utility (KMU) kann ein Crypto Officer (CO) Benutzerverwaltungsvorgänge durchführen. Beispielsweise können Sie Benutzer erstellen und löschen und Benutzerpasswörter ändern. Weitere Informationen über CO-Benutzer finden Sie unter [Tabelle mit HSM-Benutzerberechtigungen für Management Utility AWS CloudHSM](#). Wenn Sie einen neuen Cluster aktivieren, wechselt der Benutzer von einem [Precrypto Officer \(PRECO\)](#) zu einem Crypto Officer (CO).

Crypto-Benutzer (Crypto User, CU)

Ein Crypto-Benutzer (CU) kann die folgenden Schlüsselverwaltungs- und kryptografischen Vorgänge ausführen.

- Schlüsselverwaltung – Erstellen, Löschen, Freigeben, Importieren und Exportieren von kryptographischen Schlüsseln.
- Kryptografische Operationen – Verwenden Sie kryptographische Schlüssel für die Verschlüsselung, Entschlüsselung, Signatur, Verifizierung und mehr.

Weitere Informationen hierzu finden Sie unter [Tabelle mit HSM-Benutzerberechtigungen für Management Utility AWS CloudHSM](#).

Appliance-Benutzer (Appliance User, AU)

Der Appliance-Benutzer (AU) kann Kloning- und Synchronisierungsvorgänge auf Ihrem Cluster durchführen. HSMs AWS CloudHSM verwendet die AU, um die HSMs in einem AWS CloudHSM

Cluster zu synchronisieren. Die AU ist auf allen Geräten vorhanden, die von HSMs bereitgestellt werden AWS CloudHSM, und verfügt über eingeschränkte Berechtigungen. Weitere Informationen hierzu finden Sie unter [Tabelle mit HSM-Benutzerberechtigungen für Management Utility AWS CloudHSM](#).

AWS kann keine Operationen an Ihrem ausführen HSMs . AWS kann Ihre Benutzer oder Schlüssel nicht anzeigen oder ändern und mit diesen Schlüsseln keine kryptografischen Operationen ausführen.

Tabelle mit HSM-Benutzerberechtigungen für Management Utility AWS CloudHSM

In der folgenden Tabelle sind die Operationen des Hardware-Sicherheitsmoduls (HSM) aufgeführt, sortiert nach dem Typ des HSM-Benutzers oder der HSM-Sitzung, in der der Vorgang ausgeführt werden kann. AWS CloudHSM

	Crypto Officer (CO)	Crypto-Benutzer (Crypto User, CU)	Appliance-Benutzer (Appliance User, AU)	Nicht authentifizierte Sitzungen
Grundlegende Cluster-Informationen erhalten ¹	 Ja	 Ja	 Ja	 Ja
Das eigene Passwort ändern	 Ja	 Ja	 Ja	Nicht zutreffend
Das Passwort eines Benutzers ändern	 Ja	 Nein	 Nein	 Nein

	Crypto Officer (CO)	Crypto-Benutzer (Crypto User, CU)	Appliance -Benutzer (Appliance User, AU)	Nicht authentif izierte Sitzungen
Benutzer hinzufügen oder entfernen	 Ja	 Nein	 Nein	 Nein
Den Synchroni sierungsstatus erhalten ²	 Ja	 Ja	 Ja	 Nein
Maskierte Objekte extrahier en und einfügen ³	 Ja	 Ja	 Ja	 Nein
Funktionen zur Schlüssel verwaltung ⁴	 Nein	 Ja	 Nein	 Nein
Verschlüsseln/ Entschlüsseln	 Nein	 Ja	 Nein	 Nein
Signieren/ Überprüfen	 Nein	 Ja	 Nein	 Nein

	Crypto Officer (CO)	Crypto-Benutzer (Crypto User, CU)	Appliance-Benutzer (Appliance User, AU)	Nicht authentifizierte Sitzungen
Generieren Sie Digests und HMACs	 Nein	 Ja	 Nein	 Nein

- [1] Zu den grundlegenden Clusterinformationen gehören die Anzahl der HSMs Mitglieder im Cluster und die IP-Adresse, das Modell, die Seriennummer, die Geräte-ID, die Firmware-ID usw. jedes HSM.
- [2] Der Benutzer kann verschiedene Digests (Hashes) erhalten, die den Schlüsseln auf dem HSM entsprechen. Eine Anwendung kann diese Gruppen von Digests vergleichen, um den Synchronisationsstatus innerhalb eines HSMs Clusters zu ermitteln.
- [3] Maskierte Objekte sind Schlüssel, die verschlüsselt werden, bevor sie das HSM verlassen. Sie können außerhalb des HSM nicht entschlüsselt werden. Sie werden erst entschlüsselt, nachdem sie in ein HSM eingefügt wurden, das sich in demselben Cluster befindet wie das HSM, aus dem sie extrahiert wurden. Eine Anwendung kann maskierte Objekte extrahieren und einfügen, um sie HSMs in einem Cluster zu synchronisieren.
- [4] Die Funktionen zur Schlüsselverwaltung umfassen unter anderem das Erstellen, Löschen, Verpacken, Entpacken und Ändern der Schlüsselattribute.

Erstellen Sie HSM-Benutzer mit dem Management Utility AWS CloudHSM

Verwenden Sie `createUser` im AWS CloudHSM Management Utility (CMU), um neue Benutzer auf dem Hardware Security Module (HSM) zu erstellen. Sie müssen sich als CO anmelden, um einen Benutzer zu erstellen.

Um einen neuen CO-Benutzer zu erstellen

1. Verwenden Sie das Configure-Tool, um die CMU-Konfiguration zu aktualisieren.

Linux

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

Windows

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

2. Starten von CMU.

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. Melden Sie sich beim HSM als CO-Benutzer an.

```
aws-cloudhsm > loginHSM C0 admin co12345
```

Stellen Sie sicher, dass die Anzahl der Verbindungen, die die CMU auflistet, mit der Anzahl der Verbindungen HSMs im Cluster übereinstimmt. Wenn nicht, melden Sie sich ab und beginnen Sie von vorne.

4. Verwenden Sie `createUser`, um einen CO-Benutzer namens **example_officer** mit dem Passwort **password1** zu erstellen.

```
aws-cloudhsm > createUser C0 example_officer password1
```

Die CMU fordert Sie auf, den Vorgang zum Erstellen eines Benutzers auszuführen.

```
*****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. Geben Sie **y** ein.

Um einen neuen CU-Benutzer zu erstellen

1. Verwenden Sie das Configure-Tool, um die CMU-Konfiguration zu aktualisieren.

Linux

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

Windows

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

2. Starten von CMU.

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. Melden Sie sich beim HSM als CO-Benutzer an.

```
aws-cloudhsm > loginHSM C0 admin co12345
```

Stellen Sie sicher, dass die Anzahl der Verbindungen, die in den CMU-Listen aufgeführt sind, mit der Anzahl der Verbindungen HSMs im Cluster übereinstimmt. Wenn nicht, melden Sie sich ab und beginnen Sie von vorne.

4. Verwenden Sie `createUser`, um einen CU-Benutzer namens **example_user** mit dem Passwort **password1** zu erstellen.

```
aws-cloudhsm > createUser CU example_user password1
```

Die CMU fordert Sie auf, den Vorgang zum Erstellen eines Benutzers auszuführen.

```
*****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. Geben Sie **y** ein.

Für weitere Informationen über `createUser` finden Sie unter [createUser](#).

Führen Sie mithilfe AWS CloudHSM des Management Utility alle HSM-Benutzer im Cluster auf

Verwenden Sie den `listUsers` Befehl im AWS CloudHSM Management Utility (CMU), um alle Benutzer im AWS CloudHSM Cluster aufzulisten. Sie müssen sich nicht anmelden, um `listUsers` auszuführen, und alle Benutzertypen können Benutzer auflisten.

Um alle Benutzer im Cluster aufzulisten

1. Verwenden Sie das Configure-Tool, um die CMU-Konfiguration zu aktualisieren.

Linux

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

Windows

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

2. Starten von CMU.

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. Verwenden Sie `listUsers`, um alle Benutzer im Cluster aufzulisten.

```
aws-cloudhsm > listUsers
```

CMU listet alle Benutzer des Clusters auf.

Users on server 0(10.0.2.9):

Number of users found:4

User Id	User Type	User Name	2FA
MofnPubKey	LoginFailureCnt		
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	2FA
MofnPubKey	LoginFailureCnt		
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	N0
	0		N0
2	CO	example_officer	N0
	0		N0
3	CU	example_user	N0
	0		N0

Für weitere Informationen über listUsers finden Sie in der [listUsers](#).

Ändern Sie HSM-Benutzerkennwörter mit AWS CloudHSM der Management Utility

Verwenden Sie es changePswd in der AWS CloudHSM Management Utility (CMU), um das Passwort eines HSM-Benutzers (Hardware Security Module) zu ändern.

Bei Benutzertypen und Passwörtern wird zwischen Groß- und Kleinschreibung unterschieden, bei Benutzernamen jedoch nicht.

CO, Crypto-Benutzer (CU) und Appliance-Benutzer (AU) können ihr eigenes Passwort ändern. Um das Passwort eines anderen Benutzers zu ändern, müssen Sie sich als CO anmelden. Sie können das Passwort eines Benutzers, der gerade angemeldet ist, nicht ändern.

So ändern Sie Ihr eigenes Passwort

1. Verwenden Sie das Configure-Tool, um die CMU-Konfiguration zu aktualisieren.

Linux

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

Windows

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

2. Starten von CMU.

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. Anmelden beim HSM.

```
aws-cloudhsm > loginHSM C0 admin co12345
```

Stellen Sie sicher, dass die Anzahl der Verbindungen, die in den CMU-Listen aufgeführt sind, mit der Anzahl der Verbindungen HSMs im Cluster übereinstimmt. Wenn nicht, melden Sie sich ab und beginnen Sie von vorne.

4. Mit changePswd können Sie Ihr eigenes Passwort ändern.

```
aws-cloudhsm > changePswd C0 example_officer <new password>
```

Die CMU informiert Sie über den Vorgang zum Ändern des Passworts.

```
*****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. Geben Sie y ein.

Die CMU informiert Sie über den Vorgang zum Ändern des Passworts.

```
Changing password for example_officer(C0) on 3 nodes
```

So ändern Sie das Passwort eines anderen Benutzers

1. Verwenden Sie das Configure-Tool, um die CMU-Konfiguration zu aktualisieren.

Linux

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

Windows

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

2. Starten von CMU.

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. Melden Sie sich beim HSM als CO-Benutzer an.

```
aws-cloudhsm > loginHSM CO admin co12345
```

Stellen Sie sicher, dass die Anzahl der Verbindungen, die in den CMU-Listen aufgeführt sind, mit der Anzahl der Verbindungen HSMs im Cluster übereinstimmt. Wenn nicht, melden Sie sich ab und beginnen Sie von vorne.

4. Verwenden Sie changePswd, um das Passwort eines anderen Benutzers zu ändern.

```
aws-cloudhsm > changePswd CU example_user <new password>
```

Die CMU informiert Sie über den Vorgang zum Ändern des Passworts.

```
*****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. Geben Sie **y** ein.

Die CMU informiert Sie über den Vorgang zum Ändern des Passworts.

```
Changing password for example_user(CU) on 3 nodes
```

Für weitere Informationen über changePswd finden Sie unter [changePswd](#).

Löschen Sie HSM-Benutzer mithilfe des Management Utility AWS CloudHSM

Verwenden Sie es deleteUser in der AWS CloudHSM Management Utility (CMU), um einen Benutzer des Hardware Security Module (HSM) zu löschen. Sie müssen sich als CO anmelden, um einen anderen Benutzer zu löschen.

Tip

Sie können keine Crypto-Benutzer (CU) löschen, die Schlüssel besitzen.

So löschen Sie einen Benutzer

1. Verwenden Sie das Configure-Tool, um die CMU-Konfiguration zu aktualisieren.

Linux

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

Windows

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

2. Starten von CMU.

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. Melden Sie sich beim HSM als CO-Benutzer an.

```
aws-cloudhsm > loginHSM C0 admin co12345
```

Stellen Sie sicher, dass die Anzahl der Verbindungen, die die CMU auflistet, mit der Anzahl der Verbindungen HSMs im Cluster übereinstimmt. Wenn nicht, melden Sie sich ab und beginnen Sie von vorne.

4. Verwenden Sie `deleteUser`, um einen Benutzer zu löschen.

```
aws-cloudhsm > deleteUser C0 example_officer
```

CMU löscht den Benutzer.

```
Deleting user example_officer(C0) 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)
```

Für weitere Informationen über `deleteUser` finden Sie unter [deleteUser](#).

2FA für Benutzer mit dem AWS CloudHSM Management Utility verwalten

Um die Sicherheit zu erhöhen, können Sie die Zwei-Faktor-Authentifizierung (2FA) konfigurieren, um den Cluster zu schützen. AWS CloudHSM Sie können 2FA nur für Crypto Officers (CO) aktivieren.

Wenn Sie sich mit einem 2FA-fähigen HSM-Konto (Hardware Service Module) bei einem Cluster anmelden, geben Sie `cloudhsm_mgmt_util` (CMU) Ihr Passwort – den ersten Faktor, was Sie wissen – und CMU stellt Ihnen ein Token zur Verfügung und fordert Sie auf, das Token signieren zu lassen.

Um den zweiten Faktor bereitzustellen – was Sie haben – signieren Sie das Token mit einem privaten Schlüssel aus einem Schlüsselpaar, das Sie bereits erstellt und dem HSM-Benutzer zugeordnet haben. Um auf den Cluster zuzugreifen, stellen Sie der CMU das signierte Token zur Verfügung.

 Note

Sie können 2FA nicht für Crypto-Benutzer (CU) oder Anwendungen aktivieren. Die Zwei-Faktor-Authentifizierung (2FA) ist nur für CO-Benutzer verfügbar.

Topics

- [Quorum-Authentifizierung und 2FA in AWS CloudHSM Clustern mithilfe des Management Utility AWS CloudHSM](#)
- [2FA-Schlüsselpaar-Anforderungen für die AWS CloudHSM Verwendung von AWS CloudHSM Management Utility](#)
- [Erstellen Sie Benutzer mit aktivierter 2FA für Benutzer der Management Utility AWS CloudHSM](#)
- [Verwalten Sie 2FA für HSM-Benutzer mit dem Management Utility AWS CloudHSM](#)
- [Deaktivieren Sie 2FA für HSM-Benutzer mithilfe des Management Utility AWS CloudHSM](#)
- [Konfigurationsreferenz für 2FA mit Management Utility AWS CloudHSM](#)

Quorum-Authentifizierung und 2FA in AWS CloudHSM Clustern mithilfe des Management Utility AWS CloudHSM

Der Cluster verwendet denselben Schlüssel für die Quorum-Authentifizierung und für die Zwei-Faktor-Authentifizierung (2FA). Das bedeutet, dass ein Benutzer mit aktivierter 2FA effektiv für M-of-N-access -Control (MoFN) registriert ist. Beachten Sie die folgenden Punkte, um 2FA und die Quorum-Authentifizierung erfolgreich für denselben HSM-Benutzer zu verwenden:

- Wenn Sie heute die Quorum-Authentifizierung für einen Benutzer verwenden, sollten Sie dasselbe Schlüsselpaar verwenden, das Sie für den Quorum-Benutzer erstellt haben, um 2FA für den Benutzer zu aktivieren.
- Wenn Sie die 2FA-Anforderung für einen Nicht-2FA-Benutzer hinzufügen, der kein Quorum-Authentifizierungsbenutzer ist, registrieren Sie diesen Benutzer als MoFN-Benutzer mit 2FA-Authentifizierung.

- Wenn Sie die 2FA-Anforderung entfernen oder das Passwort für einen 2FA-Benutzer ändern, der auch ein Quorum-Authentifizierungsbenutzer ist, entfernen Sie auch die Registrierung des Quorumbenutzers als MoFN-Benutzer.
- Wenn Sie die 2FA-Anforderung aufheben oder das Passwort für einen 2FA-Benutzer ändern, der auch ein Quorum-Authentifizierungsbenutzer ist, Sie aber trotzdem möchten, dass dieser Benutzer an der Quorumauthentifizierung teilnimmt, müssen Sie diesen Benutzer erneut als MoFN-Benutzer registrieren.

Weitere Informationen über die Quorum-Authentifizierung finden Sie unter [Verwendung von CMU zur Verwaltung der Quorum-Authentifizierung](#).

2FA-Schlüsselpaar-Anforderungen für die AWS CloudHSM Verwendung von AWS CloudHSM Management Utility

Um die Zwei-Faktor-Authentifizierung (2FA) für einen Benutzer eines AWS CloudHSM Hardware-Sicherheitsmoduls (HSM) zu aktivieren, verwenden Sie einen Schlüssel, der die folgenden Anforderungen erfüllt.

Sie können ein neues Schlüsselpaar erstellen oder einen vorhandenen Schlüssel verwenden, der die folgenden Anforderungen erfüllt.

- Schlüsseltyp: Asymmetrisch
- Schlüsselnutzung: Signieren und Verifizieren
- Schlüsselspezifikation: RSA_2048
- Der Signierungsalgorithmus umfasst:
 - sha256WithRSAEncryption

Note

Wenn Sie die Quorumauthentifizierung verwenden oder planen, die Quorumauthentifizierung zu verwenden, finden Sie weitere Informationen unter [the section called “Quorum-Authentifizierung”](#).

Erstellen Sie Benutzer mit aktivierter 2FA für Benutzer der Management Utility AWS CloudHSM

Verwenden Sie das AWS CloudHSM Management Utility CMU (CMU) und das key pair, um einen neuen Crypto Office (CO) -Benutzer mit aktivierter Zwei-Faktor-Authentifizierung (2FA) zu erstellen.

Um CO-Benutzer mit aktivierter 2FA zu erstellen

1. Führen Sie an einem Terminal die folgenden Schritte aus:
 - a. Greifen Sie auf Ihr HSM zu und melden Sie sich beim CloudHSM Management Utility an:

```
/opt/cloudhsm/bin/cloudhsm_mgmt_util /opt/cloudhsm/etc/cloudhsm_mgmt_util.cfg
```

- b. Melden Sie sich als CO an und erstellen Sie mit dem folgenden Befehl eine neue Benutzer-MFA mit 2FA:

```
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. Belassen Sie das obige Terminal in diesem Zustand. Drücken Sie nicht die Eingabetaste und geben Sie keinen Dateinamen ein.
2. Führen Sie in einem anderen Terminal die folgenden Schritte aus:
 - a. Greifen Sie auf Ihr HSM zu und melden Sie sich beim CloudHSM Management Utility an:

```
/opt/cloudhsm/bin/cloudhsm_mgmt_util /opt/cloudhsm/etc/cloudhsm_mgmt_util.cfg
```

- b. Generieren Sie mit den folgenden Befehlen ein öffentlich-privates Schlüsselpaar:

```
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. Führen Sie den folgenden Befehl aus, um eine JSON-Abfragefunktion zum Extrahieren des Digest aus der Authdata-Datei zu installieren:

```
sudo yum install jq
```

- d. Um den Digest-Wert zu extrahieren, suchen Sie zunächst die folgenden Daten in der Authdata-Datei:

```
{  
  "Version": "1.0",  
  "PublicKey": "",  
  "Data": [  
    {  
      "HsmId": <"HSM ID">,  
      "Digest": <"DIGEST">,  
      "Signature": ""  
    }  
  ]  
}
```

 Note

Der erhaltene Digest ist Base64-kodiert. Um den Digest zu signieren, muss die Datei jedoch zuerst dekodiert und dann signiert werden. Der folgende Befehl dekodiert den Digest und speichert den dekodierten Inhalt in „digest1.bin“.

```
cat authdata | jq '.Data[0].Digest' | cut -c2- | rev | cut -c2- | rev |  
base64 -d > digest1.bin
```


- e. Konvertiert den Inhalt des öffentlichen Schlüssels, fügt „\n“ hinzu und entfernt Leerzeichen, wie hier gezeigt:

```
-----BEGIN PUBLIC KEY-----\n<PUBLIC KEY>\n-----END PUBLIC KEY-----
```

 Important

Der obige Befehl zeigt, wie „\n“ unmittelbar nach BEGIN PUBLIC KEY----- hinzugefügt wird, Leerzeichen zwischen „\n“ und dem ersten Zeichen des öffentlichen Schlüssels entfernt werden, „\n“ vor -----END PUBLIC KEY hinzugefügt wird und Leerzeichen zwischen „\n“ und dem Ende des öffentlichen Schlüssels entfernt werden.

Dies ist das PEM-Format für den öffentlichen Schlüssel, das in der Authdata-Datei akzeptiert wird.

- f. Fügen Sie den Inhalt des öffentlichen Schlüssels im PEM-Format in den Abschnitt mit dem öffentlichen Schlüssel der Authdata-Datei ein.

```
vi authdata
```

```
{
  "Version": "1.0",
  "PublicKey": "-----BEGIN PUBLIC KEY-----\n<"PUBLIC KEY">\n-----END PUBLIC KEY-----",
  "Data": [
    {
      "HsmId": "<"HSM ID">",
      "Digest": "<"DIGEST">",
      "Signature": ""
    }
  ]
}
```

- g. Signieren Sie die Token-Datei mit dem folgenden Befehl:

```
openssl pkeyutl -sign -in digest1.bin -inkey private_key.pem -pkeyopt
digest:sha256 | base64
Output Expected:
```

```
<"THE SIGNATURE">
```

 Note

Verwenden Sie, wie im obigen Befehl gezeigt, `openssl pkeyutl` statt `openssl dgst` zum Signieren.

- h. Fügen Sie den signierten Digest in die Authdata-Datei im Feld „Signatur“ ein.

```
vi authdata
```

```
{
  "Version": "1.0",
  "PublicKey": "-----BEGIN PUBLIC KEY----- ... -----END PUBLIC KEY-----",
  "Data": [
    {
      "HsmId": <"HSM ID">,
      "Digest": <"DIGEST">,
      "Signature": <"Kkd1 ... rkrvJ6Q==">
    },
    {
      "HsmId": <"HSM ID">,
      "Digest": <"DIGEST">,
      "Signature": <"K1hxy ... Q261Q==">
    }
  ]
}
```

3. Gehen Sie zurück zum ersten Terminal und drücken Sie **Enter**:

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 of authentication for this user. Paste the signatures and the corresponding public key into 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)
```

Verwalten Sie 2FA für HSM-Benutzer mit dem Management Utility AWS CloudHSM

Verwenden Sie es `changePswd` im AWS CloudHSM Management Utility (CMU), um die Zwei-Faktor-Authentifizierung (2FA) für einen Benutzer zu ändern. Jedes Mal, wenn Sie 2FA aktivieren, müssen Sie einen öffentlichen Schlüssel für 2FA-Logins angeben.

`changePswd` führt eines der folgenden Szenarien aus:

- Ändern Sie das Passwort für einen 2FA-Benutzer
- Ändern Sie das Passwort für einen Nicht-2FA-Benutzer
- 2FA zu einem Nicht-2FA-Benutzer hinzufügen
- 2FA von einem 2FA-Benutzer entfernen
- Den Schlüssel für einen 2FA-Benutzer rotieren

Sie können Aufgaben auch kombinieren. Sie können beispielsweise 2FA von einem Benutzer entfernen und gleichzeitig das Passwort ändern, oder Sie können den 2FA-Schlüssel wechseln und das Benutzerpasswort ändern.

Um Passwörter zu ändern oder Schlüssel für CO-Benutzer mit aktivierter 2FA zu rotieren

1. Verwenden Sie CMU, um sich als CO mit aktivierter 2FA am HSM anzumelden.
2. Verwenden Sie `changePswd`, um das Passwort zu ändern oder den Schlüssel von CO-Benutzern mit aktivierter 2FA zu wechseln. Verwenden Sie den `-2fa`-Parameter und geben Sie einen Speicherort im Dateisystem an, an dem das System die `authdata`-Datei schreiben kann. Diese Datei enthält einen Digest für jedes HSM im Cluster.

```
aws-cloudhsm > changePswd CO example-user <new-password> -2fa /path/to/authdata
```

CMU fordert Sie auf, den privaten Schlüssel zu verwenden, um die Digests in der `authdata`-Datei zu signieren und die Signaturen mit dem öffentlichen Schlüssel zurückzugeben.

3. Verwenden Sie den privaten Schlüssel, um die Digests in der `authdata`-Datei zu signieren, fügen Sie die Signaturen und den öffentlichen Schlüssel zur `authdata`-Datei im JSON-Format hinzu und teilen Sie CMU dann den Speicherort der `authdata`-Datei mit. Weitere Informationen finden Sie unter [the section called “Konfigurationsreferenz”](#).

 Note

Der Cluster verwendet denselben Schlüssel für die Quorum-Authentifizierung und für 2FA. Wenn Sie die Quorumauthentifizierung verwenden oder planen, die Quorumauthentifizierung zu verwenden, finden Sie weitere Informationen unter [the section called “Quorum-Authentifizierung”](#).

Deaktivieren Sie 2FA für HSM-Benutzer mithilfe des Management Utility AWS CloudHSM

Verwenden Sie das AWS CloudHSM Management Utility (CMU), um die Zwei-Faktor-Authentifizierung (2FA) für Benutzer des Hardware-Sicherheitsmoduls (HSM) in zu deaktivieren. AWS CloudHSM

So deaktivieren Sie 2FA für CO-Benutzer mit aktivierter 2FA

1. Verwenden Sie CMU, um sich als CO mit aktivierter 2FA am HSM anzumelden.
2. Verwenden Sie `changePswd`, um 2FA von CO-Benutzern zu entfernen, bei denen 2FA aktiviert ist.

```
aws-cloudhsm > changePswd CO example-user <new password>
```

CMU fordert Sie auf, den Vorgang zum Ändern des Passworts zu bestätigen.

 Note

Wenn Sie die 2FA-Anforderung entfernen oder das Passwort für einen 2FA-Benutzer ändern, der auch ein Quorum-Authentifizierungsbutzer ist, entfernen Sie auch die Registrierung des Quorumbenutzers als MoFN-Benutzer. Weitere Informationen über Quorum-Benutzer und 2FA finden Sie unter [the section called “Quorum-Authentifizierung”](#).

3. Geben Sie **y** ein.

Die CMU bestätigt den Vorgang zum Ändern des Passworts.

Konfigurationsreferenz für 2FA mit Management Utility AWS CloudHSM

Im Folgenden finden Sie ein Beispiel für die Eigenschaften der Zwei-Faktor-Authentifizierung (2FA) in der authdata Datei sowohl für die von der AWS CloudHSM Management Utility (CMU) generierte Anfrage als auch für Ihre Antworten.

```
{
  "Version": "1.0",
  "PublicKey": "-----BEGIN PUBLIC KEY----- ... -----END PUBLIC KEY-----",
  "Data": [
    {
      "HsmId": "hsm-lgavqitns2a",
      "Digest": "k501p3f6foQRVQH7S8Rrjcau6h3TYqsSdr16A54+qG8=",
      "Signature": "Kkd1 ... rkrvJ6Q=="
    },
    {
      "HsmId": "hsm-lgavqitns2a",
      "Digest": "IyBcx4I5Vyx1jztwvXinCBQd9lDx8oQe7iRrWjBAi1w=",
      "Signature": "K1hxy ... Q261Q=="
    }
  ]
}
```

Daten

Oberster Knoten Enthält einen untergeordneten Knoten für jedes HSM im Cluster. Erscheint in Anfragen und Antworten für alle 2FA-Befehle.

Digest

Dies müssen Sie unterschreiben, um den zweiten Authentifizierungsfaktor bereitzustellen. CMU wurde in Anfragen für alle 2FA-Befehle generiert.

HsmId

Die ID Ihres HSM. Erscheint in Anfragen und Antworten für alle 2FA-Befehle.

PublicKey

Der öffentliche Schlüsselteil des Schlüsselpaar, das Sie generiert haben, wurde als Zeichenfolge im PEM-Format eingefügt. Sie geben dies in Antworten für createUser und changePswd ein.

Signature

Der mit Base 64 kodierte signierte Digest. Sie geben dies in Antworten für alle 2FA-Befehle ein.

Version

Die Version der JSON-formatierten Datei mit den Authentifizierungsdaten erscheint in Anfragen und Antworten für alle 2FA-Befehle.

Verwendung von CloudHSM Management Utility (CMU) zur Verwaltung der Quorum-Authentifizierung (M-von-N-Zugriffskontrolle)

Die HSMs in Ihrem AWS CloudHSM Cluster unterstützen die Quorumauthentifizierung, die auch als M of N-Zugriffskontrolle bezeichnet wird. Mit der Quorum-Authentifizierung kann kein einzelner Benutzer im HSM quorum-kontrollierte Operationen im HSM durchführen. Stattdessen muss eine Mindestanzahl von HSM-Benutzern (mindestens 2) zusammenarbeiten, um diese Operationen durchzuführen. Mit der Quorum-Authentifizierung können Sie eine zusätzliche Schutzebene hinzufügen, indem Sie Genehmigungen von mehr als einem HSM-Benutzer erfordern.

Die Quorum-Authentifizierung kann die folgenden Vorgänge steuern:

- HSM-Benutzerverwaltung durch [Crypto Officers \(COs\)](#) — HSM-Benutzer erstellen und löschen und das Passwort eines anderen HSM-Benutzers ändern. Weitere Informationen finden Sie unter [Benutzerverwaltung mit aktivierter Quorum-Authentifizierung für AWS CloudHSM Management Utility](#).

Beachten Sie die folgenden, zusätzlichen Informationen zur Verwendung der Quorum-Authentifizierung in AWS CloudHSM.

- Ein HSM-Benutzer kann sein eigenes Quorum-Token signieren, d. h. der anfordernde Benutzer kann eine der erforderlichen Genehmigungen für die Quorum-Authentifizierung bereitstellen.
- Sie wählen die Mindestzahl der Quorum-Genehmiger für die Quorum-kontrollierten Operationen aus. Die kleinste Zahl, die Sie wählen können, ist zwei (2), und die größte Zahl, die Sie wählen können, ist acht (8).
- Das HSM kann bis zu 1024 Quorum-Token speichern. Wenn das HSM beim Versuch, ein neues zu erstellen, bereits 1024 Token hat, löscht das HSM eines der abgelaufenen Token. Standardmäßig verfallen Token zehn Minuten nach ihrer Erstellung.
- Der Cluster verwendet denselben Schlüssel für die Quorum-Authentifizierung und für die Zwei-Faktor-Authentifizierung (2FA). [Weitere Informationen zur Verwendung von Quorum-Authentifizierung und 2FA finden Sie unter Quorum-Authentifizierung und 2FA.](#)

Die folgenden Themen stellen weitere Informationen zur Quorum-Authentifizierung in AWS CloudHSM zur Verfügung.

Themen

- [Quorum-Authentifizierungsprozess für Management Utility AWS CloudHSM](#)
- [Richten Sie die Quorum-Authentifizierung für AWS CloudHSM Crypto Officers ein](#)
- [Benutzerverwaltung mit aktivierter Quorum-Authentifizierung für AWS CloudHSM Management Utility](#)
- [Ändern Sie den Quorum-Mindestwert mit dem AWS CloudHSM Management Utility](#)

Quorum-Authentifizierungsprozess für Management Utility AWS CloudHSM

Die folgenden Schritte fassen die Verfahren zur Quorum-Authentifizierung zusammen. Die exakten Schritte und Tools finden Sie unter [Benutzerverwaltung mit aktivierter Quorum-Authentifizierung für AWS CloudHSM Management Utility](#).

1. Jeder HSM-Benutzer erstellt einen asymmetrischen Schlüssel zum Signieren. Er oder sie tut dies außerhalb des HSMs und achtet darauf, den Schlüssel angemessen zu schützen.
2. Jeder HSM-Benutzer meldet sich beim HSM an und registriert den öffentlichen Teil seines Signierungsschlüssels (den öffentlichen Schlüssel) beim HSM.
3. Wenn ein HSM-Benutzer eine Quorum-kontrollierte Operation durchführen möchte, meldet er sich beim HSM an und erhält ein Quorum-Token.
4. Der HSM-Benutzer gibt das Quorum-Token an einen oder mehrere andere HSM-Benutzer weiter und bittet um deren Zustimmung.
5. Die anderen HSM-Benutzer genehmigen, indem sie mit ihren Schlüsseln das Quorum-Token kryptographisch signieren. Dies geschieht außerhalb des HSMs.
6. Wenn der HSM-Benutzer über die erforderliche Anzahl von Genehmigungen verfügt, meldet er sich beim HSM an und gibt das Quorum-Token und die Genehmigungen (Signaturen) an das HSM weiter.
7. Das HSM verwendet die registrierten, öffentlichen Schlüssel jedes Unterzeichners, um die Signaturen zu verifizieren. Wenn die Signaturen gültig sind, genehmigt das HSM das Token.
8. Der HSM-Benutzer kann nun eine quorum-kontrollierte Operation durchführen.

Richten Sie die Quorum-Authentifizierung für AWS CloudHSM Crypto Officers ein

In den folgenden Themen werden die Schritte beschrieben, die Sie ausführen müssen, um Ihr Hardware-Sicherheitsmodul (HSM) so zu konfigurieren, dass AWS CloudHSM [Crypto Officers \(COs\) die Quorum-Authentifizierung verwenden](#) können. Sie müssen diese Schritte nur einmal ausführen, wenn Sie die Quorumauthentifizierung für zum ersten Mal konfigurieren. COs Nachdem Sie diese Schritte abgeschlossen haben, fahren Sie mit [Benutzerverwaltung mit aktivierter Quorum-Authentifizierung für AWS CloudHSM Management Utility](#) fort.

Themen

- [Voraussetzungen](#)
- [Schritt 1. Erstellen und Registrieren eines Schlüssels für das Signieren](#)
- [Schritt 2. Setzen des Quorum-Mindestwerts für das HSM](#)

Voraussetzungen

Um dieses Beispiel zu verstehen, sollten Sie mit dem [cloudhsm_mgmt_util-Kommandozeilen-Tool \(CMU\)](#) vertraut sein. In diesem Beispiel hat der AWS CloudHSM Cluster zwei HSMs, von denen jeder dasselbe hat COs, wie in der folgenden Ausgabe des listUsers Befehls dargestellt. Weitere Informationen zum Erstellen von Benutzern finden Sie unter [HSM-Benutzer](#).

```
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	N0
0	N0		
2	AU	app_user	N0
0	N0		
3	CO	officer1	N0
0	N0		
4	CO	officer2	N0
0	N0		
5	CO	officer3	N0
0	N0		
6	CO	officer4	N0
0	N0		

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		

Schritt 1. Erstellen und Registrieren eines Schlüssels für das Signieren

Um die Quorumauthentifizierung zu verwenden, muss jeder CO alle der folgenden Schritte ausführen:

Themen

- [Erstellen eines RSA-Schlüsselpaares](#)
- [Erstellen und signieren Sie ein Registrierungstoken](#)
- [Registrieren Sie den öffentlichen Schlüssel beim HSM](#)

Erstellen eines RSA-Schlüsselpaares

Es gibt viele verschiedene Möglichkeiten, ein Schlüsselpaar zu erstellen und zu schützen. In den folgenden Beispielen wird gezeigt, wie dies mit [OpenSSL](#) durchgeführt wird.

Example– Erstellen eines privaten Schlüssels mit OpenSSL

Im folgenden Beispiel wird gezeigt, wie OpenSSL verwendet wird, um einen 2048-Bit-RSA-Schlüssel zu erstellen, der durch eine Pass-Phrase geschützt ist. Um dieses Beispiel zu verwenden,

officer1.key ersetzen Sie es durch den Namen der Datei, in der Sie den Schlüssel speichern möchten.

```
$ 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:
```

Generieren eines öffentlichen Schlüssels mit dem privaten Schlüssel, den Sie gerade erstellt haben.

Example– Erstellen Sie einen öffentlichen Schlüssel mit OpenSSL

Das folgende Beispiel zeigt, wie Sie OpenSSL verwenden, um einen öffentlichen Schlüssel aus dem privaten Schlüssel zu erstellen, den Sie gerade erstellt haben.

```
$ openssl rsa -in officer1.key -outform PEM -pubout -out officer1.pub
Enter pass phrase for officer1.key:
writing RSA key
```

Erstellen und signieren Sie ein Registrierungstoken

Sie erstellen ein Token und signieren es mit dem privaten Schlüssel, den Sie gerade im vorherigen Schritt generiert haben.

Example– Erstellen Sie ein Token

Das Registrierungstoken ist nur eine Datei mit beliebigen zufälligen Daten, die die maximale Größe von 245 Byte nicht überschreiten. Sie signieren das Token mit dem privaten Schlüssel, um nachzuweisen, dass Sie Zugriff auf den privaten Schlüssel haben. Der folgende Befehl verwendet Echo, um eine Zeichenfolge in eine Datei umzuleiten.

```
$ echo <token to be signed> > officer1.token
```

Signieren Sie das Token und speichern Sie es in einer Signaturdatei. Sie benötigen das signierte Token, das unsignierte Token und den öffentlichen Schlüssel, um den CO als MoFN-Benutzer beim HSM zu registrieren.

Example– Signieren Sie das Token

Verwenden Sie OpenSSL und den privaten Schlüssel, um das Registrierungstoken zu signieren und die Signaturdatei zu erstellen.

```
$ openssl dgst -sha256 \
    -sign officer1.key \
    -out officer1.token.sig officer1.token
```

Registrieren Sie den öffentlichen Schlüssel beim HSM

Nach dem Erstellen eines Schlüssels muss der CO den öffentlichen Teil des Schlüssels (den öffentlichen Schlüssel) beim HSM registrieren.

So registrieren Sie einen öffentlichen Schlüssel bei dem HSM

1. Starten Sie das Befehlszeilen-Tool `cloudhsm_mgmt_util` mit folgendem Befehl.

```
$ /opt/cloudhsm/bin/cloudhsm_mgmt_util /opt/cloudhsm/etc/cloudhsm_mgmt_util.cfg
```

2. Verwenden Sie den Befehl `loginHSM`, um sich bei den HSMs als CO anzumelden. Weitere Informationen finden Sie unter [???](#).
3. Verwenden Sie den Befehl [registerQuorumPubKey](#), um den öffentlichen Schlüssel zu registrieren. Weitere Informationen finden Sie im folgenden Beispiel. Alternativ können Sie auch den Befehl `help registerQuorumPubKey` ausführen.

Example– Registrieren eines öffentlichen Schlüssels beim HSM

Das folgende Beispiel zeigt, wie Sie mit dem Befehl `registerQuorumPubKey` im `cloudhsm_mgmt_util`-Befehlszeilen-Tool den öffentlichen Schlüssel eines COs beim HSM registrieren. Der CO muss bei dem HSM angemeldet sein, um diesen Befehl zu verwenden. Ersetzen Sie diese Werte durch Ihre eigenen Werte:

```
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>

Der Pfad zu einer Datei, die ein unsigniertes Registrierungstoken enthält. Kann beliebige Daten mit einer maximalen Dateigröße von 245 Byte enthalten.

Erforderlich: Ja

<officer1.token.sig>

Der Pfad zu einer Datei, die den vom SHA256_PKCS-Mechanismus signierten Hash des Registrierungstokens enthält.

Erforderlich: Ja

<officer1.pub>

Der Pfad zu der Datei, die den öffentlichen Schlüssel eines asymmetrischen RSA-2048-Schlüsselpaars enthält. Verwenden Sie den privaten Schlüssel, um das Registrierungstoken zu signieren.

Erforderlich: Ja

Nachdem alle ihre öffentlichen Schlüssel COs registriert haben, zeigt die Ausgabe des `listUsers` Befehls dies in der `MofnPubKey` Spalte an, wie im folgenden Beispiel gezeigt.

```
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 LoginFailureCnt	User Type 2FA	User Name	MofnPubKey
1 0	PRECO NO	admin	NO
2 0	AU NO	app_user	NO
3 0	CO NO	officer1	YES
4 0	CO NO	officer2	YES
5 0	CO NO	officer3	YES
6 0	CO NO	officer4	YES
7 0	CO NO	officer5	YES

Schritt 2. Setzen des Quorum-Mindestwerts für das HSM

Um die Quorumauthentifizierung für zu verwenden COs, muss sich ein CO beim HSM anmelden und dann den Quorum-Mindestwert festlegen, der auch als M-Wert bezeichnet wird. Dies ist die Mindestanzahl von CO-Genehmigungen, die erforderlich sind, um HSM-Benutzermanagement-Vorgänge auszuführen. Jeder CO auf dem HSM kann den Quorum-Mindestwert festlegen, auch diejenigen, COs die keinen Schlüssel zum Signieren registriert haben. Sie können den Quorum-Mindestwert jederzeit ändern. Weitere Informationen finden Sie unter [Ändern Sie den Mindestwert](#).

So legen Sie den Quorum-Mindestwert im HSM fest

1. Starten Sie das Befehlszeilen-Tool `cloudhsm_mgmt_util` mit folgendem Befehl.

```
$ /opt/cloudhsm/bin/cloudhsm_mgmt_util /opt/cloudhsm/etc/cloudhsm_mgmt_util.cfg
```

2. Verwenden Sie den Befehl loginHSM, um sich bei den HSMs als CO anzumelden. Weitere Informationen finden Sie unter [???](#).
3. Verwenden Sie den setMValue-Befehl, um den Quorum-Mindestwert festzulegen. Weitere Informationen finden Sie im folgenden Beispiel. Alternativ können Sie auch den Befehl help setMValue ausführen.

Example– Setzen des Quorum-Mindestwerts für das HSM

In diesem Beispiel wird ein Quorum-Mindestwert von zwei verwendet. Sie können einen beliebigen Wert zwischen zwei (2) und acht (8) wählen, bis hin zur Gesamtzahl von COs auf dem HSM. In diesem Beispiel hat das HSM sechs COs, sodass der maximal mögliche Wert sechs ist.

Um den folgenden Beispielbefehl zu verwenden, ersetzen Sie die letzte Zahl (2) durch den bevorzugten Quorum-Mindestwert.

```
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
```

Im vorhergehenden Beispiel bestimmt die erste Zahl (3) den HSM-Service, dessen Quorum-Mindestwert Sie festlegen.

In der folgenden Tabelle sind die HSM-Dienstkennungen zusammen mit ihren Namen, Beschreibungen und den Befehlen aufgeführt, die im Dienst enthalten sind.

Dienstkennung	Service-Name	Service description (Service-Beschreibung)	HSM-Befehle
3	USER_MGMT	HSM-Benutzerverwaltung	• createUser • deleteUser

Dienstkennung	Service-Name	Service description (Service-Beschreibung)	HSM-Befehle
			changePswd (gilt nur, wenn das Passwort eines anderen HSM-Benutzers geändert wird)
4	MISC_CO	Diverser CO-Service	setMValue

Um den Quorum-Mindestwert für einen Service abzurufen, verwenden Sie den `getMValue`-Befehl wie im folgenden Beispiel.

```
aws-cloudhsm > getMValue 3
MValue of service 3[USER_MGMT] on server 0 : [2]
MValue of service 3[USER_MGMT] on server 1 : [2]
```

Die Ausgabe aus dem vorausgehenden Befehl `getMValue` zeigt, dass der Quorum-Mindestwert für HSM-Benutzermanagement-Vorgänge (Service 3) jetzt zwei ist.

Nachdem Sie diese Schritte abgeschlossen haben, fahren Sie mit [Benutzerverwaltung mit aktivierter Quorum-Authentifizierung für AWS CloudHSM Management Utility](#) fort.

Benutzerverwaltung mit aktivierter Quorum-Authentifizierung für AWS CloudHSM Management Utility

Ein AWS CloudHSM [Crypto Officer \(CO\)](#) auf dem Hardware-Sicherheitsmodul (HSM) kann die Quorumauthentifizierung für die folgenden Vorgänge auf dem HSM konfigurieren:

- Erstellen von HSM-Benutzern
- Löschen von HSM-Benutzern
- Ändern des Passworts eines anderen HSM-Benutzers

Nachdem das HSM für die Quorumauthentifizierung konfiguriert wurde, COs können HSM-Benutzerverwaltungsvorgänge nicht mehr eigenständig ausgeführt werden. Das folgende Beispiel

zeigt die Ausgabe, die angezeigt wird, wenn ein CO versucht, einen neuen Benutzer auf dem HSM anzulegen. Der Befehl schlägt mit einem RET_MXN_AUTH_FAILED-Fehler fehl. Dies weist darauf hin, dass die Quorum-Authentifizierung nicht möglich war.

```
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
```

Zum Ausführen eines HSM-Benutzermanagement-Vorgangs muss ein CO folgende Aufgaben erledigen:

1. [Abrufen eines Quorum-Tokens](#).
2. [Holen Sie sich Genehmigungen \(Signaturen\)](#) von anderen ein. COs
3. [Genehmigen des Tokens im HSM](#).
4. [Durchführen des HSM-Benutzermanagement-Vorgangs](#).

Wenn Sie das HSM noch nicht für die Quorumauthentifizierung für konfiguriert haben COs, tun Sie das jetzt. Weitere Informationen finden Sie unter [Erstmalige Einrichtung](#).

Schritt 1. Abrufen eines Quorum-Tokens

Zunächst muss der CO das cloudhsm_mgmt_util-Befehlszeilen-Tool verwenden, um ein Quorum-Token anzufordern.

So fordern Sie ein Quorum-Token an

1. Starten Sie das Befehlszeilen-Tool cloudhsm_mgmt_util mit folgendem Befehl.

```
$ /opt/cloudhsm/bin/cloudhsm_mgmt_util /opt/cloudhsm/etc/cloudhsm_mgmt_util.cfg
```


2. Verwenden Sie den Befehl `loginHSM`, um sich bei den HSMs als CO anzumelden. Weitere Informationen finden Sie unter [???](#).
3. Verwenden Sie zum Abrufen eines Quorum-Tokens den Befehl `getToken`. Weitere Informationen finden Sie im folgenden Beispiel. Alternativ können Sie auch den Befehl `help getToken` ausführen.

Example– Abrufen eines Quorum-Tokens

In diesem Beispiel wird ein Quorum-Token für den CO mit dem Benutzernamen „officer1“ abgerufen und in einer Datei namens `officer1.token` gespeichert. Zum Verwenden des Beispielbefehls ersetzen Sie diese Werte durch Ihre eigenen:

- **officer1**— Der Name des CO, der das Token erhält. Dies muss derselbe CO sein, der am HSM angemeldet ist und diesen Befehl ausführt.
- **officer1.token**— Der Name der Datei, die zum Speichern des Quorum-Tokens verwendet werden soll.

Beim folgenden Befehl identifiziert 3 den Service, für den Sie das Token verwenden können, das Sie abrufen. In diesem Fall ist das Token für die HSM-Benutzermanagement-Vorgänge (Service 3). Weitere Informationen finden Sie unter [Schritt 2. Setzen des Quorum-Mindestwerts für das HSM](#).

```
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
```

Schritt 2. Holen Sie sich Signaturen beim Genehmigen COs

Ein CO, der über ein Quorum-Token verfügt, muss das Token von anderen genehmigen lassen. COs Um ihre Zustimmung zu erteilen, COs verwenden die anderen ihren Signaturschlüssel, um das Token kryptografisch zu signieren. Dies geschieht außerhalb des HSMs.

Es gibt viele verschiedene Möglichkeiten, ein Token zu signieren. Im folgenden Beispiel wird gezeigt, wie dies mit [OpenSSL](#) durchgeführt wird. Wenn Sie ein anderes Signatur-Tool verwenden, müssen Sie sicherstellen, dass dieses den privaten Schlüssel (Signaturschlüssel) des COs zur Signierung des SHA-256-Digests des Tokens verwendet.

Example— Holen Sie sich Signaturen beim Genehmigen COs

In diesem Beispiel benötigt der CO mit dem Token (officer1) mindestens zwei Genehmigungen. Die folgenden Beispielbefehle zeigen, wie zwei Personen OpenSSL verwenden COs können, um das Token kryptografisch zu signieren.

Beim ersten Befehl signiert „officer1“ das eigene Token. Wenn Sie die folgenden Beispielbefehle verwenden, ersetzen Sie diese Werte durch Ihre eigenen:

- *officer1.key* und *officer2.key* — Der Name der Datei, die den Signaturschlüssel des CO enthält.
- *officer1.token.sig1* und *officer1.token.sig2* — Der Name der Datei, die zum Speichern der Signatur verwendet werden soll. Stellen Sie sicher, dass jede Signatur in einer anderen Datei gespeichert wird.
- *officer1.token* — Der Name der Datei, die das Token enthält, das der CO signiert.

```
$ openssl dgst -sha256 -sign officer1.key -out officer1.token.sig1 officer1.token
Enter pass phrase for officer1.key:
```

Beim folgenden Befehl signiert „officer2“ dasselbe Token.

```
$ openssl dgst -sha256 -sign officer2.key -out officer1.token.sig2 officer1.token
Enter pass phrase for officer2.key:
```

Schritt 3. Genehmigen des signierten Tokens auf dem HSM

Nachdem ein CO die Mindestanzahl an Genehmigungen (Signaturen) von anderen Personen erhalten hat COs, muss er oder sie das signierte Token auf dem HSM genehmigen.

So genehmigen Sie das signierte Token auf dem HSM

1. Erstellen Sie eine Token-Genehmigungs-Datei. Weitere Informationen finden Sie im folgenden Beispiel.
2. Starten Sie das Befehlszeilen-Tool `cloudhsm_mgmt_util` mit folgendem Befehl.

```
$ /opt/cloudhsm/bin/cloudhsm_mgmt_util /opt/cloudhsm/etc/cloudhsm_mgmt_util.cfg
```

3. Verwenden Sie den Befehl `loginHSM`, um sich bei den HSMs als CO anzumelden. Weitere Informationen finden Sie unter [???](#).
4. Verwenden Sie den Befehl `approveToken` zum Genehmigen des signierten Tokens, indem Sie die Token-Genehmigungs-Datei übergeben. Weitere Informationen finden Sie im folgenden Beispiel.

Example– Erstellen einer Token-Genehmigungs-Datei und Genehmigen des signierten Tokens auf dem HSM

Die Token-Genehmigungs-Datei ist eine Textdatei in einem Format, das für das HSM erforderlich ist. Die Datei enthält Informationen über das Token, die Genehmigenden und dessen Signaturen. Im Folgenden wird ein Beispiel einer Token-Genehmigungs-Datei gezeigt.

```
# 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;
```

Nach dem Erstellen der Token-Genehmigungsdatei verwendet der CO das `cloudhsm_mgmt_util`-Befehlszeilen-Tool für die Anmeldung am HSM. Anschließend genehmigt der CO das Token mit dem Befehl `approveToken`, wie im folgenden Beispiel gezeigt. *approval.txt* Ersetzen Sie es durch den Namen der Token-Genehmigungsdatei.

```
aws-cloudhsm > approveToken approval.txt
approveToken success on server 0(10.0.2.14)
approveToken success on server 1(10.0.1.4)
```

Wurde der Befehl erfolgreich ausgeführt, hat das HSM das Quorum-Token genehmigt. Zum Prüfen des Token-Status nutzen Sie den Befehl `listTokens`, wie im folgenden Beispiel gezeigt. Die Befehlsausgabe zeigt, dass das Token über die erforderliche Anzahl von Genehmigungen verfügt.

Die Token-Gültigkeitsdauer gibt an, wie lange das Token auf dem HSM erhalten bleibt. Sie können das Token auch nach Ablauf der Token-Gültigkeitsdauer (null Sekunden) noch verwenden.

```
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
```

Schritt 4. Verwenden des Tokens für Benutzermanagement-Vorgänge

Nachdem ein CO über ein Token mit ausreichender Anzahl an Genehmigungen verfügt (wie im vorherigen Abschnitt gezeigt), kann er eine der folgenden HSM-Benutzermanagement-Vorgänge ausführen:

- Erstellen eines HSM-Benutzers mit dem Befehl [createUser](#)
- Löschen eines HSM-Benutzers mit dem Befehl `deleteUser`
- Ändern des Passworts eines anderen HSM-Benutzers mit dem Befehl `changePswd`

Weitere Informationen zur Verwendung dieser Befehle finden Sie unter [HSM-Benutzer](#).

Der CO kann das Token nur für einen Vorgang nutzen. Wurde dieser erfolgreich ausgeführt, verliert das Token seine Gültigkeit. Um einen weiteren HSM-Benutzermanagement-Vorgang auszuführen, benötigt der CO ein neues Quorum-Token sowie neue Signaturen von den Genehmigenden. Zudem muss er das neue Token auf dem HSM genehmigen.

Note

Das MofN-Token ist nur gültig, solange Ihre aktuelle Anmeldesitzung geöffnet ist. Wenn Sie sich von `cloudhsm_mgmt_util` abmelden oder die Netzwerkverbindung unterbrochen wird, ist das Token nicht mehr gültig. Ebenso kann ein autorisiertes Token nur innerhalb von

cloudhsm_mgmt_util verwendet werden, es kann nicht zur Authentifizierung in einer anderen Anwendung verwendet werden.

Beim folgenden Beispielbefehl erstellt der CO einen neuen Benutzer auf dem HSM.

```
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
```

Nachdem der vorherige Befehl erfolgreich durchgeführt wurde, zeigt der nachfolgende Befehl listUsers den neuen Benutzer an.

```
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		

Wenn der CO versucht, einen weiteren HSM-Benutzermanagement-Vorgang durchzuführen, schlägt dieser aufgrund eines Quorum-Authentifizierungsfehlers fehl, wie im folgenden Beispiel gezeigt.

```
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
```

Der Befehl `listTokens` zeigt an, dass der CO nicht über genehmigte Tokens verfügt, wie im folgenden Beispiel gezeigt. Um einen weiteren HSM-Benutzermanagement-Vorgang auszuführen, benötigt der CO ein neues Quorum-Token sowie neue Signaturen von den Genehmigenden. Zudem muss er das neue Token auf dem HSM genehmigen.

```
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

```

Ändern Sie den Quorum-Mindestwert mit dem AWS CloudHSM Management Utility

Nachdem Sie [den Quorum-Mindestwert so festgelegt](#) haben, dass AWS CloudHSM [Crypto Officers \(COs\)](#) die Quorumauthentifizierung verwenden können, möchten Sie möglicherweise den Quorum-Mindestwert ändern. Das HSM lässt nur eine Änderung des Quorum-Mindestwerts zu, wenn die Anzahl der Genehmiger gleich oder höher als der aktuelle Quorum-Mindestwert ist. Wenn der Quorum-Mindestwert beispielsweise zwei ist, COs müssen mindestens zwei Personen die Änderung des Quorum-Mindestwerts genehmigen.

Für die Quorum-Genehmigung zum Ändern des Quorum-Mindestwerts benötigen Sie ein Quorum-Token für den Befehl `setMValue` (Service 4). Um ein Quorum-Token für den Befehl `setMValue` (Service 4) zu erhalten, muss der Quorum-Mindestwert für Service 4 größer als 1 sein. Das bedeutet, dass Sie, bevor Sie den Quorum-Mindestwert für COs (Dienst 3) ändern können, möglicherweise den Quorum-Mindestwert für Dienst 4 ändern müssen.

In der folgenden Tabelle sind die HSM-Dienstkennungen zusammen mit ihren Namen, Beschreibungen und den Befehlen aufgeführt, die im Dienst enthalten sind.

Dienstkennung	Service-Name	Service description (Service-Beschreibung)	HSM-Befehle
3	USER_MGMT	HSM-Benutzerverwaltung	• <code>createUser</code> • <code>deleteUser</code> • <code>changePswd</code> (gilt nur, wenn das Passwort eines

Dienstkennung	Service-Name	Service description (Service-Beschreibung)	HSM-Befehle
			anderen HSM-Benutzers geändert wird)
4	MISC_CO	Diverser CO-Service	• setMValue

So ändern Sie den Quorum-Mindestwert für Verschlüsselungsverantwortliche

1. Starten Sie das Befehlszeilen-Tool `cloudhsm_mgmt_util` mit folgendem Befehl.

```
$ /opt/cloudhsm/bin/cloudhsm_mgmt_util /opt/cloudhsm/etc/cloudhsm_mgmt_util.cfg
```

2. Verwenden Sie den Befehl `loginHSM`, um sich bei den HSMs als CO anzumelden. Weitere Informationen finden Sie unter [???](#).
3. Verwenden Sie den Befehl `getMValue`, um den Quorum-Mindestwert für Service 3 abzurufen. Weitere Informationen finden Sie im folgenden Beispiel.
4. Verwenden Sie den Befehl `getMValue`, um den Quorum-Mindestwert für Service 4 abzurufen. Weitere Informationen finden Sie im folgenden Beispiel.
5. Wenn der Quorum-Mindestwert für Service 4 niedriger ist als der Wert für Service 3, verwenden Sie den Befehl `setMValue`, um den Wert für Service 4 zu ändern. Ändern Sie den Wert für Service 4 in einen Wert, der gleich oder höher als der Wert für Service 3 ist. Weitere Informationen finden Sie im folgenden Beispiel.
6. [Rufen Sie ein Quorum-Token](#) ab und geben Sie Service 4 als den Service an, für den Sie das Token verwenden können.
7. [Holen Sie sich Genehmigungen \(Unterschriften\)](#) von anderen ein. COs
8. [Genehmigen des Tokens im HSM](#).
9. Verwenden Sie den `setMValue` Befehl, um den Quorum-Mindestwert für Dienst 3 zu ändern (Benutzerverwaltungsvorgänge, die von COs ausgeführt werden).

Example– Abrufen von Quorum-Mindestwerten und Ändern des Werts für Service 4

Das folgende Beispiel zeigt, dass der Quorum-Mindestwert für Service 3 derzeit 2 ist.

```
aws-cloudhsm > getMValue 3
MValue of service 3[USER_MGMT] on server 0 : [2]
MValue of service 3[USER_MGMT] on server 1 : [2]
```

Das folgende Beispiel zeigt, dass der Quorum-Mindestwert für Service 4 derzeit 1 ist.

```
aws-cloudhsm > getMValue 4
MValue of service 4[MISC_C0] on server 0 : [1]
MValue of service 4[MISC_C0] on server 1 : [1]
```

Um den Quorum-Mindestwert für Service 4 zu ändern, verwenden Sie den Befehl `setMValue` und legen Sie einen Wert fest, der gleich oder höher als der Wert für Service 3 ist. Im folgenden Beispiel wird der Quorum-Mindestwert für Service 4 auf 2 festgelegt. Dies ist der gleiche Wert wie für Service 3.

```
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
```

Die folgenden Befehle zeigen, dass der Quorum-Mindestwert für Service 3 und Service 4 jetzt 2 ist.

```
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_C0] on server 0 : [2]
MValue of service 4[MISC_C0] on server 1 : [2]
```

Schlüssel rein AWS CloudHSM

Bevor Sie Ihren AWS CloudHSM Cluster für die Kryptoverarbeitung verwenden können, müssen Sie [Benutzer](#) und Schlüssel für die Hardware-Sicherheitsmodule (HSM) in Ihrem Cluster erstellen.

Verwenden Sie einen der folgenden Befehle AWS CloudHSM, um die Schlüssel HSMs in Ihrem Cluster zu verwalten:

- PKCS-#11-Bibliothek
- JCE-Anbieter
- CNG- und KSP-Anbieter
- CloudHSM-CLI

Bevor Sie Schlüssel verwalten können, müssen Sie sich mit dem Benutzernamen und dem Passwort eines Crypto-Benutzers (CU) am HSM anmelden. Nur ein CU kann einen Schlüssel erstellen. Der CU, die einen Schlüssel erstellt, besitzt und verwaltet diesen Schlüssel.

In den folgenden Themen finden Sie weitere Informationen zur Verwaltung von Schlüsseln in AWS CloudHSM.

Themen

- [Wichtige Synchronisations- und Haltbarkeitseinstellungen in AWS CloudHSM](#)
- [AES-Schlüssel einpacken AWS CloudHSM](#)
- [Verwenden von vertrauenswürdigen Schlüsseln in AWS CloudHSM](#)
- [Schlüsselverwaltung mit CloudHSM CLI](#)
- [Schlüsselmanagement mit der AWS CloudHSM KMU](#)

Wichtige Synchronisations- und Haltbarkeitseinstellungen in AWS CloudHSM

AWS CloudHSM synchronisiert jeden Token-Schlüssel, den Sie erstellen. Die Schlüsselsynchronisierung ist größtenteils ein automatischer Prozess, aber Sie können mindestens zwei Hardware-Sicherheitsmodule (HSM) in Ihrem Cluster verwenden, um Schlüssel langlebiger zu machen. In diesem Thema werden Einstellungen für die Schlüsselsynchronisierung, allgemeine

Probleme, mit denen Kunden bei der Arbeit mit Schlüsseln in einem Cluster konfrontiert sind, sowie Strategien zur Erhöhung der Haltbarkeit von Schlüsseln beschrieben.

In diesem Thema werden die wichtigsten Synchronisierungseinstellungen unter AWS CloudHSM, allgemeine Probleme, mit denen Kunden bei der Arbeit mit Schlüsseln in einem Cluster konfrontiert sind, sowie Strategien zur Erhöhung der Haltbarkeit von Schlüsseln beschrieben.

Themen

- [AWS CloudHSM wichtige Konzepte](#)
- [Grundlegendes AWS CloudHSM zur Schlüsselsynchronisierung](#)
- [Ändern Sie die Einstellungen für die Haltbarkeit von AWS CloudHSM Client-Schlüsseln](#)
- [Synchronisieren von Schlüsseln zwischen geklonten Clustern AWS CloudHSM](#)

AWS CloudHSM wichtige Konzepte

Im Folgenden finden Sie Konzepte, die Sie bei der Arbeit mit Tasten beachten sollten AWS CloudHSM.

Token-Schlüssel

Persistente Schlüssel, die Sie beim Generieren, Importieren oder Entpacken von Schlüsseln erstellen. AWS CloudHSM synchronisiert Token-Schlüssel in einem Cluster.

Sitzungsschlüssel

Kurzlebige Schlüssel, die nur auf einem Hardware-Sicherheitsmodul (HSM) im Cluster vorhanden sind. AWS CloudHSM synchronisiert keine Sitzungsschlüssel in einem Cluster.

Clientseitige Schlüsselsynchronisierung

Ein clientseitiger Prozess, der Tokenschlüssel klonet, die Sie beim Generieren, Importieren oder Entpacken von Schlüsseln erstellen. Sie können die Haltbarkeit von Tokenschlüsseln erhöhen, indem Sie einen Cluster mit mindestens zwei Schlüsseln ausführen HSMs.

Serverseitige Schlüsselsynchronisierung

Klont in regelmäßigen Abständen Schlüssel für jedes HSM im Cluster. Erfordert keine Verwaltung.

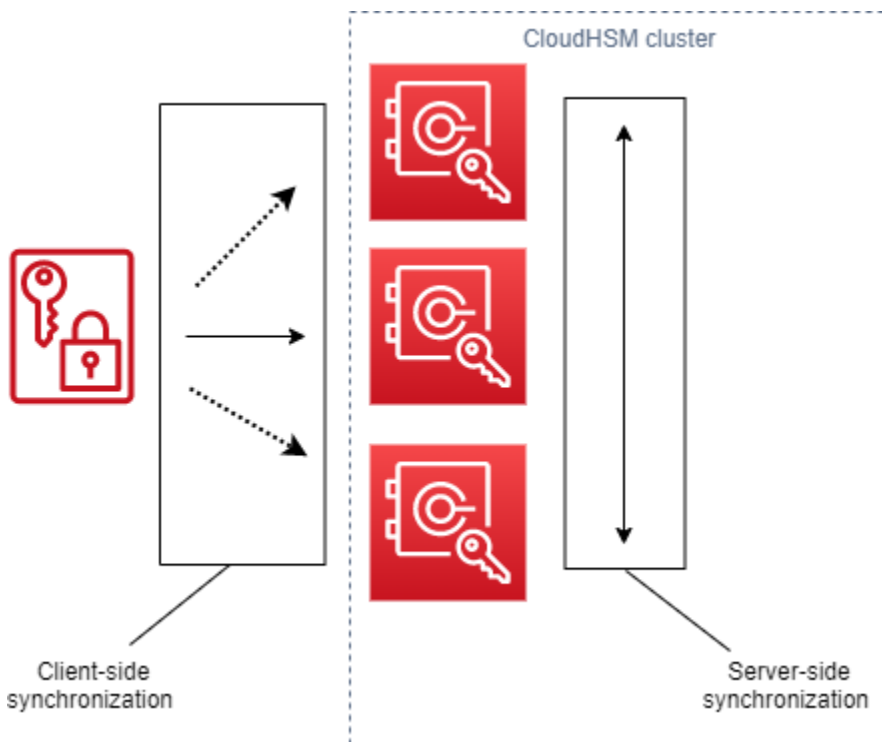
Einstellungen für die Haltbarkeit von Client-Schlüsseln

Einstellungen, die Sie auf dem Client konfigurieren und die sich auf die Haltbarkeit von Schlüsseln auswirken. Diese Einstellungen funktionieren in Client SDK 5 und Client SDK 3 unterschiedlich.

- Verwenden Sie im Client-SDK 5 diese Einstellung, um einen einzelnen HSM-Cluster auszuführen.
- Verwenden Sie im Client-SDK 3 diese Einstellung, um die Anzahl von Vorgängen anzugeben, die für eine erfolgreiche Schlüsselerstellung HSMs erforderlich sind.

Grundlegendes AWS CloudHSM zur Schlüsselsynchronisierung

AWS CloudHSM verwendet die Schlüsselsynchronisierung, um Tokenschlüssel für alle Hardware-Sicherheitsmodule (HSM) in einem Cluster zu klonen. Sie erstellen Token-Schlüssel als persistente Schlüssel bei der Schlüsselgenerierung, beim Import oder beim Entpacken. Um diese Schlüssel über den Cluster zu verteilen, bietet CloudHSM die clientseitige und serverseitige Schlüssel-Synchronisierung.



Das Ziel der Schlüsselsynchronisierung – sowohl serverseitig als auch clientseitig – besteht darin, neue Schlüssel so schnell wie möglich im Cluster zu verteilen, nachdem Sie sie erstellt haben. Dies ist wichtig, da die nachfolgenden Aufrufe, die Sie zur Verwendung neuer Schlüssel tätigen, an jedes verfügbare HSM im Cluster weitergeleitet werden können. Wenn der Aufruf, den Sie tätigen, ohne den Schlüssel an ein HSM weitergeleitet wird, schlägt der Aufruf fehl. Sie können diese Art von Fehlern beheben, indem Sie angeben, dass Ihre Anwendungen nachfolgende Aufrufe wiederholen, die nach der Schlüsselerstellung ausgeführt wurden. Die für die Synchronisierung benötigte Zeit kann je nach der Arbeitslast Ihres Clusters und anderen immateriellen Vermögenswerten variieren. Verwenden Sie

CloudWatch Metriken, um das Timing zu bestimmen, das Ihre Anwendung in einer solchen Situation verwenden sollte. Weitere Informationen finden Sie unter [CloudWatch -Metriken](#).

Die Herausforderung bei der Schlüsselsynchronisierung in einer Cloud-Umgebung ist die Haltbarkeit der Schlüssel. Sie erstellen Schlüssel auf einem einzigen HSM und beginnen oft sofort, diese Schlüssel zu verwenden. Wenn das HSM, auf dem Sie Schlüssel erstellen, ausfällt, bevor die Schlüssel auf ein anderes HSM im Cluster geklont wurden, verlieren Sie die Schlüssel und den Zugriff auf alles, was mit den Schlüsseln verschlüsselt wurde. Um dieses Risiko zu minimieren, bieten wir Client-Synchronisierung an. Die clientseitige Synchronisation ist ein clientseitiger Prozess, bei dem die Schlüssel geklont werden, die Sie beim Generieren, Importieren oder Entpacken von Schlüsseln erstellen. Durch das Klonen von Schlüsseln während der Erstellung werden Schlüssel langlebiger. Natürlich können Sie Schlüssel in einem Cluster nicht mit einem einzigen HSM klonen. Um die Lebensdauer von Schlüsseln zu erhöhen, empfehlen wir Ihnen außerdem, Ihren Cluster so zu konfigurieren, dass mindestens zwei Schlüssel verwendet werden. Mit der clientseitigen Synchronisation und einem Cluster mit zwei HSMs können Sie die Herausforderung der Schlüsselbeständigkeit in einer Cloud-Umgebung bewältigen.

Ändern Sie die Einstellungen für die Haltbarkeit von AWS CloudHSM Client-Schlüsseln

Die Schlüsselsynchronisierung ist größtenteils ein automatischer Prozess, aber Sie können die Einstellungen für die Haltbarkeit von Schlüsseln auf der Clientseite verwalten. Die clientseitigen Einstellungen für die Haltbarkeit von Schlüsseln funktionieren in Client-SDK 5 und Client-SDK 3 unterschiedlich.

- In Client SDK 5 führen wir das Konzept der Schlüsselverfügbarkeitsquoren ein, bei dem Sie Cluster mit mindestens zwei Quoren ausführen müssen. HSMs Sie können die clientseitigen Einstellungen für die Haltbarkeit von Schlüsseln verwenden, um die beiden HSM-Anforderungen abzulehnen. Weitere Informationen zu Quora finden Sie unter [the section called “Konzepte”](#).
- Im Client-SDK 3 geben Sie mithilfe der clientseitigen Einstellungen für die Haltbarkeit von Schlüsseln die Anzahl der HSMs Schlüssel an, für die die Schlüsselerstellung erfolgreich sein muss, damit der gesamte Vorgang als erfolgreich gewertet wird.

Client-SDK 5: Einstellungen für die Haltbarkeit von Client-Schlüsseln

In Client-SDK 5 ist die Schlüsselsynchronisierung ein vollautomatischer Prozess. Beim Quorum für die Schlüsselverfügbarkeit müssen neu erstellte Schlüssel auf zwei HSMs im Cluster

vorhanden sein, bevor Ihre Anwendung den Schlüssel verwenden kann. Um das Quorum für die Schlüsselverfügbarkeit verwenden zu können, muss Ihr Cluster über mindestens zwei verfügbare HSMs verfügen.

Wenn Ihre Clusterkonfiguration die Anforderungen an die Haltbarkeit von Schlüsseln nicht erfüllt, schlägt jeder Versuch, einen Token-Schlüssel zu erstellen oder zu verwenden, fehl und die folgende Fehlermeldung wird in den Protokollen angezeigt:

```
Key <key handle> does not meet the availability requirements - The key must be available on at least 2 HSMs before being used.
```

Sie können die Client-Konfigurationseinstellungen verwenden, um das Quorum für die Schlüsselverfügbarkeit zu deaktivieren. Möglicherweise möchten Sie sich abmelden, um beispielsweise einen Cluster mit einem einzigen HSM auszuführen.

Konzepte des Client-SDK 5

Quorum für Schlüsselverfügbarkeit

AWS CloudHSM gibt die Anzahl der HSMs in einem Cluster vorhandenen Schlüssel an, bevor Ihre Anwendung den Schlüssel verwenden kann. Erfordert Cluster mit mindestens zwei HSMs.

Verwaltung der Einstellungen für die Haltbarkeit von Client-Schlüsseln

Um die Einstellungen für die Haltbarkeit von Client-Schlüsseln zu verwalten, müssen Sie das Configure-Tool für Client-SDK 5 verwenden.

PKCS #11 library

Um die Haltbarkeit von Client-Schlüsseln für Client-SDK 5 unter Linux zu deaktivieren

- Verwenden Sie das Configure-Tool, um die Einstellungen für die Haltbarkeit von Client-Schlüsseln zu deaktivieren.

```
$ sudo /opt/cloudhsm/bin/configure-pkcs11 --disable-key-availability-check
```

Um die Haltbarkeit von Client-Schlüsseln für Client-SDK 5 unter Windows zu deaktivieren

- Verwenden Sie das Configure-Tool, um die Einstellungen für die Haltbarkeit von Client-Schlüsseln zu deaktivieren.

```
PS C:\> & "C:\Program Files\Amazon\CloudHSM\bin\configure-pkcs11.exe" --disable-key-availability-check
```

OpenSSL Dynamic Engine

Um die Haltbarkeit von Client-Schlüsseln für Client-SDK 5 unter Linux zu deaktivieren

- Verwenden Sie das Configure-Tool, um die Einstellungen für die Haltbarkeit von Client-Schlüsseln zu deaktivieren.

```
$ sudo /opt/cloudhsm/bin/configure-dyn --disable-key-availability-check
```

OpenSSL Dynamic Engine Provider

Um die Haltbarkeit von Client-Schlüsseln für Client-SDK 5 unter Linux zu deaktivieren

- Verwenden Sie das Configure-Tool, um die Einstellungen für die Haltbarkeit von Client-Schlüsseln zu deaktivieren.

```
$ sudo /opt/cloudhsm/bin/configure-openssl-provider --disable-key-availability-check
```

Key Storage Provider (KSP)

Um die Haltbarkeit von Client-Schlüsseln für Client-SDK 5 unter Windows zu deaktivieren

- Verwenden Sie das Configure-Tool, um die Einstellungen für die Haltbarkeit von Client-Schlüsseln zu deaktivieren.

```
PS C:\> & "C:\Program Files\Amazon\CloudHSM\bin\configure-ksp.exe" --disable-key-availability-check
```


JCE provider

Um die Haltbarkeit von Client-Schlüsseln für Client-SDK 5 unter Linux zu deaktivieren

- Verwenden Sie das Configure-Tool, um die Einstellungen für die Haltbarkeit von Client-Schlüsseln zu deaktivieren.

```
$ sudo /opt/cloudhsm/bin/configure-jce --disable-key-availability-check
```

Um die Haltbarkeit von Client-Schlüsseln für Client-SDK 5 unter Windows zu deaktivieren

- Verwenden Sie das Configure-Tool, um die Einstellungen für die Haltbarkeit von Client-Schlüsseln zu deaktivieren.

```
PS C:\> & "C:\Program Files\Amazon\CloudHSM\bin\configure-jce.exe" --disable-key-availability-check
```

CloudHSM CLI

Um die Haltbarkeit von Client-Schlüsseln für Client-SDK 5 unter Linux zu deaktivieren

- Verwenden Sie das Configure-Tool, um die Einstellungen für die Haltbarkeit von Client-Schlüsseln zu deaktivieren.

```
$ sudo /opt/cloudhsm/bin/configure-cli --disable-key-availability-check
```

Um die Haltbarkeit von Client-Schlüsseln für Client-SDK 5 unter Windows zu deaktivieren

- Verwenden Sie das Configure-Tool, um die Einstellungen für die Haltbarkeit von Client-Schlüsseln zu deaktivieren.

```
PS C:\> & "C:\Program Files\Amazon\CloudHSM\bin\configure-cli.exe" --disable-key-availability-check
```

Client-SDK 3: Einstellungen für die Haltbarkeit von Client-Schlüsseln

In Client-SDK 3 ist die Schlüsselsynchronisierung größtenteils ein automatischer Prozess, aber Sie können die Einstellungen für die Haltbarkeit von Schlüsseln des Clients verwenden, um die Haltbarkeit von Schlüsseln zu erhöhen. Sie geben die Anzahl HSMs an, bei der die Schlüsselerstellung erfolgreich sein muss, damit der gesamte Vorgang als erfolgreich gewertet wird. Bei der clientseitigen Synchronisation wird stets versucht, Schlüssel für jedes HSM im Cluster zu klonen, unabhängig davon, welche Einstellung Sie wählen. Ihre Einstellung erzwingt die Schlüsselerstellung anhand der von HSMs Ihnen angegebenen Anzahl. Wenn Sie einen Wert angeben und das System den Schlüssel nicht auf diese Anzahl von replizieren kann HSMs, entfernt das System automatisch unerwünschtes Schlüsselmateriale, und Sie können es erneut versuchen.

Important

Wenn Sie keine Einstellungen für die Haltbarkeit von Client-Schlüsseln festlegen (oder den Standardwert 1 verwenden), besteht die Gefahr, dass Ihre Schlüssel verloren gehen. Wenn Ihr aktuelles HSM ausfällt, bevor der serverseitige Dienst diesen Schlüssel auf ein anderes HSM geklont hat, verlieren Sie das Schlüsselmateriale.

Um die Haltbarkeit der Schlüssel zu maximieren, sollten Sie HSMs für die clientseitige Synchronisation mindestens zwei angeben. Denken Sie daran, dass die Arbeitslast auf Ihrem Cluster unabhängig davon, wie viele HSMs Sie angeben, dieselbe bleibt. Bei der clientseitigen Synchronisation wird stets nach besten Kräften versucht, Schlüssel für jedes HSM im Cluster zu klonen.

Empfehlungen

- Minimum: Zwei HSMs pro Cluster
- Maximum: Eins weniger als die Gesamtzahl der HSMs in Ihrem Cluster

Wenn die clientseitige Synchronisation fehlschlägt, bereinigt der Client-Dienst alle unerwünschten Schlüssel, die möglicherweise erstellt wurden und nun unerwünscht sind. Bei dieser Bereinigung handelt es sich um eine bestmögliche Lösung, die möglicherweise nicht immer funktioniert. Wenn die Bereinigung fehlschlägt, müssen Sie möglicherweise unerwünschtes Schlüsselmateriale löschen. Weitere Informationen finden Sie unter [Schlüssel-Synchronisierungsfehler](#).

Einrichtung der Konfigurationsdatei für die Haltbarkeit von Client-Schlüsseln

Um die Einstellungen für die Haltbarkeit von Client-Schlüsseln festzulegen, müssen Sie `cloudhsm_client.cfg` bearbeiten.

Bearbeiten der Clientkonfigurationsdatei

1. Öffnen Sie `cloudhsm_client.cfg`.

Linux:

```
/opt/cloudhsm/etc/cloudhsm_client.cfg
```

Windows:

```
C:\ProgramData\Amazon\CloudHSM\data\cloudhsm_client.cfg
```

2. Fügen Sie im `client` Knoten der Datei einen Wert hinzu `create_object_minimum_nodes` und geben Sie einen Wert für die Mindestanzahl HSMs an, für die Schlüssel erfolgreich erstellt AWS CloudHSM werden müssen, damit die Schlüsselerstellung erfolgreich ist.

```
"create_object_minimum_nodes" : 2
```

Note

Das Befehlszeilentool `key_mgmt_util` (KMU) bietet eine zusätzliche Einstellung für die Haltbarkeit von Client-Schlüsseln. Weitere Informationen finden Sie unter [the section called "KMU- und clientseitige Schlüsselsynchronisierung"](#).

Konfigurationsreferenz

Dies sind die clientseitigen Synchronisierungseigenschaften, die in einem Auszug aus der `cloudhsm_client.cfg`:

```
{
  "client": {
    "create_object_minimum_nodes" : 2,
    ...
  },
}
```

```
    ...  
}
```

`create_object_minimum_nodes`

Gibt die Mindestanzahl von Vorgängen an, die HSMs erforderlich sind, um die Schlüsselgenerierung, den Schlüsselimport oder das Entpacken von Schlüsseln als erfolgreich zu betrachten. Falls eingestellt, ist die Voreinstellung 1. Das bedeutet, dass der clientseitige Dienst bei jedem Vorgang zur Schlüsselerstellung versucht, Schlüssel für jedes HSM im Cluster zu erstellen. Um jedoch einen Erfolg zurückzugeben, muss nur ein einziger Schlüssel auf einem HSM im Cluster erstellt werden.

KMU- und clientseitige Schlüsselsynchronisierung

Wenn Sie Schlüssel mit dem Befehlszeilentool `key_mgmt_util` (KMU) erstellen, verwenden Sie einen optionalen Befehlszeilenparameter (`-min_srv`), um die Anzahl der zu klonenden Schlüssel zu begrenzen. HSMs Wenn Sie den Befehlszeilenparameter und einen Wert in der Konfigurationsdatei angeben, wird der GRÖßERE der beiden Werte berücksichtigt. AWS CloudHSM

Weitere Informationen finden Sie unter den folgenden Themen:

- [Gen-Paar DSAKey](#)
- [ECCKeyGen-Paar](#)
- [RSAKeyGen-Paar](#)
- [genSymKey](#)
- [importPrivateKey](#)
- [importPubKey](#)
- [imSymKey](#)
- [insertMaskedObject](#)
- [unWrapKey](#)

Synchronisieren von Schlüsseln zwischen geklonten Clustern AWS CloudHSM

Die clientseitige und serverseitige Synchronisation dient nur der Synchronisierung von Schlüsseln innerhalb desselben Clusters. AWS CloudHSM Wenn Sie ein Backup eines Clusters in eine andere Region kopieren, verwenden Sie den Befehl [key replicate, um einen Schlüssel zwischen zwei Clustern zu replizieren](#). Sie können geklonte Cluster für regionsübergreifende Redundanz oder zur Vereinfachung Ihres Disaster Recovery-Prozesses verwenden. Wenn Sie CloudHSM CLI nicht installiert haben, lesen Sie die Anweisungen unter. [???](#)

AES-Schlüssel einpacken AWS CloudHSM

In diesem Thema werden die Optionen für das Umschließen von AES-Schlüsseln beschrieben AWS CloudHSM. Die AES Key Wrapping verwendet einen AES-Schlüssel (den Verschlüsselungsschlüssel), um einen anderen Schlüssel eines beliebigen Typs (den Zielschlüssel) zu verschlüsseln. Sie verwenden die Verschlüsselung, um gespeicherte Schlüssel zu schützen oder Schlüssel über unsichere Netzwerke zu übermitteln.

Themen

- [Unterstützte Algorithmen](#)
- [Verwenden Sie den AES-Schlüsselumbruch AWS CloudHSM](#)

Unterstützte Algorithmen

AWS CloudHSM bietet drei Optionen für das Umschließen von AES-Schlüsseln, die jeweils darauf basieren, wie der Zielschlüssel vor dem Umbruch aufgefüllt wird. Das Padding erfolgt automatisch in Übereinstimmung mit dem von Ihnen verwendeten Algorithmus, wenn Sie die Verschlüsselung aufrufen. In der folgenden Tabelle sind die unterstützten Algorithmen und die zugehörigen Details aufgeführt, um Ihnen bei der Auswahl eines geeigneten Verschlüsselungsmechanismus für Ihre Anwendung zu helfen.

AES-Verschlüsselungsalgorithmus	Spezifikation	Unterstützte Zielschlüsseltypen	Padding-Schema	AWS CloudHSM Verfügbarkeit des Clients
AES-Verschlüsselung mit Zero Padding	RFC 5649 und SP 800—38F	Alle	Fügt zur Blockausrichtung Schlüssel-Bits nach Bedarf Nullen hinzu	SDK 3.1 und höher
AES-Verschlüsselung ohne Padding	RFC 3394 und SP 800—38F	Blockausgerichtete Schlüssel wie AES und 3DES	Keine	SDK 3.1 und höher
AES-Verschlüsselung mit PKCS #5 Padding	Keine	Alle	Mindestens 8 Bytes werden gemäß PKCS #5 Padding-Schema zur Blockausrichtung hinzugefügt	Alle

Weitere Informationen zur Verwendung der AES-Verschlüsselungsalgorithmen aus der vorangegangenen Tabelle in Ihrer Anwendung finden Sie unter [Verwenden von AES Key Wrap in AWS CloudHSM](#).

Grundlegendes zu Initialisierungsvektoren in der AES-Verschlüsselung

Vor dem Verschlüsseln hängt CloudHSM einen Initialisierungsvektor (IV) an den Zielschlüssel für die Datenintegrität an. Jeder Verschlüsselungsalgorithmus weist spezifische Einschränkungen auf, welcher IV-Typ zulässig ist. Um die IV einzustellen, haben Sie zwei Möglichkeiten AWS CloudHSM:

- Implizit: Setzen Sie den IV auf NULL und CloudHSM verwendet für diesen Algorithmus den Standardwert für Ver- und Entschlüsselungsvorgänge (empfohlen)
- Explizit: Legen Sie den IV fest, indem Sie den Standard-IV-Wert an die Verschlüsselungsfunktion übergeben.

Important

Sie müssen wissen, welchen IV Sie in Ihrer Anwendung verwenden. Um den Schlüssel zu entschlüsseln, müssen Sie denselben IV angeben, den Sie zum Verschlüsseln des Schlüssels verwendet haben. Wenn Sie einen impliziten IV zum Verschlüsseln verwenden, wenden Sie zum Entschlüsseln einen impliziten IV an. Bei einem impliziten IV verwendet CloudHSM zum Entschlüsseln den Standardwert.

In der folgenden Tabelle werden die zulässigen Werte beschrieben IVs, für die der Wrapping-Algorithmus festlegt.

AES-Verschlüsselungsalgorithmus	Implizite IV	Expliziter IV
AES-Verschlüsselung mit Zero Padding	Erforderlich Standardwert: (intern berechneter IV basierend auf der Spezifikation)	Nicht zulässig
AES-Verschlüsselung ohne Padding	Zulässig (empfohlen) Standardwert: 0xA6A6A6A6A6A6A6A6	Zulässig Nur dieser Wert akzeptiert: 0xA6A6A6A6A6A6A6A6
AES-Verschlüsselung mit PKCS #5 Padding	Zulässig (empfohlen) Standardwert: 0xA6A6A6A6A6A6A6A6	Zulässig Nur dieser Wert akzeptiert: 0xA6A6A6A6A6A6A6A6

Verwenden Sie den AES-Schlüsselumbruch AWS CloudHSM

Sie packen und entpacken Schlüssel wie folgt:

- Wählen Sie wie in der folgenden Tabelle dargestellt in der [PCS #11-Bibliothek](#) den entsprechenden Mechanismus für die Funktionen C_WrapKey und C_UnWrapKey aus.

- Wählen Sie wie in der folgenden Tabelle dargestellt im [JCE-Anbieter](#) den entsprechenden Algorithmus, den Modus und die Padding-Kombination zur Implementierung der Cipher-Methoden `Cipher.WRAP_MODE` und `Cipher.UNWRAP_MODE` aus.
- Wählen Sie in der [CloudHSM-CLI](#) den entsprechenden Algorithmus aus der Liste der unterstützten [Der Befehl zum Entpacken von Schlüsseln in der CloudHSM-CLI](#) Algorithmen [Der Key Wrap-Befehl in der CloudHSM-CLI](#) und Algorithmen aus, wie in der folgenden Tabelle dargestellt.
- Verwenden Sie wie in der folgenden Tabelle dargestellt in [key_mgmt_util \(KMU\)](#) die Befehle [Exportieren Sie einen AWS CloudHSM Schlüssel mit KMU](#) und [Entpacken Sie einen AWS CloudHSM Schlüssel mit KMU](#) mit entsprechenden m-Werten.

AES-Verschlüsselungsalgorithmus	PKCS #11-Mechanismus	Java-Methode	CloudHSM CLI-Unterbefehl	Argument des Key Management Utility (KMU)
AES-Verschlüsselung mit Zero Padding	CKM_CLOUD_HSM_AES_KEY_WRAP_ZERO_PAD (Anbieter definierter Mechanismus)	AESWrap/ECB/ZeroPadding	aes-zero-pad	m = 6
AES-Verschlüsselung ohne Padding	CKM_CLOUD_HSM_AES_KEY_WRAP_NO_PAD (Anbieter definierter Mechanismus)	AESWrap/ECB/NoPadding	aes-no-pad	m = 5
AES-Verschlüsselung mit PKCS #5 Padding	CKM_CLOUD_HSM_AES_KEY_WRAP_PKCS5_PAD (Anbieter	AESWrap/ECB/PKCS5Padding	aes-pkcs5-pad	m = 4

AES-Verschlüsselungsalgorithmus	PKCS #11-Mechanismus	Java-Methode	CloudHSM CLI-Unterbefehl	Argument des Key Management Utility (KMU)
	definierter Mechanismus)			

Verwenden von vertrauenswürdigen Schlüsseln in AWS CloudHSM

AWS CloudHSM unterstützt Trusted Key Wrapping, um Datenschlüssel vor Insider-Bedrohungen zu schützen. In diesem Thema wird beschrieben, wie Vertrauenswürdige Schlüssel zum Schutz von Daten erstellt werden.

Themen

- [Informationen zu vertrauenswürdigen Schlüsseln finden Sie in AWS CloudHSM](#)
- [Vertrauenswürdige Schlüsselattribute in AWS CloudHSM](#)
- [Wie verwendet man vertrauenswürdige Schlüssel, um Datenschlüssel einzuschließen AWS CloudHSM](#)
- [Wie entpacke ich einen Datenschlüssel mit einem vertrauenswürdigen Schlüssel für AWS CloudHSM](#)

Informationen zu vertrauenswürdigen Schlüsseln finden Sie in AWS CloudHSM

Ein vertrauenswürdiger Schlüssel ist ein Schlüssel, der zum Umschließen anderer Schlüssel verwendet wird und den Administratoren und Kryptografiebeauftragte (COs) anhand des Attributs ausdrücklich als vertrauenswürdige identifizieren. CKA_TRUSTED Darüber hinaus geben Administratoren und Kryptografiebeauftragte (COs) anhand CKA_UNWRAP_TEMPLATE verwandter Attribute an, welche Aktionen Datenschlüssel ausführen können, sobald sie durch einen vertrauenswürdigen Schlüssel entpackt wurden. Datenschlüssel, die durch den vertrauenswürdigen Schlüssel entpackt wurden, müssen auch diese Attribute enthalten, damit der Entpackvorgang erfolgreich ist. Dadurch wird sichergestellt, dass entpackte Datenschlüssel nur für die von Ihnen beabsichtigte Verwendung zulässig sind.

Verwenden Sie das Attribut `CKA_WRAP_WITH_TRUSTED`, um alle Datenschlüssel zu identifizieren, die Sie mit vertrauenswürdigen Schlüsseln umschließen möchten. Auf diese Weise können Sie Datenschlüssel einschränken, sodass Anwendungen nur vertrauenswürdige Schlüssel verwenden können, um sie zu entpacken. Sobald Sie dieses Attribut für die Datenschlüssel festgelegt haben, ist das Attribut schreibgeschützt und kann nicht mehr geändert werden. Wenn diese Attribute vorhanden sind, können Anwendungen Ihre Datenschlüssel nur mit den Schlüsseln entpacken, denen Sie vertrauen, und Entpackungen führen immer zu Datenschlüsseln mit Attributen, die einschränken, wie diese Schlüssel verwendet werden können.

Vertrauenswürdige Schlüsselattribute in AWS CloudHSM

Mit den folgenden Attributen können Sie einen AWS CloudHSM Schlüssel als vertrauenswürdige markieren, angeben, dass ein Datenschlüssel nur mit einem vertrauenswürdigen Schlüssel ein- und ausgepackt werden kann, und steuern, was ein Datenschlüssel nach dem Entpacken tun kann:

- **CKA_TRUSTED:** Wenden Sie dieses Attribut (zusätzlich zu `CKA_UNWRAP_TEMPLATE`) auf den Schlüssel an, der Datenschlüssel umhüllt, um anzugeben, dass ein Administrator oder Crypto Officer (CO) die erforderliche Sorgfalt walten lässt und diesem Schlüssel vertraut. Nur ein Administrator oder CO kann `CKA_TRUSTED` einstellen. Der Crypto-Benutzer (CU) besitzt den Schlüssel, aber nur ein CO kann sein `CKA_TRUSTED`-Attribut festlegen.
- **CKA_WRAP_WITH_TRUSTED:** Wenden Sie dieses Attribut auf einen exportierbaren Datenschlüssel an, um anzugeben, dass Sie diesen Schlüssel nur mit Schlüsseln umschließen können, die als `CKA_TRUSTED` markiert sind. Sobald Sie `CKA_WRAP_WITH_TRUSTED` auf „true“ setzen, wird das Attribut schreibgeschützt und Sie können das Attribut nicht mehr ändern oder entfernen.
- **CKA_UNWRAP_TEMPLATE:** Wenden Sie dieses Attribut (zusätzlich zu `CKA_TRUSTED`) auf den Wrapping-Schlüssel an, um anzugeben, welche Attributnamen und -werte der Service automatisch auf Datenschlüssel anwenden muss, die der Dienst entpackt. Wenn eine Anwendung einen Schlüssel zum Entpacken einreicht, kann sie auch ihre eigene Entpackungsvorlage bereitstellen. Wenn Sie eine Unwrap-Vorlage angeben und die Anwendung eine eigene Unwrap-Vorlage bereitstellt, verwendet das HSM beide Vorlagen, um Attributnamen und -werte auf den Schlüssel anzuwenden. Wenn jedoch ein Wert in der `CKA_UNWRAP_TEMPLATE` für den Wrapping-Schlüssel mit einem Attribut kollidiert, das von der Anwendung während der Unwrap-Anfrage bereitgestellt wird, schlägt die Unwrap-Anfrage fehl.

Weitere Informationen über Attribute finden Sie in den folgenden Themen:

- [PKCS #11-Schlüsselattribute](#)

- [JCE-Schlüsselattribute](#)
- [CloudHSM-CLI-Schlüsselattribute](#)

Wie verwendet man vertrauenswürdige Schlüssel, um Datenschlüssel einzuschließen AWS CloudHSM

Um einen vertrauenswürdigen Schlüssel zum Einbinden eines Datenschlüssels zu verwenden AWS CloudHSM, müssen Sie drei grundlegende Schritte ausführen:

1. Für den Datenschlüssel, den Sie mit einem vertrauenswürdigen Schlüssel umschließen möchten, setzen Sie dessen `CKA_WRAP_WITH_TRUSTED`-Attribut auf „true“.
2. Setzen Sie für den vertrauenswürdigen Schlüssel, mit dem Sie den Datenschlüssel umschließen möchten, dessen `CKA_TRUSTED`-Attribut auf „true“.
3. Verwenden Sie den vertrauenswürdigen Schlüssel, um den Datenschlüssel zu umschließen.

Schritt 1: Das Datenschlüsselattribut **CKA_WRAP_WITH_TRUSTED** auf „true“ setzen

Wählen Sie für den Datenschlüssel, den Sie umschließen möchten, eine der folgenden Optionen, um das `CKA_WRAP_WITH_TRUSTED`-Schlüsselattribut auf „true“ zu setzen. Dadurch wird der Datenschlüssel eingeschränkt, so dass Anwendungen nur vertrauenswürdige Schlüssel verwenden können, um ihn zu verschlüsseln.

Option 1: Wenn Sie einen neuen Schlüssel generieren, setzen Sie **CKA_WRAP_WITH_TRUSTED** auf „true“

Generieren Sie einen Schlüssel mit [PKCS #11](#), [JCE](#) oder [CloudHSM-CLI](#). Weitere Einzelheiten finden Sie in den folgenden Beispielen.

PKCS #11

Um einen Schlüssel mit PKCS #11 zu generieren, müssen Sie das `CKA_WRAP_WITH_TRUSTED`-Attribut des Schlüssels auf „true“ setzen. Wie im folgenden Beispiel gezeigt, tun Sie dies, indem Sie dieses Attribut in die `CK_ATTRIBUTE` `template` des Schlüssels aufnehmen und das Attribut dann auf „true“ setzen:

```
CK_BYTE_PTR label = "test_key";
CK_ATTRIBUTE template[] = {
```

```
        {CKA_WRAP_WITH_TRUSTED, &true_val,      sizeof(CK_BBOOL)},  
        {CKA_LABEL,              label,          strlen(label)},  
        ...  
};
```

Weitere Informationen finden Sie in [unseren öffentlichen Beispielen, die die Schlüsselerzeugung mit PKCS #11 demonstrieren](#).

JCE

Um einen Schlüssel mit JCE zu generieren, müssen Sie das `WRAP_WITH_TRUSTED`-Attribut des Schlüssels auf „true“ setzen. Wie im folgenden Beispiel gezeigt, tun Sie dies, indem Sie dieses Attribut in die `KeyAttributesMap` des Schlüssels aufnehmen und das Attribut dann auf „true“ setzen:

```
final String label = "test_key";  
final KeyAttributesMap keySpec = new KeyAttributesMap();  
keySpec.put(KeyAttribute.WRAP_WITH_TRUSTED, true);  
keySpec.put(KeyAttribute.LABEL, label);  
...
```

Weitere Informationen finden Sie in [unseren öffentlichen Beispielen, die die Schlüsselerzeugung mit JCE demonstrieren](#).

CloudHSM CLI

Um einen Schlüssel mit CloudHSM CLI zu generieren, müssen Sie das `wrap-with-trusted`-Attribut des Schlüssels auf „true“ setzen. Fügen Sie dazu `wrap-with-trusted=true` in das entsprechende Argument für den Befehl zur Schlüsselerzeugung ein:

- Bei symmetrischen Schlüsseln fügen Sie `wrap-with-trusted` zum `attributes`-Argument hinzu.
- Bei öffentlichen Schlüsseln fügen Sie `wrap-with-trusted` zum `public-attributes`-Argument hinzu.
- Bei privaten Schlüsseln fügen Sie `wrap-with-trusted` zum `private-attributes`-Argument hinzu.

Weitere Informationen zur Generierung von Schlüsselpaaren finden Sie unter [Die generate-asymmetric-pair Kategorie in CloudHSM CLI](#).

Weitere Informationen zur Generierung von symmetrischen Schlüsseln finden Sie unter [Die Kategorie „Symmetrisch generieren“ in CloudHSM CLI](#).

Option 2: Wenn Sie einen vorhandenen Schlüssel verwenden, verwenden Sie die CloudHSM CLI, um **CKA_WRAP_WITH_TRUSTED** auf „true“ zu setzen

Gehen Sie wie folgt vor, um das CKA_WRAP_WITH_TRUSTED-Attribut eines vorhandenen Schlüssels auf „true“ zu setzen:

1. Verwenden Sie den [Melden Sie sich mit der CloudHSM-CLI bei einem HSM an](#)-Befehl, um sich als Crypto-Benutzer (CU) anzumelden.
2. Verwenden Sie den [Legen Sie die Attribute von Schlüsseln mit der CloudHSM-CLI fest](#)-Befehl, um das wrap-with-trusted-Schlüsselattribut auf „true“ zu setzen.

```
aws-cloudhsm > key set-attribute --filter attr.label=test_key --name wrap-with-trusted --value true
{
  "error_code": 0,
  "data": {
    "message": "Attribute set successfully"
  }
}
```

Schritt 2: Festlegen von **CKA_TRUSTED** des vertrauenswürdigen Schlüssels auf „true“

Um einen Schlüssel zu einem vertrauenswürdigen Schlüssel zu machen, muss sein CKA_TRUSTED-Attribut auf „true“ gesetzt werden. Sie können dazu entweder CloudHSM-CLI oder das CloudHSM Management Utility (CMU) verwenden.

- Wenn Sie die CloudHSM-CLI verwenden, um das CKA_TRUSTED-Attribut eines Schlüssels festzulegen, finden Sie weitere Informationen unter [Markieren Sie einen Schlüssel mithilfe der CloudHSM-CLI als vertrauenswürdig](#).
- Wenn Sie die CMU verwenden, um das CKA_TRUSTED-Attribut eines Schlüssels festzulegen, finden Sie weitere Informationen unter [So markieren Sie einen Schlüssel mit dem AWS CloudHSM Management Utility als vertrauenswürdig](#).

Schritt 3. Verwenden Sie den vertrauenswürdigen Schlüssel, um den Datenschlüssel zu umschließen

Codebeispiele finden Sie unter den folgenden Links, um den in Schritt 1 referenzierten Datenschlüssel mit dem vertrauenswürdigen Schlüssel zu verbinden, den Sie in Schritt 2 festgelegt haben. In jedem Beispiel wird gezeigt, wie Schlüssel umgebrochen werden.

- [AWS CloudHSM PKCS #11 -Beispiele](#)
- [AWS CloudHSM JCE-Beispiele](#)

Wie entpacke ich einen Datenschlüssel mit einem vertrauenswürdigen Schlüssel für AWS CloudHSM

Um einen Datenschlüssel zu entpacken AWS CloudHSM, benötigen Sie einen vertrauenswürdigen Schlüssel, der auf `true` `CKA_UNWRAP` gesetzt ist. Um ein solcher Schlüssel zu sein, muss er auch folgende Kriterien erfüllen:

- Das `CKA_TRUSTED`-Attribut des Schlüssels muss auf „true“ gesetzt werden.
- Der Schlüssel muss anhand von `CKA_UNWRAP_TEMPLATE` und verwandter Attribute angeben, welche Aktionen Datenschlüssel ausführen können, wenn sie entpackt wurden. Wenn Sie beispielsweise möchten, dass ein entpackter Schlüssel nicht exportierbar ist, legen Sie `CKA_EXPORTABLE = FALSE` als Teil von `CKA_UNWRAP_TEMPLATE` fest.

Note

`CKA_UNWRAP_TEMPLATE` ist nur mit PKCS #11 verfügbar.

Wenn eine Anwendung einen Schlüssel zum Entpacken einreicht, kann die Anwendung auch ihre eigene Unwrap-Vorlage bereitstellen. Wenn Sie eine Unwrap-Vorlage angeben und die Anwendung eine eigene Unwrap-Vorlage bereitstellt, verwendet das HSM beide Vorlagen, um Attributnamen und -werte auf den Schlüssel anzuwenden. Wenn jedoch während der Entpackungsanforderung ein Wert in `CKA_UNWRAP_TEMPLATE` des vertrauenswürdigen Schlüssels mit einem von der Anwendung bereitgestellten Attribut in Konflikt gerät, schlägt die Entpackungsanforderung fehl.

Ein Beispiel für das Entpacken eines Datenschlüssels mit einem vertrauenswürdigen Schlüssel finden Sie in [diesem PKCS-#11-Beispiel](#).

Schlüsselverwaltung mit CloudHSM CLI

Wenn Sie die [neueste SDK-Versionsserie](#) verwenden, verwenden Sie die [CloudHSM-CLI](#), um die Schlüssel in Ihrem AWS CloudHSM Cluster zu verwalten. Weitere Informationen finden Sie in den folgenden Themen.

- [Unter Verwendung vertrauenswürdiger Schlüssel](#) wird beschrieben, wie Sie mithilfe der CloudHSM-CLI vertrauenswürdige Schlüssel zur Sicherung von Daten erstellen.
- Das [Generieren von Schlüsseln](#) umfasst Anweisungen zum Erstellen von Schlüsseln, einschließlich symmetrischer Schlüssel, RSA-Schlüssel und EC-Schlüssel.
- Unter [Schlüssel löschen](#) wird beschrieben, wie Schlüsselbesitzer Schlüssel löschen.
- Unter [Teilen und Aufheben der Freigabe von Schlüsseln](#) wird beschrieben, wie Schlüsselinhaber Schlüssel teilen und deren Weitergabe rückgängig machen.
- Das [Filtern von Schlüsseln](#) enthält Richtlinien zur Verwendung von Filtern zum Auffinden von Schlüsseln.
- [Manage Key Quorum Authentication \(M of N\)](#) bietet Richtlinien zur Einrichtung und Verwendung der Quorumauthentifizierung mit Schlüsseln.

Generieren Sie Schlüssel mit CloudHSM CLI

Bevor Sie einen Schlüssel generieren können, müssen Sie [CloudHSM-CLI](#) starten und sich als Crypto-Benutzer (CU) anmelden. Zum Generieren der Schlüssel auf dem HSM verwenden Sie den entsprechenden Befehl für die Art von Schlüssel, die Sie erstellen möchten.

Themen

- [Generieren Sie symmetrische Schlüssel mit CloudHSM CLI](#)
- [Generieren Sie asymmetrische Schlüssel mit CloudHSM CLI](#)
- [AWS CloudHSM wichtige verwandte Themen](#)

Generieren Sie symmetrische Schlüssel mit CloudHSM CLI

Verwenden Sie die unter aufgeführten Befehle, um symmetrische [Die Kategorie „Symmetrisch generieren“ in CloudHSM CLI](#) Schlüssel für zu generieren. AWS CloudHSM Verwenden Sie den Befehl `help key generate-symmetric`, um alle verfügbaren Optionen anzuzeigen.

Generieren eines AES-Schlüssels

Verwenden Sie den `key generate-symmetric aes`-Befehl, um AES-Schlüssel zu generieren. Verwenden Sie den Befehl `help key generate-symmetric aes`, um alle verfügbaren Optionen anzuzeigen.

Example

Das folgende Beispiel generiert einen 32-Byte-AES-Schlüssel.

```
aws-cloudhsm > key generate-symmetric aes \  
  --label aes-example \  
  --key-length-bytes 32
```

Argumente

<LABEL>

Gibt eine benutzerdefinierte Bezeichnung für den AES-Schlüssel an.

Erforderlich: Ja

<KEY-LENGTH-BYTES>

Gibt die Schlüssellänge in Byte an.

Zulässige Werte:

- 16, 24 und 32

Erforderlich: Ja

<KEY_ATTRIBUTES>

Gibt eine durch Leerzeichen getrennte Liste von Schlüsselattributen an, die für den generierten AES-Schlüssel festgelegt werden sollen, in der Form von `KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE` (z. B. `sign=true`)

Eine Liste der unterstützten AWS CloudHSM Schlüsselattribute finden Sie unter [Schlüsselattribute für CloudHSM-CLI](#).

Erforderlich: Nein

<SESSION>

Erstellt einen Schlüssel, der nur in der aktuellen Sitzung existiert. Der Schlüssel kann nach Ende der Sitzung nicht wiederhergestellt werden. Verwenden Sie diesen Parameter, wenn Sie einen Schlüssel zum Packen nur für kurze Zeit benötigen, z. B. einen Schlüssel, der einen anderen Schlüssel verschlüsselt und dann schnell entschlüsselt. Verwenden Sie keinen Sitzungsschlüssel, um Daten zu verschlüsseln, die Sie nach dem Ende der Sitzung möglicherweise entschlüsseln müssen.

Um einen Sitzungsschlüssel in einen persistenten (Token-)Schlüssel zu ändern, verwenden Sie [key set-attribute](#).

Standardmäßig handelt es sich bei der Generierung von Schlüsseln um persistent/token Schlüssel. Die Verwendung <SESSION>ändert dies, indem sichergestellt wird, dass es sich bei einem mit diesem Argument generierten Schlüssel um eine Sitzung/einen flüchtigen Schlüssel handelt

Erforderlich: Nein

Generieren Sie einen generischen geheimen Schlüssel

Verwenden Sie den `key generate-symmetric generic-secret`-Befehl, um generische geheime Schlüssel zu generieren. Verwenden Sie den Befehl `help key generate-symmetric generic-secret`, um alle verfügbaren Optionen anzuzeigen.

Example

Das folgende Beispiel generiert einen generischen 32-Byte-Schlüssel.

```
aws-cloudhsm > key generate-symmetric generic-secret \  
  --label generic-secret-example \  
  --key-length-bytes 32
```

Argumente**<LABEL>**

Gibt eine benutzerdefinierte Bezeichnung für den generischen geheimen Schlüssel an.

Erforderlich: Ja

<KEY-LENGTH-BYTES>

Gibt die Schlüssellänge in Byte an.

Zulässige Werte:

- 1 bis 800

Erforderlich: Ja

<KEY_ATTRIBUTES>

Gibt eine durch Leerzeichen getrennte Liste von Schlüsselattributen an, die für den generierten generischen geheimen Schlüssel festgelegt werden sollen, in der Form von KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE (z. B. sign=true)

Eine Liste der unterstützten AWS CloudHSM Schlüsselattribute finden Sie unter.

[Schlüsselattribute für CloudHSM-CLI](#)

Erforderlich: Nein

<SESSION>

Erstellt einen Schlüssel, der nur in der aktuellen Sitzung existiert. Der Schlüssel kann nach Ende der Sitzung nicht wiederhergestellt werden. Verwenden Sie diesen Parameter, wenn Sie einen Schlüssel zum Packen nur für kurze Zeit benötigen, z. B. einen Schlüssel, der einen anderen Schlüssel verschlüsselt und dann schnell entschlüsselt. Verwenden Sie keinen Sitzungsschlüssel, um Daten zu verschlüsseln, die Sie nach dem Ende der Sitzung möglicherweise entschlüsseln müssen.

Um einen Sitzungsschlüssel in einen persistenten (Token-)Schlüssel zu ändern, verwenden Sie [key set-attribute](#).

Standardmäßig handelt es sich bei der Generierung von Schlüsseln um persistent/token Schlüssel. Die Verwendung <SESSION>ändert dies, indem sichergestellt wird, dass es sich bei einem mit diesem Argument generierten Schlüssel um eine Sitzung/einen flüchtigen Schlüssel handelt

Erforderlich: Nein

Generieren Sie asymmetrische Schlüssel mit CloudHSM CLI

Verwenden Sie die unter aufgeführten Befehle [Die generate-asymmetric-pair Kategorie in CloudHSM CLI](#), um asymmetrische Schlüsselpaare für Cluster zu generieren. AWS CloudHSM

Generieren Sie einen RSA-Schlüssel

Verwenden Sie den `key generate-asymmetric-pair rsa`-Befehl, um ein RSA-Schlüsselpaar zu erzeugen. Verwenden Sie den Befehl `help key generate-asymmetric-pair rsa`, um alle verfügbaren Optionen anzuzeigen.

Example

Das folgende Beispiel generiert ein 2048-Bit-RSA-Schlüsselpaar.

```
aws-cloudhsm > key generate-asymmetric-pair rsa \  
  --public-exponent 65537 \  
  --modulus-size-bits 2048 \  
  --public-label rsa-public-example \  
  --private-label rsa-private-example
```

Argumente

<PUBLIC_LABEL>

Gibt eine benutzerdefinierte Bezeichnung für den öffentlichen Schlüssel an.

Erforderlich: Ja

<PRIVATE_LABEL>

Gibt eine benutzerdefinierte Bezeichnung für den private-key an.

Erforderlich: Ja

<MODULUS_SIZE_BITS>

Gibt die Länge des Moduls in Bits an. Der minimale Wert beträgt 2048.

Erforderlich: Ja

<PUBLIC_EXPONENT>

Gibt den öffentlichen Exponenten an. Bei diesem Wert muss es sich eine ungerade Zahl gleich oder größer als 65537 handeln.

Erforderlich: Ja

<PUBLIC_KEY_ATTRIBUTES>

Gibt eine durch Leerzeichen getrennte Liste von Schlüsselattributen an, die für den generierten öffentlichen RSA-Schlüssel festgelegt werden sollen, in der Form von KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE (z. B. sign=true).

Eine Liste der unterstützten AWS CloudHSM Schlüsselattribute finden Sie unter [Schlüsselattribute für CloudHSM-CLI](#).

Erforderlich: Nein

<SESSION>

Erstellt einen Schlüssel, der nur in der aktuellen Sitzung existiert. Der Schlüssel kann nach Ende der Sitzung nicht wiederhergestellt werden. Verwenden Sie diesen Parameter, wenn Sie einen Schlüssel zum Packen nur für kurze Zeit benötigen, z. B. einen Schlüssel, der einen anderen Schlüssel verschlüsselt und dann schnell entschlüsselt. Verwenden Sie keinen Sitzungsschlüssel, um Daten zu verschlüsseln, die Sie nach dem Ende der Sitzung möglicherweise entschlüsseln müssen.

Um einen Sitzungsschlüssel in einen persistenten (Token-)Schlüssel zu ändern, verwenden Sie [key set-attribute](#).

Standardmäßig handelt es sich bei der Generierung von Schlüsseln um persistent/token Schlüssel. Die Verwendung <SESSION>ändert dies, indem sichergestellt wird, dass es sich bei einem mit diesem Argument generierten Schlüssel um eine Sitzung/einen flüchtigen Schlüssel handelt

Erforderlich: Nein

Generieren von ECC- (Elliptic Curve Cryptography-)Schlüsselpaaren

Verwenden Sie den key generate-asymmetric-pair ec-Befehl, um ein EC-Schlüsselpaar zu generieren. Wenn Sie alle verfügbaren Optionen, einschließlich einer Liste der unterstützten Ellipsenkurven anzeigen möchten, verwenden Sie den Befehl help key generate-asymmetric-pair ec.

Example

Das folgende Beispiel erzeugt ein EC-Schlüsselpaar unter Verwendung der elliptischen Kurve Secp384r1.

```
aws-cloudhsm > key generate-asymmetric-pair ec \  
  --curve secp384r1 \  
  --public-label ec-public-example \  
  --private-label ec-private-example
```

Argumente

<PUBLIC_LABEL>

Gibt eine benutzerdefinierte Bezeichnung für den öffentlichen Schlüssel an. Die maximal zulässige Größe für Client SDK 5.11 und `label` höher beträgt 127 Zeichen. Das Client-SDK 5.10 und früher ist auf 126 Zeichen begrenzt.

Erforderlich: Ja

<PRIVATE_LABEL>

Gibt eine benutzerdefinierte Bezeichnung für den private-key an. Die maximal zulässige Größe für Client-SDK 5.11 und höher `label` beträgt 127 Zeichen. Das Client-SDK 5.10 und früher ist auf 126 Zeichen begrenzt.

Erforderlich: Ja

<CURVE>

Gibt die ID für die elliptische Kurve an.

Zulässige Werte:

- prime256v1
- secp256r1
- secp224r1
- secp384r1
- secp256k1
- secp521r1
- ed25519 (wird nur auf `hsm2m.medium`-Instances im Nicht-FIPS-Modus unterstützt)

Erforderlich: Ja

<PUBLIC_KEY_ATTRIBUTES>

Gibt eine durch Leerzeichen getrennte Liste von Schlüsselattributen an, die für den generierten öffentlichen EC-Schlüssel festgelegt werden sollen, in der Form von KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE (z. B. verify=true).

Eine Liste der unterstützten Schlüsselattribute finden Sie unter [AWS CloudHSM Schlüsselattribute für CloudHSM-CLI](#)

Erforderlich: Nein

<PRIVATE_KEY_ATTRIBUTES>

Gibt eine durch Leerzeichen getrennte Liste von Schlüsselattributen an, die für den generierten privaten EC-Schlüssel festgelegt werden sollen, in der Form von KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE (z. B. sign=true).

Eine Liste der unterstützten AWS CloudHSM Schlüsselattribute finden Sie unter [Schlüsselattribute für CloudHSM-CLI](#).

Erforderlich: Nein

<SESSION>

Erstellt einen Schlüssel, der nur in der aktuellen Sitzung existiert. Der Schlüssel kann nach Ende der Sitzung nicht wiederhergestellt werden. Verwenden Sie diesen Parameter, wenn Sie einen Schlüssel zum Packen nur für kurze Zeit benötigen, z. B. einen Schlüssel, der einen anderen Schlüssel verschlüsselt und dann schnell entschlüsselt. Verwenden Sie keinen Sitzungsschlüssel, um Daten zu verschlüsseln, die Sie nach dem Ende der Sitzung möglicherweise entschlüsseln müssen.

Um einen Sitzungsschlüssel in einen persistenten (Token-)Schlüssel zu ändern, verwenden Sie [key set-attribute](#).

Standardmäßig handelt es sich bei den generierten Schlüsseln um persistente (Token-)Schlüssel. Das Übergeben von <SESSION> ändert dies und stellt sicher, dass es sich bei einem mit diesem Argument generierten Schlüssel um einen (kurzlebigen) Sitzungsschlüssel handelt.

Erforderlich: Nein

AWS CloudHSM wichtige verwandte Themen

In den folgenden Abschnitten finden Sie weitere Informationen zu Tasten in AWS CloudHSM.

- [Schlüsselattribute für CloudHSM-CLI](#)
- [Die generate-asymmetric-pair Kategorie in CloudHSM CLI](#)
- [Die Kategorie „Symmetrisch generieren“ in CloudHSM CLI](#)

Schlüssel mit CloudHSM CLI löschen

Verwenden Sie das Beispiel in diesem Thema, um einen Schlüssel mit der [CloudHSM-CLI](#) zu löschen. Nur Schlüsselinhaber können Schlüssel löschen.

Themen

- [Beispiel: Löschen Sie einen Schlüssel](#)
- [Verwandte Themen](#)

Beispiel: Löschen Sie einen Schlüssel

1. Führen Sie den key list-Befehl aus, um den Schlüssel zu identifizieren, den Sie löschen möchten

```
aws-cloudhsm > key list --filter attr.label="my_key_to_delete" --verbose
{
  "error_code": 0,
  "data": {
    "matched_keys": [
      {
        "key-reference": "0x00000000000540011",
        "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. Nachdem Sie den Schlüssel identifiziert haben, führen Sie `key delete` mit dem eindeutigen `label`-Schlüsselattribut aus, um den Schlüssel zu löschen:

```

aws-cloudhsm > key delete --filter attr.label="my_key_to_delete"
{
  "error_code": 0,
  "data": {
    "message": "Key deleted successfully"
  }
}

```



```
}  
}
```

3. Führen Sie den `key list`-Befehl mit dem eindeutigen `label`-Schlüsselattribut aus und vergewissern Sie sich, dass der Schlüssel gelöscht wurde. Wie im folgenden Beispiel gezeigt, befindet sich kein Schlüssel mit dem Label `my_key_to_delete` im HSM-Cluster:

```
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  
  }  
}
```

Verwandte Themen

- [Schlüsselattribute für CloudHSM-CLI](#)
- [Löschen Sie einen Schlüssel mit CloudHSM CLI](#)

Schlüssel mit CloudHSM CLI teilen und deren Freigabe rückgängig machen

Verwenden Sie die Befehle in diesem Thema, um Schlüssel in der [CloudHSM-CLI](#) zu teilen und deren Freigabe aufzuheben. In gehört AWS CloudHSM der Schlüssel dem Krypto-Benutzer (CU), der den Schlüssel erstellt. Der Besitzer kann die `key unshare` Befehle `key share` und verwenden, um den Schlüssel mit anderen CUs zu teilen oder die gemeinsame Nutzung aufzuheben. Benutzer, mit denen der Schlüssel gemeinsam genutzt wird, können den Schlüssel in kryptographischen Operationen verwenden. Sie können jedoch den Schlüssel nicht exportieren, löschen oder mit anderen Benutzern teilen.

Bevor Sie einen Schlüssel freigeben können, müssen Sie sich beim HSM als der Crypto-Benutzer (CU) anmelden, dem der Schlüssel gehört.

Themen

- [Beispiel: Freigabe und Aufhebung der Freigabe eines Schlüssels](#)
- [Verwandte Themen](#)

Beispiel: Freigabe und Aufhebung der Freigabe eines Schlüssels

Example

Das folgende Beispiel zeigt, wie Sie einen Schlüssel für den Crypto-Benutzer (CU) `alice` freigeben und die Freigabe aufheben. Neben den Befehlen `key share` und `key unshare` erfordern die Befehle zum Teilen und Aufheben der Freigabe auch einen bestimmten Schlüssel mithilfe von [CloudHSM-CLI-Schlüsselfiltern](#) und den spezifischen Benutzernamen des Benutzers, mit dem der Schlüssel geteilt oder nicht geteilt werden soll.

1. Führen Sie zunächst den `key list`-Befehl mit einem Filter aus, um einen bestimmten Schlüssel zurückzugeben und zu sehen, mit wem der Schlüssel bereits geteilt wurde.

```
aws-cloudhsm > key list --filter attr.label="rsa_key_to_share" --verbose
{
  "error_code": 0,
  "data": {
    "matched_keys": [
      {
        "key-reference": "0x000000000001c0686",
        "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. Sehen Sie sich die `shared-users`-Ausgabe an, um festzustellen, mit wem der Schlüssel derzeit geteilt wird.
3. Geben Sie den folgenden Befehl ein, um diesen Schlüssel mit dem Crypto-Benutzer (CU) `alice` zu teilen:

```

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"
  }
}

```

Beachten Sie, dass dieser Befehl zusammen mit dem `key share`-Befehl die eindeutige Bezeichnung des Schlüssels und den Namen des Benutzers verwendet, mit dem der Schlüssel geteilt wird.

4. Führen Sie den `key list`-Befehl aus, um zu bestätigen, dass der Schlüssel mit `alice` geteilt wurde:

```

aws-cloudhsm > key list --filter attr.label="rsa_key_to_share" --verbose
{
  "error_code": 0,
  "data": {
    "matched_keys": [
      {
        "key-reference": "0x000000000001c0686",
        "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. Führen Sie den folgenden unshare-Befehl aus, um die gemeinsame Nutzung desselben Schlüssels mit alice rückgängig zu machen:

```

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"
  }
}

```

Beachten Sie, dass dieser Befehl zusammen mit dem `key unshare`-Befehl die eindeutige Bezeichnung des Schlüssels und den Namen des Benutzers verwendet, mit dem der Schlüssel geteilt wird.

6. Führen Sie den `key list`-Befehl erneut aus und vergewissern Sie sich, dass der Schlüssel nicht an den Crypto-Benutzer `alice` weitergegeben wurde:

```
aws-cloudhsm > key list --filter attr.label="rsa_key_to_share" --verbose
{
  "error_code": 0,
  "data": {
    "matched_keys": [
      {
        "key-reference": "0x000000000001c0686",
        "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

```



```
}  
}
```

Verwandte Themen

- [Schlüsselattribute für CloudHSM-CLI](#)
- [Einen Schlüssel mit CloudHSM CLI teilen](#)
- [Freigabe eines Schlüssels mithilfe der CloudHSM-CLI rückgängig machen](#)
- [Schlüssel mithilfe der CloudHSM-CLI filtern](#)

Schlüssel mithilfe der CloudHSM-CLI filtern

Verwenden Sie die folgenden Tastenbefehle, um die standardisierten Schlüsselfiltermechanismen für [CloudHSM-CLI](#) zu nutzen.

- key list
- key delete
- key share
- key unshare
- key set-attribute

Um and/or Filterschlüssel mit der CloudHSM-CLI auszuwählen, verwenden Tastenbefehle einen standardisierten Filtermechanismus, der auf [Schlüsselattribute für CloudHSM-CLI](#) Ein Schlüssel oder eine Gruppe von Schlüsseln kann in Tastenbefehlen angegeben werden, indem ein oder mehrere AWS CloudHSM Attribute verwendet werden, die einen einzelnen Schlüssel oder mehrere Schlüssel identifizieren können. Der Schlüsselfiltermechanismus funktioniert nur für Schlüssel, die der aktuell angemeldete Benutzer besitzt und gemeinsam nutzt, sowie für alle öffentlichen Schlüssel im AWS CloudHSM Cluster.

Themen

- [Voraussetzungen](#)
- [Filtern, um einen einzelnen Schlüssel zu finden](#)
- [Fehler bei der Filterung](#)
- [Verwandte Themen](#)

Voraussetzungen

Um Schlüssel zu filtern, müssen Sie als Krypto-Benutzer angemeldet sein (CUs).

Filtern, um einen einzelnen Schlüssel zu finden

Bitte beachten Sie, dass in den folgenden Beispielen jedes Attribut, das als Filter verwendet wird, in der Form von `attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE` geschrieben werden muss. Wenn Sie beispielsweise nach dem Label-Attribut filtern möchten, schreiben Sie `attr.label=my_label`.

Example Verwenden Sie ein einzelnes Attribut, um einen einzelnen Schlüssel zu finden

Dieses Beispiel zeigt, wie mit nur einem einzigen identifizierenden Attribut nach einem einzigen eindeutigen Schlüssel gefiltert wird.

```
aws-cloudhsm > key list --filter attr.label="my_unique_key_label" --verbose
{
  "error_code": 0,
  "data": {
    "matched_keys": [
      {
        "key-reference": "0x000000000001c0686",
        "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 Verwenden Sie mehrere Attribute, um einen einzelnen Schlüssel zu finden

Das folgende Beispiel zeigt, wie Sie einen einzelnen Schlüssel mithilfe mehrerer Schlüsselattribute finden.

```
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": "0x00000000000540011",
        "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 Filtern, um einen Satz von Schlüsseln zu finden

Das folgende Beispiel zeigt, wie Sie nach einer Reihe von privaten RSA-Schlüsseln filtern.

```

aws-cloudhsm > key list --filter attr.key-type=rsa attr.class=private-key --verbose
{
  "error_code": 0,
  "data": {
    "matched_keys": [
      {
        "key-reference": "0x000000000001c0686",
        "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": "0x00000000000540011",
  "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
  }
}
```

Fehler bei der Filterung

Bestimmte Schlüsseloperationen können jeweils nur mit einem einzigen Schlüssel ausgeführt werden. Bei diesen Vorgängen gibt die CloudHSM-CLI einen Fehler aus, wenn die Filterkriterien nicht ausreichend verfeinert sind und mehrere Schlüssel den Kriterien entsprechen. Ein solches Beispiel ist unten mit dem Schlüssel delete dargestellt.

Example Filtrationsfehler beim Abgleichen von zu vielen Schlüsseln

```
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."
}
```

Verwandte Themen

- [Schlüsselattribute für CloudHSM-CLI](#)

Markieren Sie einen Schlüssel mithilfe der CloudHSM-CLI als vertrauenswürdig

Der Inhalt dieses Abschnitts enthält Anweisungen zur Verwendung der CloudHSM CLI, um einen Schlüssel als vertrauenswürdig zu kennzeichnen.

1. Melden Sie sich mit dem [CloudHSM-CLI-login-Befehl](#) als Crypto-Benutzer (CU) an.
2. Verwenden Sie den key list-Befehl, um die Schlüsselreferenz des Schlüssels zu identifizieren, den Sie als vertrauenswürdig markieren möchten. Das folgende Beispiel listet den Schlüssel mit dem Label test_aes_trusted auf.


```
aws-cloudhsm > key list --filter attr.label=test_aes_trusted
{
  "error_code": 0,
  "data": {
    "matched_keys": [
      {
        "key-reference": "0x00000000000200333",
        "attributes": {
          "label": "test_aes_trusted"
        }
      }
    ],
    "total_key_count": 1,
    "returned_key_count": 1
  }
}
```

3. Melden Sie sich mit dem [Melden Sie sich mit der CloudHSM-CLI von einem HSM ab](#)-Befehl als Crypto-Benutzer (CU) ab.
4. Melden Sie sich mit dem Befehl als [Melden Sie sich mit der CloudHSM-CLI bei einem HSM an](#)-Administrator an.
5. Verwenden Sie den Befehl [key set-attribute](#) mit der Schlüsselreferenz, die Sie in Schritt 2 identifiziert haben, und setzen Sie den vertrauenswürdigen Wert des Schlüssels auf true:

```
aws-cloudhsm > key set-attribute --filter key-reference=<Key Reference> --name
trusted --value true
{
  "error_code": 0,
  "data": {
    "message": "Attribute set successfully"
  }
}
```

Verwaltung der Quorum-Authentifizierung (M-of-N-Zugriffskontrolle) mithilfe der CloudHSM-CLI

Die Hardware-Sicherheitsmodule (HSMs) in Ihrem AWS CloudHSM Cluster unterstützen die Quorumauthentifizierung, auch bekannt als M-of-N-Zugriffskontrolle. Bei der Quorumauthentifizierung

kann kein einzelner Benutzer auf dem HSM quorumgesteuerte Operationen ausführen. Stattdessen muss eine Mindestanzahl von HSM-Benutzern (mindestens 2) zusammenarbeiten, um diese Operationen durchzuführen. Die Quorum-Authentifizierung bietet eine zusätzliche Schutzebene, da Genehmigungen von mehreren HSM-Benutzern erforderlich sind.

Die Quorum-Authentifizierung kann die folgenden Vorgänge steuern:

- Verwendung und Verwaltung von HSM-Schlüsseln durch einen [Krypto-Benutzer](#) — Erstellen von Signaturen mit einem Schlüssel oder Verpacken, Entpacken, Teilen, Aufheben der gemeinsamen Nutzung und Einstellung eines Schlüsselattributs.

Wichtige Überlegungen

- Ein HSM-Benutzer kann sein eigenes Quorum-Token signieren, d. h. der anfordernde Benutzer kann eine der erforderlichen Genehmigungen für die Quorum-Authentifizierung bereitstellen.
- Sie wählen die Mindestzahl der Quorum-Genehmiger für die Quorum-kontrollierten Operationen aus. Die kleinste Zahl, die Sie wählen können, ist zwei (2), und die größte Zahl, die Sie wählen können, ist acht (8).
- Das HSM kann bis zu 1.024 Quorum-Token speichern. Wenn das HSM bereits über 1.024 Token verfügt, wenn Sie versuchen, ein neues zu erstellen, löscht das HSM eines der abgelaufenen Token. Standardmäßig verfallen Token zehn Minuten nach ihrer Erstellung.
- Wenn die Multi-Faktor-Authentifizierung (MFA) aktiviert ist, verwendet der Cluster denselben Schlüssel für die Quorumauthentifizierung und für MFA. Weitere Informationen zur Verwendung von Quorum-Authentifizierung und MFA finden Sie unter [CloudHSM CLI zur Verwaltung von MFA verwenden](#).
- Jedes HSM kann jeweils nur ein Token pro Admin-Service, aber mehrere Token pro Crypto User-Dienst enthalten.

Die folgenden Themen stellen weitere Informationen zur Quorum-Authentifizierung in AWS CloudHSM zur Verfügung.

Themen

- [Quorum-Authentifizierungsprozess für CloudHSM CLI](#)
- [Unterstützte AWS CloudHSM Dienstnamen und Typen für die Quorum-Authentifizierung mit CloudHSM CLI](#)

- [Richten Sie die Quorum-Authentifizierung für AWS CloudHSM Krypto-Benutzer mithilfe der CloudHSM-CLI ein](#)
- [Schlüsselverwaltung und -nutzung mit aktivierter Quorum-Authentifizierung für die AWS CloudHSM Verwendung von CloudHSM CLI](#)

Quorum-Authentifizierungsprozess für CloudHSM CLI

In den folgenden Schritten werden die Quorum-Authentifizierungsprozesse für CloudHSM CLI zusammengefasst. Die exakten Schritte und Tools finden Sie unter [Schlüsselverwaltung und -nutzung mit aktivierter Quorum-Authentifizierung für die AWS CloudHSM Verwendung von CloudHSM CLI](#).

1. Jeder Benutzer des Hardware-Sicherheitsmoduls (HSM) erstellt einen asymmetrischen Schlüssel zum Signieren. Benutzer tun dies außerhalb des HSMs und achten darauf, den Schlüssel angemessen zu schützen.
2. Jeder HSM-Benutzer meldet sich beim HSM an und registriert den öffentlichen Teil seines Signierungsschlüssels (den öffentlichen Schlüssel) beim HSM.
3. Wenn ein HSM-Benutzer eine Quorum-kontrollierte Operation durchführen möchte, meldet sich derselbe Benutzer beim HSM an und erhält ein Quorum-Token.
4. Der HSM-Benutzer gibt das Quorum-Token an einen oder mehrere andere HSM-Benutzer weiter und bittet um deren Zustimmung.
5. Die anderen HSM-Benutzer genehmigen, indem sie mit ihren Schlüsseln das Quorum-Token kryptographisch signieren. Dies geschieht außerhalb des HSMs.
6. Wenn der HSM-Benutzer über die erforderliche Anzahl von Genehmigungen verfügt, meldet sich derselbe Benutzer beim HSM an und führt den quorumgesteuerten Vorgang mit dem --approval-Argument aus. Dabei wird die signierte Quorum-Tokendatei bereitgestellt, die alle erforderlichen Genehmigungen (Signaturen) enthält.
7. Das HSM verwendet die registrierten, öffentlichen Schlüssel jedes Unterzeichners, um die Signaturen zu verifizieren. Wenn die Signaturen gültig sind, genehmigt das HSM das Token und der quorumgesteuerte Vorgang wird ausgeführt.

Unterstützte AWS CloudHSM Dienstnamen und Typen für die Quorum-Authentifizierung mit CloudHSM CLI

Admin-Dienste: Die Quorum-Authentifizierung wird für Dienste mit Administratorrechten verwendet, z. B. für das Erstellen von Benutzern, das Löschen von Benutzern, das Ändern von Benutzerkennwörtern, das Festlegen von Quorumwerten und das Deaktivieren von Quorum- und MFA-Funktionen.

Crypto User Services: Die Quorum-Authentifizierung wird für privilegierte Dienste verwendet, die Krypto-Benutzern vorbehalten sind und mit einem bestimmten Schlüssel verknüpft sind, z. B. das Signieren mit einem Schlüssel, sharing/unsharing einem Schlüssel, wrapping/unwrapping einem Schlüssel und das Festlegen eines Schlüsselattributs. Der Quorumwert eines zugehörigen Schlüssels wird konfiguriert, wenn der Schlüssel generiert, importiert oder entpackt wird. Der Quorumwert muss gleich oder kleiner als die Anzahl der Benutzer sein, denen der Schlüssel zugeordnet ist. Dazu gehören Benutzer, mit denen der Schlüssel geteilt wird, und der Schlüsselbesitzer.

Jeder Dienstyp wird weiter in einen qualifizierenden Dienstnamen unterteilt, der eine bestimmte Gruppe von Quorum-unterstützten Dienstvorgängen enthält, die ausgeführt werden können.

Service-Name	Servicetyp	Serviceoperationen
user	Admin.	• user create • user delete • user change-password • user change-mfa
quorum	Admin.	• Quorum-Tokenzeichen set-quorum-value
Cluster 1	Admin.	• Cluster MTLS register-trust-anchor • Cluster-MTLS deregister-trust-anchor • Durchsetzung von Cluster-M TLS-Sätzen
Schlüsselverwaltung	Crypto-Benutzer	• Schlüsselhülle

Service-Name	Servicetyp	Serviceoperationen
		• Schlüssel auspacken • key share • key unshare • key set-attribute
Verwendung des Schlüssels	Crypto-Benutzer	• Schlüsselzeichen

[1] Der Cluster-Service ist ausschließlich auf hsm2m.medium verfügbar

Richten Sie die Quorum-Authentifizierung für AWS CloudHSM Krypto-Benutzer mithilfe der CloudHSM-CLI ein

[In diesen Themen wird beschrieben, wie Sie Ihr CloudHSM für die Quorumauthentifizierung durch Krypto-Benutzer konfigurieren.](#) Führen Sie diese Schritte einmal bei der Ersteinrichtung durch. Weitere Informationen zur Schlüsselverwaltung und -nutzung finden Sie unter [Schlüsselverwaltung und -nutzung mit aktivierter Quorum-Authentifizierung für die AWS CloudHSM Verwendung von CloudHSM CLI](#).

Themen

- [Voraussetzungen](#)
- [Schritt 1. Erstellen und Registrieren eines Schlüssels für das Signieren](#)
- [Schritt 2. Legen Sie die Schlüsselquorumwerte bei der Schlüsselgenerierung fest](#)

Voraussetzungen

- Vertrautheit mit der [CloudHSM-CLI](#)

Schritt 1. Erstellen und Registrieren eines Schlüssels für das Signieren

Um die Quorum-Authentifizierung zu verwenden, muss jeder Krypto-Benutzer alle der folgenden Schritte ausführen:

Themen

- [Erstellen eines RSA-Schlüsselpaares](#)

- [Erstellen Sie ein Registrierungstoken](#)
- [Signieren Sie das unsignierte Registrierungstoken](#)
- [Registrieren Sie den öffentlichen Schlüssel beim HSM](#)

Erstellen eines RSA-Schlüsselpaares

Es gibt viele verschiedene Möglichkeiten, ein Schlüsselpaar zu erstellen und zu schützen. In den folgenden Beispielen wird gezeigt, wie dies mit [OpenSSL](#) durchgeführt wird.

Example– Erstellen eines privaten Schlüssels mit OpenSSL

Das folgende Beispiel zeigt, wie OpenSSL verwendet wird, um einen 2048-Bit-RSA-Schlüssel zu erstellen. Um dieses Beispiel zu verwenden, *<crypto_user1.key>* ersetzen Sie es durch den Namen der Datei, in der Sie den Schlüssel speichern möchten.

```
$ openssl genrsa -out <crypto_user1.key>
Generating RSA private key, 2048 bit long modulus
.....+++
.+++
e is 65537 (0x10001)
```

Generieren eines öffentlichen Schlüssels mit dem privaten Schlüssel, den Sie gerade erstellt haben.

Example– Erstellen Sie einen öffentlichen Schlüssel mit OpenSSL

Das folgende Beispiel zeigt, wie Sie OpenSSL verwenden, um einen öffentlichen Schlüssel aus dem privaten Schlüssel zu erstellen, den Sie gerade erstellt haben.

```
$ openssl rsa -in crypto_user1.key -outform PEM -pubout -out crypto_user1.pub
writing RSA key
```

Erstellen Sie ein Registrierungstoken

Sie erstellen ein Token und signieren es mit dem privaten Schlüssel, den Sie gerade im vorherigen Schritt generiert haben.

Erstellen Sie ein Registrierungstoken

1. Verwenden Sie den folgenden Befehl, um die CloudHSM-CLI zu starten:

Linux

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

Windows

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

2. Erstellen Sie ein Registrierungstoken, indem Sie den Befehl [quorum token-sign generate](#) ausführen:

```
aws-cloudhsm > quorum token-sign generate --service registration --token /path/
tokenfile
{
  "error_code": 0,
  "data": {
    "path": "/path/tokenfile"
  }
}
```

3. Der Befehl [quorum token-sign generate](#) generiert ein Registrierungstoken im angegebenen Dateipfad. Untersuchen Sie die Tokendatei:

```
$ cat /path/tokenfile
{
  "version": "2.0",
  "tokens": [
    {
      "approval_data": <approval data in base64 encoding>,
      "unsigned": <unsigned token in base64 encoding>,
      "signed": ""
    }
  ]
}
```

Die Tokendatei besteht aus Folgendem:

- `approval_data`: Ein Base64-kodiertes zufälliges Datentoken, dessen Rohdaten das Maximum von 245 Byte nicht überschreiten.
- `unsigned`: Ein Base64-codiertes und SHA256 gehashtes Token der `approval_data`.

- signiert: Ein base64-kodiertes signiertes Token (Signatur) des unsignierten Tokens, unter Verwendung des zuvor mit OpenSSL generierten privaten RSA-Schlüssels mit 2048 Bit.

Sie signieren das unsignierte Token mit dem privaten Schlüssel, um nachzuweisen, dass Sie Zugriff auf den privaten Schlüssel haben. Sie benötigen die vollständig mit einer Signatur und dem öffentlichen Schlüssel aufgefüllte Registrierungstokendatei, um den Krypto-Benutzer als Quorum-Benutzer im Cluster zu registrieren. AWS CloudHSM

Signieren Sie das unsignierte Registrierungstoken

1. Dekodieren Sie das Base64-kodierte unsignierte Token und platzieren Sie es in einer Binärdatei:

```
$ echo -n '6BMUj6mUjjko6ZLCEdzG1WpR5sILhFJfqhW1ej30q1g=' | base64 -d >
crypto_user.bin
```

2. Verwenden Sie OpenSSL und den privaten Schlüssel, um das jetzt binäre unsignierte Registrierungstoken zu signieren und eine binäre Signaturdatei zu erstellen:

```
$ openssl pkeyutl -sign \
-inkey crypto_user1.key \
-pkeyopt digest:sha256 \
-keyform PEM \
-in crypto_user.bin \
-out crypto_user.sig.bin
```

3. Kodieren Sie die binäre Signatur in Base64:

```
$ base64 -w0 crypto_user.sig.bin > crypto_user.sig.b64
```

4. Kopieren Sie die Base64-kodierte Signatur und fügen Sie sie in die Token-Datei ein:

```
{
  "version": "2.0",
  "tokens": [
    {
      "approval_data": <approval data in base64 encoding>,
      "unsigned": <unsigned token in base64 encoding>,
      "signed": <signed token in base64 encoding>
    }
  ]
}
```



```
}
```

Registrieren Sie den öffentlichen Schlüssel beim HSM

Nach der Erstellung eines Schlüssels muss der Krypto-Benutzer den öffentlichen Schlüssel im Cluster registrieren. AWS CloudHSM

1. Starten Sie CloudHSM CLI:

Linux

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

Windows

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

2. Melden Sie sich als der Krypto-Benutzer an, dessen öffentlichen Schlüssel Sie registrieren möchten.

```
aws-cloudhsm > login --username crypto_user1 --role crypto-user
Enter password:
{
  "error_code": 0,
  "data": {
    "username": "crypto_user1",
    "role": "crypto-user"
  }
}
```

3. Registrieren Sie den öffentlichen Schlüssel mit dem. [Registrieren Sie die Token-Sign-Quorum-Strategie eines Benutzers mithilfe der CloudHSM-CLI](#) Weitere Informationen finden Sie im folgenden Beispiel. Alternativ können Sie auch den Befehl `help user change-quorum token-sign register` ausführen.

Example— Registriert einen öffentlichen Schlüssel beim AWS CloudHSM Cluster

Das folgende Beispiel zeigt, wie Sie den `user change-quorum token-sign register` Befehl in der CloudHSM-CLI verwenden, um einen öffentlichen Schlüssel für Krypto-Benutzer beim HSM zu

registrieren. Um diesen Befehl verwenden zu können, muss der Crypto-Benutzer beim HSM angemeldet sein. Ersetzen Sie diese Werte durch Ihre eigenen Werte:

```
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: Der Dateipfad zur PEM-Datei mit öffentlichem Schlüssel

Erforderlich: Ja

/path/token_file: Der Dateipfad mit dem Token, das mit dem privaten Benutzerschlüssel signiert wurde

Erforderlich: Ja

4. Nachdem alle Krypto-Benutzer ihre öffentlichen Schlüssel registriert haben, zeigt die Ausgabe des `user list` Befehls dies im Quorumfeld an und gibt an, welche aktivierte Quorumstrategie verwendet wird.

In diesem Beispiel hat der AWS CloudHSM Cluster zwei mit jeweils denselben Krypto-Benutzern HSMs, wie in der folgenden Ausgabe des Befehls dargestellt. `user list` Weitere Informationen zum Erstellen von Benutzern finden Sie unter [Benutzerverwaltung mit CloudHSM CLI](#).

```
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"
    }
]
}
```

Schritt 2. Legen Sie die Schlüsselquorumwerte bei der Schlüsselgenerierung fest

Um die Quorumauthentifizierung zu verwenden, muss sich ein Krypto-Benutzer beim HSM anmelden und dann die zugehörigen Schlüsselquorumwerte festlegen. Dies ist die Mindestanzahl an Genehmigungen durch Krypto-Benutzer, die für die Ausführung von HSM-Schlüsseloperationen erforderlich sind. [management/usage](#) Weitere Informationen zu den zugehörigen Tastenbefehlen im Zusammenhang mit der Schlüsselverwaltung oder der Schlüsselverwendung finden Sie unter.

[Unterstützte Dienste und Typen](#)

Generieren Sie ein key pair mit festgelegten Schlüsselquorumwerten

1. Verwenden Sie den folgenden Befehl, um die CloudHSM-CLI zu starten:

Linux

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

Windows

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

2. Melden Sie sich mit der CloudHSM-CLI als Crypto-Benutzer an.

```
aws-cloudhsm > login --username crypto_user1 --role crypto-user
Enter password:
{
  "error_code": 0,
  "data": {
    "username": "crypto_user1",
    "role": "crypto-user"
  }
}
```

}

In diesem Beispiel wird ein RSA-Schlüsselpaar generiert, bei dem die Schlüsselquorumwerte zwei (2) sowohl für Schlüsselverwaltungs- als auch für Schlüsselverwendungsvorgänge festgelegt sind. Sie können einen beliebigen Wert zwischen Null (0) und acht (8) wählen, bis hin zur Gesamtzahl der Krypto-Benutzer auf dem HSM. In diesem Beispiel hat das HSM drei (3) Krypto-Benutzer, sodass der maximal mögliche Wert drei (3) ist. Beachten Sie, dass wir in diesem Beispiel den Schlüssel `<crypto_user2>` während der Schlüsselgenerierung mit teilen. Beachten Sie auch, dass öffentliche Schlüssel keine Quorumwerte haben.

```
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": "0x00000000000640006",
      "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":
"0xbdf471a3d2a869492f51c767bec8780730ae6479a9a75efffe7cea3594fb28ca518630e7b1d988b45d2fedc830
    "modulus-size-bits": 2048
  },
  "private_key": {
    "key-reference": "0x00000000000640007",
    "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":
      "0xbdf471a3d2a869492f51c767bec8780730ae6479a9a75efffe7cea3594fb28ca518630e7b1d988b45d2fedc830",
    "modulus-size-bits": 2048
  }
}
}
}
}

```

Bei der Generierung eines Schlüssels mit Quorumsteuerelementen muss der Schlüssel einer Mindestanzahl von Benutzern zugeordnet werden, die dem größten Schlüsselquorumwert entspricht. Zu den assoziierten Benutzern gehören der Schlüsselinhaber und Crypto-Benutzer, mit denen der

Schlüssel geteilt wird. Um die Mindestanzahl der Benutzer zu ermitteln, mit denen der Schlüssel geteilt werden soll, ermitteln Sie den größten Quorumwert zwischen dem Quorumwert für die Schlüsselnutzung und dem Quorumwert für die Schlüsselverwaltung und subtrahieren Sie 1, um den Schlüsselbesitzer zu berücksichtigen, der standardmäßig mit dem Schlüssel verknüpft ist. Verwenden Sie den Befehl, um den Schlüssel mit mehreren Benutzern zu teilen. [Einen Schlüssel mit CloudHSM CLI teilen](#)

Wenn der Schlüssel bei der Schlüsselgenerierung nicht mit genügend Benutzern geteilt wird, schlägt dies fehl, wie unten dargestellt.

```
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."
}
```

Schlüsselverwaltung und -nutzung mit aktivierter Quorum-Authentifizierung für die AWS CloudHSM Verwendung von CloudHSM CLI

Nachdem Sie die Quorumauthentifizierung für Ihren AWS CloudHSM Cluster konfiguriert haben, können Krypto-Benutzer keine HSM-Schlüsselverwaltungs- oder Verwendungsvorgänge selbst durchführen, wenn ihrem Schlüssel Quorumwerte zugeordnet sind. In diesem Thema wird erklärt, wie ein Krypto-Benutzer ein temporäres Token erhalten kann, um eine HSM-Schlüsselverwaltung oder Schlüsselverwendung durchzuführen.

Note

Jedes Quorum-Token ist für einen Vorgang gültig. Wenn dieser Vorgang erfolgreich ist, ist das Token nicht mehr gültig und der Krypto-Benutzer muss ein neues Token erwerben. Ein Quorum-Token ist nur während Ihrer aktuellen Anmeldesitzung gültig. Wenn Sie sich von der CloudHSM-CLI abmelden oder die Netzwerkverbindung unterbrochen wird, ist das Token

nicht mehr gültig und Sie benötigen ein neues Token. Sie können ein CloudHSM-Token nur innerhalb der CloudHSM-CLI CLI. Sie können es nicht zur Authentifizierung in einer anderen Anwendung verwenden.

Das folgende Beispiel zeigt die Ausgabe, wenn ein Krypto-Benutzer versucht, eine Signatur mit einem Quorum-Schlüssel auf dem HSM zu erstellen, nachdem die Quorum-Authentifizierung konfiguriert wurde. Der Befehl schlägt mit einem Quorum Failed Fehler fehl, was bedeutet, dass die Quorumauthentifizierung fehlgeschlagen ist:

```
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"  
}
```

Ein Krypto-Benutzer muss die folgenden Aufgaben ausführen, um ein temporäres Token für die Durchführung eines Schlüsselverwaltungs- oder Schlüsselverwendungsvorgangs auf dem HSM zu erhalten:

Schritte

- [Schritt 1. Abrufen eines Quorum-Tokens](#)
- [Schritt 2. Holen Sie sich Signaturen von genehmigenden Krypto-Benutzern](#)
- [Schritt 3. Genehmigen Sie das Token auf dem CloudHSM; clustern Sie und führen Sie einen Vorgang aus](#)

Schritt 1. Abrufen eines Quorum-Tokens

1. Starten Sie CloudHSM CLI.

Linux

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

Windows

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

2. Melden Sie sich als Krypto-Benutzer beim Cluster an.

```
aws-cloudhsm > login --username <crypto_user1> --role crypto-user --  
password password123
```

Dieses Beispiel meldet `crypto_user1` sich mit der Rolle bei der CloudHSM-CLI an `crypto-user`. Ersetzen Sie diese Werte durch Ihre eigenen.

```
{  
  "error_code": 0,  
  "data": {  
    "username": "crypto_user1",  
    "role": "crypto-user"  
  }  
}
```

3. Generieren Sie mit dem `quorum token-sign generate` Befehl ein Quorum-Token.

`key-usageld` identifiziert im folgenden Befehl den Dienstnamen, für den Sie das von Ihnen generierte Token verwenden werden. In diesem Fall ist das Token für Schlüsselverwendungsoperationen (`key-usageService`) vorgesehen. In diesem Beispiel wird das `--filter` Flag verwendet, um das Token einem bestimmten Schlüssel zuzuordnen.

```
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"  
  }  
}
```

In diesem Beispiel wird ein Quorum-Token für den Krypto-Benutzer mit dem Benutzernamen `crypto_user1` abgerufen und das Token in einer Datei mit dem Namen `crypto_user1.token` gespeichert. Zum Verwenden des Beispiels ersetzen Sie diese Werte durch Ihre eigenen:

Der `quorum token-sign generate` Befehl generiert ein Quorum-Token für den Schlüsselverwendungsdienst im angegebenen Dateipfad. Sie können die Tokendatei überprüfen:

```
$ cat </path/crypto_user1.token>
{
  "version": "2.0",
  "service": "key-usage",
  "key_reference": "0x00000000000680006",
  "approval_data":
  "AAIABQAAABkAAAAAGgABi5CDa9x9VyyRIaFbkSrHgJjcnlwdG9fdXNlcgAAAAAAAAAAAAAAAAAAAAAAAAAAAA
+GJj8gXo9lKuANGNyeXB0b191c2VyAAAAAAAAAAAAAAAAAAAAAAAAAAAAAGc9AEsAAAAAAAAAAAAA==",
  "token": "5G1goW0lQU4fw4QI1bxkPGZV0VoDugFGuSKE/k67ncM=",
  "signatures": []
}
```

Die Tokendatei besteht aus Folgendem:

- service: Eine Kennung für den Quorumdienst, dem das Token zugeordnet ist.
- key_reference: Ein Bezeichner für den Schlüssel, dem dieses Quorum-Token zugeordnet ist.
- approval_data: Ein Base64-codiertes Rohdaten-Token, das vom HSM generiert wurde.
- token: Ein Base64-kodiertes und SHA-256-Hash-Token der approval_data
- Signaturen: Ein Array von Base64-codierten signierten Token (Signaturen) des Tokens ohne Vorzeichen. Jede Genehmigersignatur hat die Form eines JSON-Objektliterals:

```
{
  "username": "<APPROVER_USERNAME>",
  "role": "<APPROVER_ROLE>",
  "signature": "<APPROVER_RSA2048_BIT_SIGNATURE>"
}
```

Jede Signatur wird anhand des Ergebnisses erstellt, das ein Genehmiger unter Verwendung seines entsprechenden privaten RSA-2048-Bit-Schlüssels erstellt hat, dessen öffentlicher Schlüssel im HSM registriert wurde.

4. Überprüfen Sie das neue Quorum-Token für den Benutzerdienst. Der quorum token-sign list Befehl bestätigt, dass das Token auf CloudHSM vorhanden ist.

```
aws-cloudhsm > quorum token-sign list
{
  "error_code": 0,
  "data": {
    "tokens": [
```

```
{
  "username": "crypto_user",
  "service": "key-usage",
  "key-reference": "0x0000000000680006",
  "minimum-token-count": 2
}
```

Das `minimum-token-count` stellt eine aggregierte Clusteransicht der minimal verwendbaren Anzahl von Schlüssel-Tokens dar, die dem Benutzernamen, dem Service und der Schlüsselreferenz entsprechen und von einem einzelnen HSM im Cluster abgerufen werden.

Nehmen wir zum Beispiel einen 2-HSM-Cluster an und erhalten zwei (2) Schlüsselverwendungstoken, die von Benutzer `crypto_user1` für Schlüssel mit Referenz `0x0000000000680006` vom ersten HSM im Cluster generiert wurden, und wir erhalten ein (1) Schlüsselverwendungstoken, das von Benutzer `crypto_user1` für Schlüssel mit Referenz `0x0000000000680006` von dem anderen HSM im Cluster generiert wurde, wird angezeigt `"minimum-token-count": 1`.

Schritt 2. Holen Sie sich Signaturen von genehmigenden Krypto-Benutzern

Ein Krypto-Benutzer, der über ein Quorum-Token verfügt, muss das Token von anderen Krypto-Benutzern genehmigen lassen. Um ihre Zustimmung zu erteilen, verwenden die anderen Crypto-Benutzer ihren Signaturschlüssel, um das Token außerhalb des HSM kryptografisch zu signieren.

Es gibt viele verschiedene Möglichkeiten, ein Token zu signieren. Das folgende Beispiel zeigt, wie das Token mit [OpenSSL](#) signiert wird. Um ein anderes Signaturtool zu verwenden, stellen Sie sicher, dass das Tool den privaten Schlüssel (Signaturschlüssel) des Krypto-Benutzers verwendet, um einen SHA-256-Digest des Tokens zu signieren.

In diesem Beispiel benötigt der Krypto-Benutzer, der das Token (`crypto-user`) besitzt, mindestens zwei (2) Genehmigungen. Die folgenden Beispielbefehle zeigen, wie zwei (2) Krypto-Benutzer OpenSSL verwenden können, um das Token kryptografisch zu signieren.

1. Dekodieren Sie das Base64-kodierte unsignierte Token und platzieren Sie es in einer Binärdatei:

```
$echo -n '5Glgow0lQU4fw4QIlbXkPGZV0VoDugFGuSKE/k67ncM=' | base64 -d >
crypto_user1.bin
```

2. Verwenden Sie OpenSSL und den privaten Schlüssel des Genehmigers, um das unsignierte binäre Quorum-Token für den Benutzerdienst zu signieren und eine binäre Signaturdatei zu erstellen:

```
$openssl pkeyutl -sign \
-inkey crypto_user1.key \
-pkeyopt digest:sha256 \
-keyform PEM \
-in crypto_user1.bin \
-out crypto_user1.sig.bin
```

3. Kodieren Sie die binäre Signatur in Base64:

```
$ base64 -w0 crypto_user1.sig.bin > crypto_user1.sig.b64
```

4. Kopieren Sie die Base64-kodierte Signatur und fügen Sie sie in die Tokendatei ein. Verwenden Sie dabei das zuvor für die Signatur des Genehmigers angegebene JSON-Objektliteralformat:

```
{
  "version": "2.0",
  "service": "key-usage",
  "key_reference": "0x00000000000680006",
  "approval_data":
  "AAIABQAAABkAAAAAGgABi5CDa9x9VyyRIaFbkSrHgJjcnlwdG9fdXNlcmgAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
+GJj8gXo9lKuANGNyeXB0b191c2VyAAAAAAAAAAAAAAAAAAAAAAAAAAAAAGc9AEsAAAAAAAAAAAAA==",
  "token": "5Glgow0lQU4fw4QIlbXkPGZV0VoDugFGuSKE/k67ncM=",
  "signatures": [
    {
      "username": "crypto_user1",
      "role": "crypto-user",
      "signature": "wa7aPzmGwBjcEoZ6jAzYASp841Afg0vcI27Y/
tGlCj1E9DawnFw5Uf0IJT2Ca7T5XD2ThVkuI0B+dhAomdqYNl6aUUFrJyH9GBJ
+E0PmA5jNVm25tzeRWBjzneTg4/
zTeE2reNqrHFHicWnttQL9jS09J1znuDGWDe0HaBKWUaz2gUInJRqmeXDsZYdSvZksrqUH5dci/
RsaDE2+tGiS9g0RcIkFbsPW4HpGe2e5HVzGsqrV803PK1YQv6+fymfcNTTuoxKcHAK0jp143QSuSIu2gVq7KI8mSmmW
+oiukaNfLJr+MoDKzAvCGDg4cDArg=="
    },
    {
```

```

    "username": "crypto_user2",
    "role": "crypto-user",
    "signature": "wa7aPzmGwBjcEoZ6jAzYASp841Afg0vcI27Y/
tGlCj1E9DawnFw5Uf0IJT2Ca7T5XD2ThVkUi0B+dhAomdqYNl6aUUFrJyH9GBJ
+E0PmA5jNVm25tzeRWBjzneTg4/
zTeE2reNqrHFHicWnttQL9jS09J1znuDGWDe0HaBKWUaz2gUInJRqmeXDsZYdSvZksrqUH5dci/
RsaDE2+tGiS9g0RcIkFbsPW4HpGe2e5HVzGsqrV803PK1YQv6+fymfcNTTuoxKcHAK0jpl43QSuSIu2gVq7KI8mSmmW
+oiukaNfLJr+MoDKzAvCGDg4cDArg=="
  }
]
}

```

Schritt 3. Genehmigen Sie das Token auf dem CloudHSM; clustern Sie und führen Sie einen Vorgang aus

Sobald ein Krypto-Benutzer über die erforderlichen Genehmigungen und Signaturen verfügt hat, kann er dieses Token zusammen mit einem Schlüsselverwaltungs- oder Schlüsselverwendungsvorgang an den CloudHSM-Cluster weitergeben.

Stellen Sie sicher, dass die Schlüsseloperation dem entsprechenden Quorum-Dienst entspricht, der dem Quorum-Token zugeordnet ist. Weitere Informationen finden Sie [Unterstützte Dienste und Typen](#) unter.

Während der Transaktion wird das Token innerhalb des AWS CloudHSM Clusters genehmigt und der angeforderte Schlüsselvorgang ausgeführt. Der Erfolg der Schlüsseloperation hängt sowohl von einem gültigen genehmigten Quorum-Token als auch von einer gültigen Schlüsseloperation ab.

Example Generieren Sie eine Signatur mit dem RSA-PKCS-Mechanismus

Im folgenden Beispiel erstellt ein angemeldeter Krypto-Benutzer eine Signatur mit einem Schlüssel auf dem HSM:

```

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": "0x00000000000640007",
    "signature":
    "h6hMqXacBrT3x3MXV13RXHdQno0+IQ6iy0kVrGzo23+eoWT0ZZgrSpBCu5KcuP6IYYHw9goQ5CfPf4jI1n05m/

```

```
IUJtF1A1lmcz0HjEy1CJ7ICXNReDRye0U8m43dkJzt0UdkbtkDJGAcxkbKHLZ02uWsGXaQ8b0KhoGwsRAHHF6nldTXquIC
+pZmUS38ythybney94Wj6fzY0ER8v7VIY5ijQGa3LfxrjSG4aw6QijEEbno5LSf18ahEaVKmVEnDBL54tylCJBGvGsYSY9H
TDd2wfvP4PaxbFRyyHaw=="
}
```

Wenn der Krypto-Benutzer versucht, einen weiteren Vorgang zur Verwendung des HSM-Schlüssels mit demselben Token durchzuführen, schlägt er fehl:

```
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"
}
```

Um eine weitere HSM-Schlüsseloperation durchzuführen, muss der Krypto-Benutzer ein neues Quorum-Token generieren, neue Signaturen von Genehmigern einholen und die gewünschte Schlüsseloperation mit dem Argument `--approval` ausführen, um das Quorum-Token bereitzustellen.

Verwenden Sie den, `quorum token-sign list` um nach verfügbaren Token zu suchen. Dieses Beispiel zeigt, dass der Krypto-Benutzer keine genehmigten Token hat.

```
aws-cloudhsm > quorum token-sign list
{
  "error_code": 0,
  "data": {
    "tokens": []
  }
}
```

Schlüsselmanagement mit der AWS CloudHSM KMU

Wenn Sie die [neueste SDK-Versionsserie](#) verwenden, verwenden Sie die [CloudHSM-CLI](#), um die Schlüssel in Ihrem AWS CloudHSM Cluster zu verwalten.

Wenn Sie die [vorherige SDK-Versionsserie](#) verwenden, können Sie Schlüssel auf den Hardware-Sicherheitsmodulen (HSM) in Ihrem AWS CloudHSM Cluster mit dem Befehlszeilentool `key_mgmt_util` (KMU) verwalten. Bevor Sie Schlüssel verwalten können, müssen Sie den AWS

CloudHSM Client starten, key_mgmt_util starten und sich bei dem anmelden. HSMs Weitere Informationen finden Sie unter [Erste Schritte mit key_mgmt_util](#).

- Unter [Verwendung vertrauenswürdiger Schlüssel](#) wird beschrieben, wie Sie mithilfe von PKCS #11-Bibliotheksattributen und der CMU vertrauenswürdige Schlüssel zur Sicherung von Daten erstellen.
- Das [Generieren von Schlüsseln](#) umfasst Anweisungen zum Generieren von Schlüsseln, einschließlich symmetrischer Schlüssel, RSA-Schlüssel und EC-Schlüssel.
- Beim [Importieren von Schlüsseln](#) wird detailliert beschrieben, wie Schlüsselbesitzer Schlüssel importieren.
- [Exportieren von Schlüsseln](#) enthält Einzelheiten darüber, wie Schlüsselbesitzer Schlüssel exportieren.
- [Löschen von Schlüsseln](#) enthält Einzelheiten darüber, wie Schlüsselbesitzer Schlüssel löschen.
- Unter [Teilen und Aufheben der Freigabe von Schlüsseln](#) wird beschrieben, wie Schlüsselinhaber Schlüssel teilen und deren Weitergabe rückgängig machen.

AWS CloudHSM Generieren Sie Schlüssel mit der KMU

Um Schlüssel auf dem Hardware-Sicherheitsmodul (HSM) zu generieren, verwenden Sie den Befehl in AWS CloudHSM key_mgmt_util (KMU), der dem Schlüsseltyp entspricht, den Sie generieren möchten.

Themen

- [AWS CloudHSM Generieren Sie symmetrische Schlüssel mit der KMU](#)
- [Generieren Sie RSA-Schlüsselpaare mit der AWS CloudHSM KMU](#)
- [Generieren Sie ECC-Schlüsselpaare \(Elliptic Curve Cryptography\) mithilfe der KMU AWS CloudHSM](#)

AWS CloudHSM Generieren Sie symmetrische Schlüssel mit der KMU

Verwenden Sie den [genSymKey](#) Befehl in AWS CloudHSM key_mgmt_util (KMU), um AES und andere Arten von symmetrischen Schlüsseln für zu generieren. AWS CloudHSM Verwenden Sie den Befehl genSymKey -h, um alle verfügbaren Optionen anzuzeigen.

In dem folgenden Beispiel wird ein 256-Bit-AES-Schlüssel erstellt.

```
Command: genSymKey -t 31 -s 32 -1 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
```

Generieren Sie RSA-Schlüsselpaare mit der AWS CloudHSM KMU

Verwenden Sie den Befehl [gen key pair in AWS CloudHSM key_mgmt_util](#), um ein [RSAKey AWS CloudHSM RSA-Schlüsselpaar](#) für zu generieren. Verwenden Sie den Befehl `genRSAKeyPair -h`, um alle verfügbaren Optionen anzuzeigen.

Das folgende Beispiel generiert ein 2048-Bit-RSA-Schlüsselpaar.

```
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
```

Generieren Sie ECC-Schlüsselpaare (Elliptic Curve Cryptography) mithilfe der KMU AWS CloudHSM

Verwenden Sie den Befehl [gen key pair in AWS CloudHSM key_mgmt_util](#), um ein [ECCKey AWS CloudHSM ECC-Schlüsselpaar](#) für zu generieren. Wenn Sie alle verfügbaren Optionen, einschließlich einer Liste der unterstützten Ellipsenkurven anzeigen möchten, verwenden Sie den Befehl `genECCKeyPair -h`.

Das folgende Beispiel erzeugt ein ECC-Schlüsselpaar unter Verwendung der elliptischen Kurve P-384, die in der [NIST FIPS Publikation 186-4](#) definiert ist.

```
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
```

Importieren Sie AWS CloudHSM Schlüssel mit der KMU

Um geheime Schlüssel — also symmetrische Schlüssel und asymmetrische private Schlüssel — mithilfe von AWS CloudHSM `key_mgmt_util` in das Hardware Security Module (HSM) zu importieren, müssen Sie zunächst einen Wrapping-Schlüssel auf dem HSM erstellen. Öffentliche Schlüssel können direkt ohne Umhüllungsschlüssel importiert werden.

Themen

- [Importieren Sie geheime Schlüssel mit der KMU AWS CloudHSM](#)
- [Importieren Sie öffentliche Schlüssel mit der AWS CloudHSM KMU](#)

Importieren Sie geheime Schlüssel mit der KMU AWS CloudHSM

Gehen Sie wie folgt vor, um einen geheimen Schlüssel AWS CloudHSM mithilfe von `key_mgmt_util` (KMU) zu importieren. Wenn Sie einen geheimen Schlüssel importieren möchten, speichern Sie ihn zunächst in einer Datei. Speichern Sie symmetrische Schlüssel als Rohbytes und asymmetrische private Schlüssel im PEM-Format.

Dieses Beispiel zeigt, wie man einen geheimen Klartext-Schlüssel aus einer Datei in das HSM importiert. Verwenden Sie den Befehl, um einen verschlüsselten Schlüssel aus einer Datei in das HSM zu importieren. [unWrapKey](#)

So importieren Sie einen geheimen Schlüssel

1. Verwenden Sie den [genSymKey](#) Befehl, um einen Wrapping-Schlüssel zu erstellen. Mit dem folgenden Befehl wird ein 128-Bit-AES-Schlüssel erstellt, der nur für die aktuelle Sitzung gilt. Sie können einen Sitzungsschlüssel oder einen persistenten Schlüssel als Wrapping-Schlüssel verwenden.

```
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. Verwenden Sie, abhängig vom Typ des geheimen Schlüssels, den Sie importieren, einen der folgenden Befehle.
 - Um einem symmetrischen Schlüssel zu importieren, verwenden Sie den Befehl [imSymKey](#). Mit dem folgenden Befehl wird ein AES-Schlüssel aus einer Datei mit dem Namen `aes256.key` importiert, wobei der im vorangehenden Schritt erstellte Umhüllungsschlüssel benötigt wird. Verwenden Sie den Befehl `imSymKey -h`, um alle verfügbaren Optionen anzuzeigen.

```
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
```

- Um einen asymmetrischen privaten Schlüssel zu importieren, verwenden Sie den Befehl [importPrivateKey](#). Mit dem folgenden Befehl wird ein privater Schlüssel aus einer Datei mit dem Namen `rsa2048.key` importiert, wobei der im vorangehenden Schritt erstellte Umhüllungsschlüssel benötigt wird. Verwenden Sie den Befehl `importPrivateKey -h`, um alle verfügbaren Optionen anzuzeigen.

```
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
```

Importieren Sie öffentliche Schlüssel mit der AWS CloudHSM KMU

Verwenden Sie den [importPubKey](#) Befehl in AWS CloudHSM key_mgmt_util (KMU), um einen öffentlichen Schlüssel zu importieren. Verwenden Sie den Befehl `importPubKey -h`, um alle verfügbaren Optionen anzuzeigen.

Im folgenden Beispiel wird öffentlicher RSA-Schlüssel aus einer Datei mit dem Namen `rsa2048.pub` importiert.

```
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
```

Exportieren Sie Schlüssel mit der KMU AWS CloudHSM

Um AWS CloudHSM geheime Schlüssel — also symmetrische Schlüssel und asymmetrische private Schlüssel — mithilfe von AWS CloudHSM key_mgmt_util (KMU) aus dem Hardware Security Module (HSM) zu exportieren, müssen Sie zuerst einen Wrapping-Schlüssel erstellen. Öffentliche Schlüssel können direkt ohne Umhüllungsschlüssel exportiert werden.

Nur der Schlüsselbesitzer kann einen Schlüssel exportieren. Benutzer, mit denen der Schlüssel gemeinsam genutzt wird, können den Schlüssel in kryptographischen Operationen verwenden, aber ihn nicht exportieren. Wenn Sie dieses Beispiel ausführen, stellen Sie sicher, dass Sie einen von Ihnen erstellten Schlüssel exportieren.

⚠ Important

Der [exSymKey](#) Befehl schreibt eine (unverschlüsselte) Klartextkopie des geheimen Schlüssels in eine Datei. Der Exportvorgang erfordert einen Wrapping-Schlüssel. Der Schlüssel in der Datei jedoch ist kein Wrapping-Schlüssel. Um eine verpackte (verschlüsselte) Kopie eines Schlüssels zu exportieren, verwenden Sie den Befehl [wrapKey](#).

Themen

- [Exportieren Sie geheime Schlüssel mit der KMU AWS CloudHSM](#)
- [Exportieren Sie öffentliche Schlüssel mit der KMU AWS CloudHSM](#)

Exportieren Sie geheime Schlüssel mit der KMU AWS CloudHSM

Gehen Sie wie folgt vor, um AWS CloudHSM mithilfe von key_mgmt_util (KMU) einen geheimen Schlüssel zu exportieren.

So exportieren Sie einen geheimen Schlüssel

1. Verwenden Sie den Befehl, um einen Wrapping-Schlüssel [genSymKey](#) zu erstellen. Mit dem folgenden Befehl wird ein 128-Bit-AES-Schlüssel erstellt, der nur für die aktuelle Sitzung gilt.

```
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. Verwenden Sie, abhängig vom Typ des geheimen Schlüssels, den Sie exportieren, einen der folgenden Befehle.
 - Verwenden Sie den [exSymKey](#) Befehl, um einen symmetrischen Schlüssel zu exportieren. Der folgende Befehl exportiert einen AES-Schlüssel in eine Datei mit dem Namen `aes256.key.exp`. Verwenden Sie den Befehl `exSymKey -h`, um alle verfügbaren Optionen anzuzeigen.

```
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

Die Ausgabe des Befehls besagt, dass ein „Wrapped Symmetric Key“ in die Ausgabedatei geschrieben wurde. Die Ausgabedatei enthält jedoch einen Klartext-Schlüssel (nicht verpackt). Um einen verpackten (verschlüsselten) Schlüssel in eine Datei zu exportieren, verwenden Sie den Befehl [wrapKey](#).

- Um einem privaten Schlüssel zu exportieren, verwenden Sie den Befehl `exportPrivateKey`. Der folgende Befehl exportiert einen privaten Schlüssel in eine Datei mit dem Namen `rsa2048.key.exp`. Verwenden Sie den Befehl `exportPrivateKey -h`, um alle verfügbaren Optionen anzuzeigen.

```
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
```

Exportieren Sie öffentliche Schlüssel mit der KMU AWS CloudHSM

Verwenden Sie den `exportPubKey` Befehl in AWS CloudHSM `key_mgmt_util` (KMU), um einen öffentlichen Schlüssel zu exportieren. Verwenden Sie den Befehl `exportPubKey -h`, um alle verfügbaren Optionen anzuzeigen.

Im folgenden Beispiel wird öffentlicher RSA-Schlüssel in eine Datei mit dem Namen `rsa2048.pub.exp` exportiert.

```
Command: exportPubKey -k 524294 -out rsa2048.pub.exp
```

```
PEM formatted public key is written to rsa2048.pub.key
```

```
Cfm3ExportPubKey returned: 0x00 : HSM Return: SUCCESS
```

Löschen Sie Schlüssel mit KMU und CMU

Verwenden Sie zum Löschen eines Schlüssels den Befehl [deleteKey](#) wie in dem folgenden Beispiel. Nur der Schlüsselbesitzer kann einen Schlüssel löschen.

```
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
```

Schlüssel mit KMU und CMU teilen und deren Freigabe rückgängig machen

In AWS CloudHSM, der CU, die den Schlüssel erstellt, gehört er. Der Besitzer verwaltet den Schlüssel, kann ihn exportieren und löschen und kann ihn in kryptographischen Operationen verwenden. Der Besitzer kann den Schlüssel außerdem mit anderen CU-Benutzern teilen. Benutzer, mit denen der Schlüssel gemeinsam genutzt wird, können den Schlüssel in kryptographischen Operationen verwenden. Sie können jedoch den Schlüssel nicht exportieren, löschen oder mit anderen Benutzern teilen.

Sie können Schlüssel gemeinsam mit anderen CU-Benutzern verwenden, wenn Sie den Schlüssel erstellen, z. B. indem Sie die `-u` Parameter der Befehle [genSymKey](#) oder [gen RSAKey Pair](#) verwenden. Um vorhandene Schlüssel für einen anderen HSM-Benutzer freizugeben, verwenden Sie das Befehlszeilentool [cloudhsm_mgmt_util](#). Dies unterscheidet sich von den meisten der in diesem Abschnitt dokumentierten Aufgaben, die das Befehlszeilentool [key_mgmt_util](#) verwenden.

Bevor Sie einen Schlüssel gemeinsam nutzen können, müssen Sie `cloudhsm_mgmt_util` starten, die Verschlüsselung aktivieren end-to-end und sich bei dem anmelden. HSMs Um einen Schlüssel freizugeben, melden Sie sich am HSM als der Crypto-Benutzer (CU, Crypto User) an, dem der Schlüssel gehört. Nur der Eigentümer eines Schlüssels ist berechtigt, einen Schlüssel freizugeben.

Verwenden Sie den `shareKey` Befehl, um einen Schlüssel gemeinsam zu nutzen oder die gemeinsame Nutzung aufzuheben. Geben Sie dabei das Handle des Schlüssels und das des oder der Benutzer an. IDs Geben Sie eine kommasetrennte Benutzerliste an, um die gemeinsame Nutzung für mehrere Benutzer aufzuheben oder die gemeinsame Nutzung aufzuheben. IDs Um einen Schlüssel freizugeben, verwenden Sie den Befehl `1` als letzten Parameter, wie im folgenden Beispiel gezeigt. Um die Freigabe aufzuheben, verwenden Sie `0`.

```
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)
```

Nachfolgend finden Sie die allgemeine Syntax für den Befehl shareKey.

```
aws-cloudhsm > shareKey <key handle> <user ID> <Boolean: 1 for share, 0 for unshare>
```

So markieren Sie einen Schlüssel mit dem AWS CloudHSM Management Utility als vertrauenswürdig

Der Inhalt dieses Abschnitts enthält Anweisungen zur Verwendung des AWS CloudHSM Management Utility (CMU), um einen Schlüssel als vertrauenswürdig zu kennzeichnen.

1. Melden Sie sich mit dem Befehl [loginHSM](#) als Crypto Officer (CO) an.
2. Verwenden Sie den [Legen Sie die Attribute von AWS CloudHSM Schlüsseln mit CMU fest-](#) Befehl, bei dem OBJ_ATTR_TRUSTED (Wert 134) auf „true“ (1) gesetzt ist.

```
aws-cloudhsm > setAttribute <Key Handle> 134 1
```


Cluster-Backups in AWS CloudHSM

AWS CloudHSM erstellt regelmäßig mindestens alle 24 Stunden Backups Ihres Clusters. Jede Sicherung enthält verschlüsselte Kopien der folgenden Daten:

- Benutzer (COs CUs, und AUs)
- Schlüsselmaterial und Zertifikate
- Hardware Security Module (HSM)-Konfiguration und -Richtlinien

Sie können den Dienst nicht anweisen, Sicherungen zu erstellen, aber Sie können bestimmte Aktionen ergreifen, die den Dienst dazu zwingen, eine Sicherung zu erstellen. Der Dienst erstellt eine Sicherung, wenn Sie eine der folgenden Aktionen ausführen:

- Aktivieren eines Clusters
- Fügen Sie einem aktiven Cluster ein HSM hinzu
- Entfernen Sie ein HSM aus einem aktiven Cluster

AWS CloudHSM löscht Backups auf der Grundlage der Backup-Aufbewahrungsrichtlinie, die Sie beim Erstellen von Clustern festgelegt haben. Informationen zur Verwaltung der Aufbewahrungsrichtlinie für Sicherungen finden Sie unter [Konfigurieren Sie die Aufbewahrung von Backups](#).

Themen

- [Arbeiten mit AWS CloudHSM Cluster-Backups](#)
- [Löschen Sie AWS CloudHSM Cluster-Backups](#)
- [AWS CloudHSM Backups wiederherstellen](#)
- [Konfigurieren Sie die Aufbewahrungsrichtlinie für AWS CloudHSM Backups](#)
- [AWS CloudHSM Cluster-Backups zwischen AWS Regionen kopieren](#)
- [Arbeiten mit gemeinsam genutzten Backups in AWS CloudHSM](#)

Arbeiten mit AWS CloudHSM Cluster-Backups

Wenn Sie ein Hardware-Sicherheitsmodul (HSM) zu einem Cluster hinzufügen AWS CloudHSM , in dem zuvor ein oder mehrere aktive Module enthalten waren HSMs, stellt der Dienst das neueste

Backup auf dem neuen HSM wieder her. Verwenden Sie Backups, um Ihre selten genutzte Nutzung HSMs zu verwalten. Wenn Sie das HSM nicht mehr benötigen, löschen Sie es, um eine Sicherung zu erstellen. Wenn Sie das HSM später benötigen, erstellen Sie ein neues im selben Cluster. Mit dieser Aktion wird das Backup wiederhergestellt, das Sie zuvor mit dem Vorgang „HSM löschen“ erstellt haben.

Abgelaufene Schlüssel oder inaktive Benutzer werden entfernt

Möglicherweise möchten Sie unerwünschtes kryptografisches Material wie abgelaufene Schlüssel oder inaktive Benutzer aus Ihrer Umgebung entfernen. Dies ist ein zweistufiger Prozess. Löschen Sie zunächst diese Materialien aus Ihrem HSM. Löschen Sie anschließend alle vorhandenen Sicherungen. Wenn Sie diesen Vorgang befolgen, stellen Sie sicher, dass Sie gelöschte Informationen nicht wiederherstellen, wenn Sie einen neuen Cluster aus einem Backup initialisieren. Weitere Informationen finden Sie unter [the section called “Sicherungen löschen”](#).

Notfallwiederherstellung erwägen

Sie können einen Cluster aus einer Sicherung erstellen. Möglicherweise möchten Sie dies tun, um einen Erholungspunkt für Ihren Cluster festzulegen. Nominieren Sie eine Sicherung, die alle Benutzer, das Schlüsselmaterial und die Zertifikate enthält, die Sie für Ihren Recovery Point benötigen, und verwenden Sie dann diese Sicherung, um einen neuen Cluster zu erstellen. Weitere Informationen zur Erstellung eines Clusters aus einer Sicherung finden Sie unter [Cluster aus Sicherungen erstellen](#).

Sie können auch eine Sicherung eines Clusters in eine andere Region kopieren, in der Sie einen neuen Cluster als Klon des Originals erstellen können. Diese Lösung bietet sich für eine Reihe von Gründen an, einschließlich der Vereinfachung des Notfallwiederherstellungsprozesses. Weitere Informationen zum Kopieren von Sicherungen in Regionen finden Sie unter [Regionsübergreifendes Kopieren von Sicherungen](#).

Löschen Sie AWS CloudHSM Cluster-Backups

Nachdem Sie eine AWS CloudHSM Clustersicherung gelöscht haben, speichert der Dienst die Sicherung sieben Tage lang. Während dieser Zeit können Sie die Sicherung wiederherstellen. Nach Ablauf der sieben Tage können Sie das Backup nicht mehr wiederherstellen. Weitere Informationen zur Verwaltung von Sicherungen finden Sie unter [Sicherungen von Clustern](#).

In der folgenden Tabelle wird beschrieben, wie ein Backup gelöscht wird.

Console

So löschen Sie eine Sicherung (Konsole)

1. Öffnen Sie die AWS CloudHSM Konsole zu <https://console.aws.amazon.com/cloudhsm/Hause>.
2. Um die AWS-Region zu ändern, verwenden Sie die Regionsauswahl in der oberen rechten Ecke der Seite.
3. Wählen Sie im Navigationsbereich Sicherungen aus.
4. Wählen Sie eine Sicherung zum Löschen aus.
5. Um die ausgewählte Sicherung zu löschen, wählen Sie Aktionen, Löschen.

Das Dialogfeld „Sicherungen löschen“ wird angezeigt.

6. Wählen Sie Löschen aus.

Der Status des Backups ändert sich auf `PENDING_DELETE`. Sie können eine Sicherung, dessen Löschung noch aussteht, bis zu 7 Tage lang wiederherstellen, nachdem Sie den Löschvorgang beantragt haben.

Um Sicherungen aufzulisten (AWS CLI)

- Wenn Sie eine Liste aller Sicherungen mit dem Status `PENDING_DELETION` anzeigen wollen, führen Sie den Befehl `describe-backups` aus und schließen Sie als Filter `states=PENDING_DELETION` ein.

```
$ 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

Überprüfen Sie den Status einer Sicherung oder ermitteln Sie dessen ID, indem Sie den [describe-backups](#)-Befehl von der AWS CLI verwenden.

So löschen Sie eine Sicherung (AWS CLI)

- Führen Sie an der Eingabeaufforderung den Befehl [delete-backup](#) aus und übergeben Sie dabei die ID der Sicherung, die gelöscht werden soll.

```
$ 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

Weitere Informationen [DeleteBackup](#) zum Löschen von Backups mithilfe der API finden Sie unter.

AWS CloudHSM Backups wiederherstellen

AWS CloudHSM speichert gelöschte Backups sieben Tage lang. Während dieser Zeit können Sie die Sicherung wiederherstellen. Nach Ablauf der sieben Tage können Sie das Backup nicht mehr wiederherstellen. Weitere Informationen zur Verwaltung von Sicherungen finden Sie unter [Sicherungen von Clustern](#).

In der folgenden Tabelle wird beschrieben, wie ein Backup gelöscht wird.

Console

So stellen Sie eine Sicherung wieder her (Konsole)

1. Öffnen Sie die AWS CloudHSM Konsole zu <https://console.aws.amazon.com/cloudhsm/Hause>.
2. Um die AWS-Region zu ändern, verwenden Sie die Regionsauswahl in der oberen rechten Ecke der Seite.
3. Wählen Sie im Navigationsbereich Sicherungen aus.
4. Wählen Sie eine Sicherung in dem PENDING_DELETE-Zustand aus, den Sie wiederherstellen möchten.
5. Um das ausgewählte Backup wiederherzustellen, wählen Sie Aktionen, Wiederherstellen.

AWS CLI

So stellen Sie eine Sicherung wieder her (AWS CLI)

- Um eine Sicherung wiederherzustellen, geben Sie den [restore-backup](#) Befehl aus und übergeben Sie die ID einer Sicherung mit dem Status PENDING_DELETION.

```
$ aws cloudhsmv2 restore-backup --backup-id <backup ID>
{
  "Backup": {
    "ClusterId": "cluster-dygnwhmscg5",
    "CreateTimestamp": 1534461854.64,
    "BackupState": "READY",
    "BackupId": "backup-ro5c4er4aac"
  }
}
```

AWS CloudHSM API

Weitere Informationen [RestoreBackup](#) zum Wiederherstellen von Backups mithilfe der API finden Sie unter.

Konfigurieren Sie die Aufbewahrungsrichtlinie für AWS CloudHSM Backups

AWS CloudHSM löscht Backups auf der Grundlage der Backup-Aufbewahrungsrichtlinie, die Sie bei der Erstellung eines Clusters festgelegt haben. Die Aufbewahrungsrichtlinie für Sicherungen gilt für Cluster. Wenn Sie eine Sicherung in eine andere Region verschieben, ist diese Sicherung nicht mehr mit einem Cluster verknüpft und es gibt keine Aufbewahrungsrichtlinie für Sicherungen. Sie müssen alle Backups, die nicht mit einem Cluster verknüpft sind, manuell löschen. AWS CloudHSM löscht das letzte Backup eines Clusters nicht. AWS CloudHSM Behält bei Clustern, die von hsm1.medium migriert wurden, das neueste hsm1.medium-Backup bei, bis hsm1.medium das Ende seiner Lebensdauer erreicht hat.

[AWS CloudTrail](#) meldet Sicherungen, die zum Löschen markiert sind. Sie können Sicherungen, die der Service löscht, genauso wiederherstellen, wie Sie [manuell gelöschte Sicherungen](#) wiederherstellen würden. Um zu verhindern, dass es zu einem Race Condition kommt, sollten Sie die Aufbewahrungsrichtlinie für Sicherungen für den Cluster ändern, bevor Sie eine vom Service gelöschte Sicherung wiederherstellen. Wenn Sie die Aufbewahrungsrichtlinie beibehalten und ausgewählte Sicherungen beibehalten möchten, können Sie angeben, dass der Service [Sicherungen von der Aufbewahrungsrichtlinie für Cluster-Sicherungen ausschließt](#).

Weitere AWS CloudHSM Informationen zur [Senken Sie die Kosten, indem Sie sie an Ihre Bedürfnisse anpassen](#) Preisgestaltung finden Sie unter.

Verwaltete Aufbewahrung von Backups

Für Cluster, die vor dem 18. November 2020 erstellt wurden, gilt eine Aufbewahrungsrichtlinie für Backups von 90 Tagen zuzüglich des Alters des Clusters. Wenn Sie beispielsweise am 18. November 2019 einen Cluster erstellt haben, würde der Service Ihrem Cluster eine Backup-Aufbewahrungsrichtlinie von einem Jahr plus 90 Tagen (455 Tagen) zuweisen. Sie können für diesen Zeitraum eine beliebige Zahl zwischen 7 und 379 Tagen festlegen. AWS CloudHSM löscht das letzte Backup eines Clusters nicht. Weitere Informationen zur Verwaltung von Sicherungen finden Sie unter [Sicherungen von Clustern](#).

Note

Sie können die verwaltete Aufbewahrung von Backups vollständig deaktivieren, indem Sie sich an uns wenden [AWS Support](#).

In der folgenden Tabelle wird beschrieben, wie Sie die Aufbewahrung von Backups einrichten.

Console

So konfigurieren Sie die Aufbewahrungsrichtlinie für Sicherungen (Konsole)

1. Öffnen Sie die AWS CloudHSM Konsole zu <https://console.aws.amazon.com/cloudhsm/Hause>.
2. Um die AWS-Region zu ändern, verwenden Sie die Regionsauswahl in der oberen rechten Ecke der Seite.
3. Klicken Sie auf die Cluster-ID eines Clusters im Status Aktiv, um die Sicherungsaufbewahrungsrichtlinie für diesen Cluster zu verwalten.
4. Um die Aufbewahrungsrichtlinie für Sicherungen zu ändern, wählen Sie Aktionen, Aufbewahrungszeitraum für Sicherungen ändern.

Daraufhin wird das Dialogfeld Sicherungsdauer ändern angezeigt.

5. Geben Sie im Feld Aufbewahrungszeitraum für Sicherungen (in Tagen) einen Wert zwischen 7 und 379 Tagen ein.
6. Wählen Sie Aufbewahrungszeitraum für Sicherungen ändern aus.

Um eine Sicherung von der Aufbewahrungsrichtlinie für Sicherungen auszuschließen oder aufzunehmen (Konsole)

1. Öffne die AWS CloudHSM Konsole zu <https://console.aws.amazon.com/cloudhsm/Hause>.
2. Um Ihre Sicherungen anzusehen, wählen Sie im Navigationsbereich Sicherungen.
3. Klicken Sie auf die Sicherungs-ID eines Sicherungen im Status Bereit, um es auszuschließen oder einzubeziehen.
4. Führen Sie auf der Seite Sicherungsdetail eine der folgenden Aktionen aus.
 - Um eine Sicherung mit einem Datum als Ablaufzeit auszuschließen, wählen Sie Aktionen, Ablauf deaktivieren.
 - Um eine Sicherung einzubeziehen, das nicht abläuft, wählen Sie Aktionen, Cluster-Aufbewahrungsrichtlinie verwenden aus.

AWS CLI

So konfigurieren Sie die Aufbewahrungsrichtlinie für Sicherungen (AWS CLI)

- Geben Sie in der Eingabeaufforderung den Befehl `modify-cluster` ein. Geben Sie die Cluster-ID und die Aufbewahrungsrichtlinie für Sicherungen an.

```
$ 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"
}
```


Um eine Sicherung von der Aufbewahrungsrichtlinie für Sicherungen auszuschließen (AWS CLI)

- Geben Sie in der Eingabeaufforderung den Befehl `modify-backup-attributes` ein. Geben Sie die Sicherungs-ID an und setzen Sie das Never-Ablauf-Flag, um die Sicherung beizubehalten.

```
$ aws cloudhsmv2 modify-backup-attributes --backup-id <backup ID> \
                                         --never-expires
{
  "Backup": {
    "BackupId": "backup-ro5c4er4aac",
    "BackupState": "READY",
    "ClusterId": "cluster-dygnwhmscg5",
    "NeverExpires": true
  }
}
```

Um eine Sicherung in die Aufbewahrungsrichtlinie für Sicherungen aufzunehmen (AWS CLI)

- Geben Sie in der Eingabeaufforderung den Befehl `modify-backup-attributes` ein. Geben Sie die Backup-ID an und setzen Sie das `no-never-expires` Kennzeichen so, dass das Backup in die Backup-Aufbewahrungsrichtlinie aufgenommen wird, was bedeutet, dass der Service das Backup irgendwann löscht.

```
$ 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

In den folgenden Themen erfahren Sie, wie Sie die Aufbewahrung von Sicherungen mithilfe der API verwalten.

- [ModifyCluster](#)
- [ModifyBackupAttributes](#)

AWS CloudHSM Cluster-Backups zwischen AWS Regionen kopieren

Sie können AWS CloudHSM Cluster-Backups aus vielen Gründen regionsübergreifend kopieren, z. B. für regionsübergreifende Ausfallsicherheit, globale Workloads und [Notfallwiederherstellung](#). Nachdem Sie Sicherungen kopiert haben, werden sie in der Zielregion mit einem CREATE_IN_PROGRESS-Status angezeigt. Nach erfolgreichem Abschluss des Kopiervorgangs ändert sich der Status der Sicherung in READY. Wenn die Kopie fehlschlägt, ändert sich der Status der Sicherung auf DELETED. Überprüfen Sie Ihre Eingabeparameter auf Fehler und stellen Sie sicher, dass sich die angegebene Quellsicherung nicht in einem DELETED-Status befindet, bevor Sie den Vorgang erneut ausführen. Informationen über Sicherungen oder wie Sie einen Cluster aus einer Sicherung erstellen, finden Sie unter [Sicherungen von Clustern](#) oder [Cluster aus Sicherungen erstellen](#).

Beachten Sie Folgendes:

- Um eine Cluster-Sicherung in eine Zielregion zu kopieren, muss Ihr Konto über die entsprechenden IAM-Richtlinienberechtigungen verfügen. Damit Sie die Sicherung in eine andere Region kopieren können, muss Ihre IAM-Richtlinie den Zugriff auf die Quellregion erlauben, in der sich die Sicherung befindet. Sobald regionsübergreifend kopiert wurde, muss Ihre IAM-Richtlinie den Zugriff auf die Zielregion erlauben, um mit der kopierten Sicherung zu interagieren, welches die Verwendung der Operation [CreateCluster](#) einschließt. Weitere Informationen finden Sie unter [Erstellen von IAM-Administratoren](#).
- Der ursprüngliche Cluster und der Cluster, der möglicherweise aus einer Sicherung in der Zielregion erstellt wird, werden nicht verknüpft. Sie müssen jeden dieser Cluster unabhängig voneinander verwalten. Weitere Informationen finden Sie unter [Cluster](#).
- Backups können nicht zwischen AWS Partitionen kopiert werden. Backups können zwischen AWS eingeschränkten Regionen und Standardregionen innerhalb derselben Partition kopiert werden.

Sicherungen in verschiedene Regionen kopieren (Konsole)

Um Sicherungen in verschiedene Regionen zu kopieren (Konsole)

1. Öffnen Sie die AWS CloudHSM Konsole zu <https://console.aws.amazon.com/cloudhsm/Hause>.
2. Um die AWS-Region zu ändern, verwenden Sie die Regionsauswahl in der oberen rechten Ecke der Seite.
3. Wählen Sie im Navigationsbereich Sicherungen aus.
4. Wählen Sie eine Sicherung zum Kopieren in eine andere Region.
5. Um die ausgewählte Sicherung zu kopieren, wählen Sie Aktionen, Sicherung in eine andere Region kopieren.

Das Dialogfenster „Sicherung in eine andere Region kopieren“ wird angezeigt.

6. Wählen Sie im Feld Zielregion eine Region von Eine Region auswählen aus.
7. (Optional) Geben Sie einen Tag-Schlüssel und einen optionalen Tag-Wert ein. Wählen Sie Tag hinzufügen, wenn Sie mehr als ein Tag zum Cluster hinzufügen möchten.
8. Klicken Sie auf Copy backup (Backup kopieren).

Sicherungen in verschiedene Regionen kopieren (AWS CLI)

Um die Sicherungs-ID zu ermitteln, führen Sie den [describe-backups](#)-Befehl aus.

Um Sicherungen in verschiedene Regionen zu kopieren (AWS CLI)

- Führen Sie über die Eingabeaufforderung den folgenden [copy-backup-to-region](#)-Befehl aus. Geben Sie die Zielregion und die Sicherungs-ID der Sicherung an, von der die Sicherung stammt. Wenn Sie eine Sicherungs-ID angeben, wird die zugehörige Sicherung kopiert.

```
$ aws cloudhsmv2 copy-backup-to-region --destination-region <destination region> \
                                     --backup-id <backup ID>
```

Backups in verschiedene Regionen kopieren (AWS CloudHSM API)

Im folgenden Thema erfahren Sie, wie Sie Sicherungen mithilfe der API in verschiedene Regionen kopieren.

- [CopyBackupToRegion](#)

Arbeiten mit gemeinsam genutzten Backups in AWS CloudHSM

CloudHSM lässt sich in AWS Resource Access Manager (AWS RAM) integrieren, um die gemeinsame Nutzung von Ressourcen zu ermöglichen. AWS RAM ist ein Dienst, der es Ihnen ermöglicht, einige CloudHSM-Ressourcen mit anderen zu teilen AWS-Konten oder über AWS Organizations. Mit teilen Sie Ressourcen AWS RAM, die Sie besitzen, indem Sie eine gemeinsame Nutzung erstellen. Eine Ressourcenfreigabe legt die freizugebenden Ressourcen und die Konsumenten fest, für die sie freigegeben werden sollen. Zu den Verbrauchern können folgende Angaben zählen:

- AWS-Konten Spezifisch innerhalb oder außerhalb der Organisation in AWS Organizations
- Eine Organisationseinheit innerhalb ihrer Organisation in AWS Organizations
- Eine ganze Organisation in AWS Organizations

Weitere Informationen zu AWS RAM finden Sie im [AWS RAM Benutzerhandbuch](#).

In diesem Thema wird erklärt, wie Sie Ressourcen teilen, deren Eigentümer Sie sind, und wie Sie Ressourcen verwenden, die mit Ihnen gemeinsam genutzt werden.

Inhalt

- [Voraussetzungen für die gemeinsame Nutzung von Backups](#)
- [Ein Backup teilen](#)
- [Die gemeinsame Nutzung eines gemeinsam genutzten Backups rückgängig machen](#)
- [Identifizieren eines gemeinsam genutzten Backups](#)
- [Berechtigungen für gemeinsam genutzte Backups](#)
- [Fakturierung und Messung](#)

Voraussetzungen für die gemeinsame Nutzung von Backups

- Um ein Backup mit anderen zu teilen, müssen Sie es in Ihrem besitzen AWS-Konto. Das bedeutet, dass die Ressource Ihrem Konto zugewiesen oder bereitgestellt werden muss. Sie können ein Backup, das mit Ihnen geteilt wurde, nicht teilen.
- Um ein Backup gemeinsam zu nutzen, muss es sich im Status READY befinden.

- Um ein Backup mit Ihrer Organisation oder einer Organisationseinheit in zu teilen AWS Organizations, müssen Sie das Teilen mit aktivieren AWS Organizations. Weitere Informationen finden Sie unter [Freigabe für AWS Organizations aktivieren](#) im AWS RAM -Benutzerhandbuch.

Ein Backup teilen

Wenn Sie ein Backup mit anderen teilen AWS-Konten, ermöglichen Sie ihnen, Cluster aus dem Backup wiederherzustellen, die die im Backup gespeicherten Schlüssel und Benutzer enthalten.

Um ein Backup gemeinsam zu nutzen, müssen Sie es zu einer Ressourcenfreigabe hinzufügen. Eine Ressourcenfreigabe ist eine AWS RAM -Ressource, mit der Sie Ihre Ressourcen in mehreren AWS-Konten gemeinsam nutzen können. Eine Ressourcenfreigabe gibt die freizugebenden Ressourcen und die Konsumenten an, für die sie freigegeben werden. Wenn Sie ein Backup über die CloudHSM-Konsole teilen, fügen Sie es einer vorhandenen Ressourcenfreigabe hinzu. [Um das Backup zu einer neuen Ressourcenfreigabe hinzuzufügen, müssen Sie zuerst die Ressourcenfreigabe mithilfe der AWS RAM Konsole erstellen.](#)

Wenn Sie Teil einer Organisation in Ihrer Organisation sind AWS Organizations und die gemeinsame Nutzung innerhalb Ihrer Organisation aktiviert ist, erhalten Verbraucher in Ihrer Organisation automatisch Zugriff auf das gemeinsam genutzte Backup. Andernfalls erhalten Verbraucher eine Einladung zur Teilnahme an Resource Share und erhalten nach Annahme der Einladung Zugriff auf das gemeinsam genutzte Backup.

Sie können ein Backup, das Ihnen gehört, über die AWS RAM Konsole oder teilen AWS CLI.

Um ein Backup, das Ihnen gehört, über die AWS RAM Konsole zu teilen

Siehe [Erstellen einer Ressourcenfreigabe](#) im AWS RAM -Benutzerhandbuch.

Um ein Backup, das Ihnen gehört, mit anderen zu teilen (AWS RAM Befehl)

Verwenden Sie den Befehl [create-resource-share](#).

Um ein Backup zu teilen, das Ihnen gehört (CloudHSM-Befehl)

Important

Sie können ein Backup zwar mithilfe der PutResourcePolicy CloudHSM-Operation teilen, wir empfehlen jedoch, stattdessen AWS Resource Access Manager (AWS RAM) zu verwenden. Die Verwendung AWS RAM bietet mehrere Vorteile, da sie die Richtlinie für Sie erstellt, die gleichzeitige gemeinsame Nutzung mehrerer Ressourcen ermöglicht und die

Auffindbarkeit gemeinsam genutzter Ressourcen verbessert. Wenn Sie Backups verwenden PutResourcePolicy und möchten, dass Verbraucher die Möglichkeit haben, die Backups zu beschreiben, die Sie mit ihnen geteilt haben, müssen Sie das Backup mithilfe der AWS RAM PromoteResourceShareCreatedFromPolicy API-Operation auf eine standardmäßige AWS RAM Resource Share hochstufen.

Verwenden Sie den Befehl [put-resource-policy](#).

1. Erstellen Sie eine Datei mit dem Namen `policy.json` und kopieren Sie die folgende Richtlinie hinein.

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. Aktualisieren Sie `policy.json` mit dem Backup-ARN und den Kennungen, mit denen Sie ihn teilen möchten. Im folgenden Beispiel wird dem Root-Benutzer nur Lesezugriff für das durch 123456789012 identifizierte AWS Konto gewährt.

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

Sie können Berechtigungen nur auf Kontoebene gewähren. DescribeBackups Wenn Sie ein Backup mit einem anderen Kunden teilen, kann jeder Principal, der über die entsprechenden DescribeBackups Rechte für dieses Konto verfügt, das Backup beschreiben.

3. Führen Sie den Befehl [put-resource-policy](#) aus.

```
$ aws cloudhsmv2 put-resource-policy --resource-arn <resource-arn> --policy file://policy.json
```

 Note

Zu diesem Zeitpunkt kann der Kunde das Backup verwenden, es wird jedoch nicht in der DescribeBackups Antwort mit dem gemeinsam genutzten Parameter angezeigt. In den nächsten Schritten wird beschrieben, wie die gemeinsame Nutzung von AWS RAM Ressourcen gefördert werden kann, damit das Backup in die Antwort aufgenommen wird.

4. Holen Sie sich den AWS RAM Resource Share ARN.

```
$ aws ram list-resources --resource-owner SELF --resource-arns <backup-arn>
```

Dies gibt eine Antwort zurück, die der folgenden ähnelt:

```
{
  "resources": [
    {
      "arn": "<project-arn>",
      "type": "<type>",
      "resourceShareArn": "<resource-share-arn>",
      "creationTime": "<creation-time>",
      "lastUpdatedTime": "<last-update-time>"
    }
  ]
}
```

Kopieren Sie aus der Antwort den **<resource-share-arn>** Wert, der in den nächsten Schritten verwendet werden soll.

5. Führen Sie den Befehl AWS RAM [promote-resource-share-created-from-policy](#) aus.

```
$ aws ram promote-resource-share-created-from-policy --resource-share-arn <resource-share-arn>
```

6. Um zu überprüfen, ob die Ressourcenfreigabe heraufgestuft wurde, können Sie den Befehl ausführen. AWS RAM [get-resource-shares](#)

```
$ aws ram get-resource-shares --resource-owner SELF --resource-share-arns <resource-share-arn>
```

Wenn die Richtlinie heraufgestuft wurde, wird in der Antwort Folgendes `featureSet` aufgeführt `STANDARD`. Das bedeutet auch, dass das Backup anhand der neuen Konten in der Richtlinie beschrieben werden kann.

Die gemeinsame Nutzung eines gemeinsam genutzten Backups rückgängig machen

Wenn Sie die gemeinsame Nutzung einer Ressource aufheben, kann der Verbraucher sie nicht mehr zur Wiederherstellung eines Clusters verwenden. Verbraucher können weiterhin auf alle Cluster zugreifen, die sie aus dem gemeinsam genutzten Backup wiederhergestellt haben.

Um die gemeinsame Nutzung eines Backups, das Ihnen gehört, rückgängig zu machen, müssen Sie es aus der Ressourcenfreigabe entfernen. Sie können dies über die AWS RAM Konsole oder AWS CLI tun.

Um die Freigabe eines gemeinsam genutzten Backups, das Ihnen gehört, mithilfe der AWS RAM Konsole rückgängig zu machen

Siehe [Aktualisieren einer Ressourcenfreigabe](#) im AWS RAM -Benutzerhandbuch.

So machen Sie die gemeinsame Nutzung eines Backups rückgängig, dessen Eigentümer Sie sind (Befehl)AWS RAM

Verwenden Sie den Befehl [disassociate-resource-share](#).

So heben Sie die Freigabe eines geteilten Backups auf, das Ihnen gehört (CloudHSM-Befehl)

Verwenden Sie den Befehl [delete-resource-policy](#).

```
$ aws cloudhsmv2 delete-resource-policy --resource-arn <resource-arn>
```

Identifizieren eines gemeinsam genutzten Backups

Verbraucher können ein mit ihnen geteiltes Backup über die CloudHSM-Konsole identifizieren und. AWS CLI

Um Backups zu identifizieren, die mit Ihnen über die CloudHSM-Konsole geteilt wurden

1. [Öffnen Sie die AWS CloudHSM Konsole zu Hause. https://console.aws.amazon.com/cloudhsm/](https://console.aws.amazon.com/cloudhsm/)
2. Um das zu ändern AWS-Region, verwenden Sie die Regionsauswahl in der oberen rechten Ecke der Seite.
3. Wählen Sie im Navigationsbereich Sicherungen aus.
4. Wählen Sie in der Tabelle den Tab Gemeinsam genutzte Backups aus.

Um Backups zu identifizieren, die mit Ihnen geteilt wurden, verwenden Sie AWS CLI

Verwenden Sie den Befehl [describe-backups](#) mit dem `--shared` Parameter, um die Backups zurückzugeben, die für Sie freigegeben wurden.

Berechtigungen für gemeinsam genutzte Backups

Berechtigungen für Besitzer

Backup-Besitzer können ein gemeinsam genutztes Backup beschreiben und verwalten sowie es zur Wiederherstellung eines Clusters verwenden.

Berechtigungen für Konsumenten

Backup-Benutzer können ein gemeinsam genutztes Backup nicht ändern, aber sie können es beschreiben und zur Wiederherstellung eines Clusters verwenden.

Fakturierung und Messung

Für die gemeinsame Nutzung von Backups fallen keine zusätzlichen Gebühren an.

Geklonte Cluster in AWS CloudHSM

Verwenden Sie die CloudHSM-CLI, um einen Cluster in einer Remote-Region zu synchronisieren, wenn der Cluster in dieser Region ursprünglich aus dem Backup eines Clusters in einer anderen Region erstellt wurde. Nehmen wir an, Sie haben einen Cluster in eine andere Region (Ziel) kopiert und möchten später Änderungen aus dem ursprünglichen Cluster (Quelle) synchronisieren. In solchen Szenarien verwenden Sie die Befehle [key replicate](#) und [user replicate](#), um die Cluster zu synchronisieren. Wenn Sie CloudHSM CLI nicht installiert haben, lesen Sie die Anweisungen unter [???](#)

Verwandte Themen

- [Benutzer replizieren](#)
- [Schlüssel replizieren](#)
- [Regionsübergreifendes Kopieren von Sicherungen](#)

AWS CloudHSM Ressourcen taggen

Ein Tag ist eine Bezeichnung, die Sie einer AWS Ressource zuweisen. Sie können Ihren AWS CloudHSM -Clustern Tags zuweisen. Jedes Tag besteht aus einem Tag-Schlüssel und einem Tag-Wert, die Sie beide selbst definieren können. Der Tag-Schlüssel kann beispielsweise Kostenstelle und der Tag-Wert 12345 sein. Tag-Schlüssel müssen für jeden Cluster eindeutig sein.

Sie können Tags für eine Vielzahl von Zwecken verwenden. Sie können damit Ihre AWS -Kosten kategorisieren und verfolgen. Sie können Tags anwenden, die geschäftliche Kategorien (wie Kostenstellen, Anwendungsnamen oder Eigentümer) darstellen, um die Kosten für mehrere Services zu organisieren. Wenn Sie Ihren AWS Ressourcen Tags hinzufügen, AWS wird ein Kostenzuordnungsbericht generiert, in dem die Nutzung und die Kosten nach Tags zusammengefasst sind. Sie können diesen Bericht verwenden, um Ihre AWS CloudHSM Kosten in Form von Projekten oder Anwendungen anzuzeigen, anstatt alle AWS CloudHSM Kosten als einzelne Position anzuzeigen.

Weitere Informationen zur Verwendung von Tags für die Kostenzuordnung finden Sie unter [Verwenden von Kostenzuordnungs-Tags](#) im AWS Billing -Benutzerhandbuch.

Sie können die [AWS CloudHSM Konsole](#) oder eines der [AWS SDKs Befehlszeilentools](#) verwenden, um Tags hinzuzufügen, zu aktualisieren, aufzulisten und zu entfernen.

Themen

- [Fügen Sie Tags für AWS CloudHSM Ressourcen hinzu oder aktualisieren Sie sie](#)
- [Listet Tags für AWS CloudHSM Ressourcen auf](#)
- [Tags aus AWS CloudHSM Ressourcen entfernen](#)

Fügen Sie Tags für AWS CloudHSM Ressourcen hinzu oder aktualisieren Sie sie

Sie können Tags über die [AWS CloudHSM -Konsole](#), die [AWS Command Line Interface \(AWS CLI\)](#) oder die AWS CloudHSM -API hinzufügen oder aktualisieren.

So fügen Sie Tags hinzu oder aktualisieren sie (Konsole)

1. Öffne die AWS CloudHSM Konsole zu <https://console.aws.amazon.com/cloudhsm/Hause>.

2. Wählen Sie den Cluster aus, den Sie mit Tags versehen möchten.
3. Wählen Sie Tags aus.
4. Führen Sie die folgenden Schritte aus, um ein Tag hinzuzufügen:
 - a. Wählen Sie Tag bearbeiten und dann Tag hinzufügen aus.
 - b. Geben Sie unter Key (Schlüssel) einen Schlüssel für das Tag ein.
 - c. (Optional) Geben Sie unter Value (Wert) einen Wert für das Tag ein.
 - d. Wählen Sie Speichern.
5. Führen Sie die folgenden Schritte aus, um ein Tag zu aktualisieren:
 - a. Wählen Sie Tag bearbeiten.

 Note

Wenn Sie den Tag-Schlüssel für ein vorhandenes Tag aktualisieren, löscht die Konsole das vorhandene Tag und erstellt ein neues.

- b. Geben Sie den neuen Tag-Wert ein.
- c. Wählen Sie Speichern.

So können Sie Tags hinzufügen oder aktualisieren (AWS CLI)

1. Geben Sie an der Eingabeaufforderung den Befehl [tag-resource](#) unter Angabe der Tags und der ID des Clusters ein, den Sie mit Tags versehen möchten. Wenn Sie die Cluster-ID nicht kennen, geben Sie den Befehl [describe-clusters](#) ein.

```
$ aws cloudhsmv2 tag-resource --resource-id <cluster ID> \  
--tag-list Key="<tag key>",Value="<tag value>"
```

2. Verwenden Sie zum Aktualisieren von Tags den gleichen Befehl, geben Sie jedoch einen vorhandenen Tag-Schlüssel an. Wenn Sie einen neuen Tag-Wert für ein vorhandenes Tag angeben, wird das Tag mit dem neuen Wert überschrieben.

Um Tags hinzuzufügen oder zu aktualisieren (AWS CloudHSM API)

- Senden Sie eine [TagResource](#)-Anforderung. Geben Sie die Tags und die ID des Clusters an, den Sie mit Tags versehen möchten.

Listet Tags für AWS CloudHSM Ressourcen auf

Sie können Tags für einen Cluster über die [AWS CloudHSM Konsole](#) [AWS CLI](#), die oder die AWS CloudHSM API auflisten.

So listen Sie Tags auf (Konsole)

1. Öffnen Sie die AWS CloudHSM Konsole zu <https://console.aws.amazon.com/cloudhsm/Hause>.
2. Wählen Sie den Cluster, dessen Tags Sie auflisten möchten.
3. Wählen Sie Tags aus.

So listen Sie Tags (AWS CLI) auf

- Geben Sie an der Eingabeaufforderung den Befehl [list-tags](#) unter Angabe der ID des Clusters ein, dessen Tags Sie auflisten. Wenn Sie die Cluster-ID nicht kennen, geben Sie den Befehl [describe-clusters](#) ein.

```
$ aws cloudhsmv2 list-tags --resource-id <cluster ID>
{
  "TagList": [
    {
      "Key": "Cost Center",
      "Value": "12345"
    }
  ]
}
```

Um Tags aufzulisten (AWS CloudHSM API)

- Senden Sie eine [ListTags](#)-Anforderung, wobei Sie die ID des Clusters angeben, dessen Tags Sie auflisten.

Tags aus AWS CloudHSM Ressourcen entfernen

Sie können Tags mithilfe der [AWS CloudHSM Konsole](#), der oder der AWS CloudHSM API aus einem AWS CloudHSM Cluster entfernen. [AWS CLI](#)

So entfernen Sie Tags (Konsole)

1. Öffnen Sie die AWS CloudHSM Konsole zu <https://console.aws.amazon.com/cloudhsm/Hause>.
2. Wählen Sie den Cluster aus, dessen Tags Sie entfernen.
3. Wählen Sie Tags aus.
4. Wählen Sie für das Tag, das Sie entfernen möchten, Tag bearbeiten und dann Tag entfernen aus.
5. Wählen Sie Speichern.

So entfernen Sie Tags (AWS CLI)

- Geben Sie in der Eingabeaufforderung den Befehl [untag-resource](#) unter Angabe der Tag-Schlüssel der zu entfernenden Tags und der ID des Clusters ein, dessen Tags Sie entfernen möchten. Wenn Sie die AWS CLI zum Entfernen von Tags verwenden, geben Sie nur die Tag-Schlüssel an, nicht die Tag-Werte.

```
$ aws cloudhsmv2 untag-resource --resource-id <cluster ID> \  
                                --tag-key-list "<tag key>"
```

Um Tags zu entfernen (AWS CloudHSM API)

- Senden Sie eine [UntagResource](#)Anfrage in der AWS CloudHSM API und geben Sie die ID des Clusters und die Tags an, die Sie entfernen möchten.

AWS CloudHSM Befehlszeilentools

Zusätzlich zu den AWS Command Line Interface (AWS CLI), die Sie für die Verwaltung Ihrer AWS-Ressourcen verwenden, AWS CloudHSM bietet es Befehlszeilentools zum Erstellen und Verwalten von Benutzern und Schlüsseln für das Hardware-Sicherheitsmodul (HSM) auf Ihrem. HSMs In verwenden Sie die vertraute CLI AWS CloudHSM, um Ihren Cluster zu verwalten, und die CloudHSM-Befehlszeilentools, um Ihr HSM zu verwalten.

Dies sind die verschiedenen Befehlszeilen-Tools:

Zur Verwaltung von Clustern HSMs

[HSMv2 Cloud-Befehle im AWS CLI](#) und [HSM2 PowerShell Cmdlets im Shell-Modul AWSPower](#)

- Diese Tools rufen AWS CloudHSM Cluster ab, erstellen, löschen und taggen und: HSMs
- Um die Befehle in [HSMv2 Cloud-Befehlen in der CLI](#) zu verwenden, müssen Sie sie [installieren](#) und [konfigurieren](#) AWS CLI.
- [HSM2 PowerShell Cmdlets im AWSPower Shell-Modul](#) sind in einem Windows-Modul und einem plattformübergreifenden PowerShell PowerShell Core-Modul verfügbar.

So verwalten Sie HSM-Benutzer

[CloudHSM-CLI](#)

- Verwenden Sie die [CloudHSM-CLI](#), um Benutzer zu erstellen, Benutzer zu löschen, Benutzer aufzulisten, Benutzerkennwörter zu ändern und die Benutzer-Multi-Faktor-Authentifizierung (MFA) zu aktualisieren. Es ist nicht in der AWS CloudHSM-Clientsoftware enthalten. Anleitungen zur Installation dieses Tools finden Sie unter [CloudHSM CLI installieren und konfigurieren](#).

Hilfstoools

Zwei Tools helfen Ihnen bei der Verwendung von Tools und AWS CloudHSM Softwarebibliotheken:

- Das [Configure-Tool](#) aktualisiert Ihre CloudHSM-Client-Konfigurationsdateien. Dies ermöglicht AWS CloudHSM die Synchronisation der HSMs in einem Cluster.

AWS CloudHSM bietet zwei Hauptversionen, und Client SDK 5 ist die neueste Version. Es bietet eine Reihe von Vorteilen gegenüber Client-SDK 3 (der vorherigen Serie).

- Mit [pkpspeed](#) wird die Leistung der HSM-Hardware unabhängig von Software-Bibliotheken gemessen.

Tools für frühere SDKs

Verwenden Sie das Key Management Tool (KMU), um symmetrische Schlüssel und asymmetrische Schlüsselpaare zu erstellen, zu löschen, zu importieren und zu exportieren:

- [key_mgmt_util](#). Dieses Tool ist Bestandteil der AWS CloudHSM -Clientsoftware.

Verwenden Sie das CloudHSM Management Tool (CMU), um HSM-Benutzer zu erstellen und zu löschen, einschließlich der Implementierung der Quorum-Authentifizierung von Benutzerverwaltungsaufgaben

- [cloudhsm_mgmt_util](#). Dieses Tool ist Bestandteil der AWS CloudHSM -Clientsoftware.

In den folgenden Themen werden die Befehlszeilentools näher beschrieben, die für die Verwaltung und Verwendung verfügbar sind. AWS CloudHSM

Themen

- [AWS CloudHSM Tool konfigurieren](#)
- [AWS CloudHSM Befehlszeilenschnittstelle \(CLI\)](#)
- [AWS CloudHSM Verwaltungsdienstprogramm \(CMU\)](#)
- [AWS CloudHSM Schlüsselverwaltungsprogramm \(KMU\)](#)

AWS CloudHSM Tool konfigurieren

AWS CloudHSM synchronisiert automatisch Daten zwischen allen Hardware-Sicherheitsmodulen (HSM) in einem Cluster. Das configure-Tool aktualisiert die HSM-Daten der von den Synchronisationsmechanismen verwendeten Konfigurationsdateien. Wird verwendet, um die HSM-Daten zu aktualisieren, bevor Sie die Befehlszeilentools verwenden, insbesondere wenn sich die HSMs im Cluster geändert haben.

AWS CloudHSM umfasst zwei Hauptversionen des Client-SDK:

- Client-SDK 5: Dies ist unser neuestes und standardmäßiges Client-SDK. Informationen zu den Vorteilen und Nutzen, die es bietet, finden Sie unter [Vorteile von AWS CloudHSM Client SDK 5](#).
- Client-SDK 3: Dies ist unser älteres Client-SDK. Es enthält einen vollständigen Satz von Komponenten für die Kompatibilität von plattform- und sprachbasierten Anwendungen und Verwaltungstools.

Anweisungen zur Migration von Client SDK 3 auf Client SDK 5 finden Sie unter [Migration von AWS CloudHSM Client SDK 3 zu Client SDK 5](#).

Themen

- [AWS CloudHSM Konfigurationstool für das Client-SDK 5](#)
- [AWS CloudHSM Konfigurationstool für das Client-SDK 3](#)

AWS CloudHSM Konfigurationstool für das Client-SDK 5

Verwenden Sie das AWS CloudHSM Client SDK 5-Konfigurationstool, um die clientseitigen Konfigurationsdateien zu aktualisieren.

Jede Komponente in Client-SDK 5 enthält ein Configure-Tool mit einer Bezeichnung der Komponente im Dateinamen des Configure-Tools. Die PKCS #11-Bibliothek für das Client-SDK 5 enthält beispielsweise ein Configure-Tool namens `configure-pkcs11` unter Linux oder `configure-pkcs11.exe` auf Windows.

Themen

- [AWS CloudHSM Konfigurationssyntax für das Client-SDK 5](#)
- [AWS CloudHSM Konfigurationsparameter für das Client-SDK 5](#)
- [AWS CloudHSM Konfigurationsbeispiele für das Client-SDK 5](#)
- [Bootstrap OpenSSL Anbieter](#)
- [Erweiterte Konfigurationen für das Client-SDK-5-Configure-Tool](#)
- [AWS CloudHSM Verwandte Themen zum Client-SDK 5](#)

AWS CloudHSM Konfigurationssyntax für das Client-SDK 5

In der folgenden Tabelle wird die Syntax der AWS CloudHSM Konfigurationsdateien für das Client SDK 5 veranschaulicht. Weitere Informationen zu den Parametern finden Sie unter [the section called "Parameters"](#).

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 Konfigurationsparameter für das Client-SDK 5

Im Folgenden finden Sie eine Liste von Parametern für die Konfiguration von AWS CloudHSM Client SDK 5.

-a *<ENI IP address>*

Fügt die angegebene IP-Adresse den Client-SDK 5-Konfigurationsdateien hinzu. Geben Sie eine beliebige ENI-IP-Adresse eines HSM aus dem Cluster ein. Weitere Informationen über die Verwendung dieser Option finden Sie unter [Bootstrap für Client-SDK 5](#).

Erforderlich: Ja

--hsm-ca-cert *<customerCA certificate file path>*

Pfad zu dem Verzeichnis, in dem das Zertifikat der Zertifizierungsstelle (CA) gespeichert ist, mit dem EC2-Client-Instances mit dem Cluster verbunden werden. Sie erstellen diese Datei, wenn Sie den Cluster initialisieren. Standardmäßig sucht das System am folgenden Speicherort nach dieser Datei:

Linux

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

Windows

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

Weitere Informationen zur Initialisierung des Clusters oder zum Platzieren des Zertifikats finden Sie unter [???](#) und [???](#).

Erforderlich: Nein

--cluster-id *<cluster ID>*

Führt einen DescribeClusters-Aufruf aus, um alle IP-Adressen der HSM-Elastic-Network-Schnittstelle (ENI) im Cluster mit der Cluster-ID zu finden. Das System fügt die ENI-IP-Adressen zu den AWS CloudHSM Konfigurationsdateien hinzu.

Note

Wenn Sie den `--cluster-id` Parameter von einer EC2-Instance innerhalb einer VPC verwenden, die keinen Zugriff auf das öffentliche Internet hat, müssen Sie einen VPC-

Schnittstellen-Endpunkt erstellen, mit dem Sie eine Verbindung herstellen können. AWS CloudHSM Weitere Informationen über VPC-Endpunkte finden Sie unter [???](#).

Erforderlich: Nein

`--endpoint <endpoint>`

Geben Sie den AWS CloudHSM API-Endpunkt an, der für den `DescribeClusters` Aufruf verwendet wird. Sie müssen diese Option in Kombination mit `--cluster-id` festlegen.

Erforderlich: Nein

`--region <region>`

Geben Sie die Region Ihres Clusters an. Sie müssen diese Option in Kombination mit `--cluster-id` festlegen.

Wenn Sie den `--region`-Parameter nicht angeben, wählt das System die Region aus, indem es versucht, die Umgebungsvariablen `AWS_DEFAULT_REGION` oder `AWS_REGION` zu lesen. Wenn diese Variablen nicht festgelegt sind, überprüft das System die Region, die Ihrem Profil in Ihrer AWS-Config-Datei zugeordnet ist (normalerweise `~/.aws/config`), sofern Sie in der `AWS_CONFIG_FILE`-Umgebungsvariable keine andere Datei angegeben haben. Wenn keine der oben genannten Optionen festgelegt ist, verwendet das System standardmäßig die `us-east-1`-Region.

Erforderlich: Nein

`--client-cert-hsm-tls <client certificate hsm tls path>`

Pfad zum Client-Zertifikat, das für die gegenseitige Authentifizierung zwischen TLS Client und HSM verwendet wird.

Verwenden Sie diese Option nur, wenn Sie mindestens einen Vertrauensanker auf HSM mit CloudHSM CLI registriert haben. Sie müssen diese Option in Kombination mit `--client-key-hsm-tls-file` festlegen.

Erforderlich: Nein

`--client-key-hsm-tls <client key hsm tls path>`

Pfad zum Client-Schlüssel, der für die gegenseitige Authentifizierung zwischen TLS Client und HSM verwendet wird.

Verwenden Sie diese Option nur, wenn Sie mindestens einen Vertrauensanker auf HSM mit CloudHSM CLI registriert haben. Sie müssen diese Option in Kombination mit `--client-cert-hsm-tls-file` festlegen.

Erforderlich: Nein

`--log-level` **<error / warn / info / debug / trace>**

Gibt die Mindestprotokollierungsebene an, die das System in die Protokolldatei schreiben soll. Jede Stufe umfasst die vorherigen Stufen, wobei Fehler die Mindeststufe und Trace die Höchststufe ist. Das heißt, wenn Sie Fehler angeben, schreibt das System nur Fehler in das Protokoll. Wenn Sie trace angeben, schreibt das System Fehler, Warnungen, Informations- (Info-) und Debugmeldungen in das Protokoll. Weitere Informationen finden Sie unter [Client-SDK-5-Protokollierung](#).

Erforderlich: Nein

`--log-rotation` **<daily / weekly>**

Gibt die Häufigkeit an, mit der das System Protokolle rotiert. Weitere Informationen finden Sie unter [Client-SDK-5-Protokollierung](#).

Erforderlich: Nein

`--log-Datei` **<file name with path>**

Gibt an, wo das System die Protokolldatei schreiben wird. Weitere Informationen finden Sie unter [Client-SDK-5-Protokollierung](#).

Erforderlich: Nein

`--Protokolltyp` **<term / file>**

Gibt an, ob das System das Protokoll in eine Datei oder ein Terminal schreibt. Weitere Informationen finden Sie unter [Client-SDK-5-Protokollierung](#).

Erforderlich: Nein

`-h | --help`

Zeigt Hilfe an.

Erforderlich: Nein

`--disable-key-availability-check`

Markierung, um das Quorum für die Schlüsselverfügbarkeit zu deaktivieren. Verwenden Sie dieses Flag, um anzugeben, dass das Schlüsselverfügbarkeitsquorum deaktiviert werden AWS CloudHSM soll und dass Sie Schlüssel verwenden können, die nur auf einem HSM im Cluster vorhanden sind. Weitere Hinweise zur Verwendung dieses Flags zum Festlegen eines Quorums für Schlüsselverfügbarkeit finden Sie unter [???](#).

Erforderlich: Nein

`--enable-key-availability-check`

Markierung, um das Quorum für die Schlüsselverfügbarkeit zu aktivieren. Verwenden Sie dieses Flag, um anzugeben, dass das Schlüsselverfügbarkeitsquorum verwendet werden AWS CloudHSM soll und dass Sie Schlüssel erst verwenden dürfen, wenn diese Schlüssel auf zwei HSMs im Cluster vorhanden sind. Weitere Hinweise zur Verwendung dieses Flags zum Festlegen eines Quorums für Schlüsselverfügbarkeit finden Sie unter [???](#).

Standardmäßig aktiviert.

Erforderlich: Nein

`-- -init disable-validate-key-at`

Verbessert die Leistung, indem angegeben wird, dass Sie einen Initialisierungsaufruf überspringen können, um die Berechtigungen für einen Schlüssel für nachfolgende Aufrufe zu überprüfen. Verwenden Sie es mit Bedacht.

Hintergrund: Einige Mechanismen in der PKCS #11-Bibliothek unterstützen mehrteilige Operationen, bei denen ein Initialisierungsaufruf überprüft, ob Sie den Schlüssel für nachfolgende Aufrufe verwenden können. Dies erfordert einen Bestätigungsaufruf an das HSM, was die Latenz des gesamten Vorgangs erhöht. Mit dieser Option können Sie den nachfolgenden Aufruf deaktivieren und möglicherweise die Leistung verbessern.

Erforderlich: Nein

`-- -init enable-validate-key-at`

Gibt an, dass Sie einen Initialisierungsaufruf verwenden sollten, um die Berechtigungen für einen Schlüssel für nachfolgende Aufrufe zu überprüfen. Dies ist die Standardoption. Verwenden Sie `enable-validate-key-at-init`, um diese Initialisierungsrufe wieder aufzunehmen, nachdem Sie mit `disable-validate-key-at-init` sie unterbrochen haben.

Erforderlich: Nein

AWS CloudHSM Konfigurationsbeispiele für das Client-SDK 5

Diese Beispiele zeigen, wie das Konfigurationstool für AWS CloudHSM Client SDK 5 verwendet wird.

Bootstrappen des Client-SDK 5

Example

In diesem Beispiel wird der `-a`-Parameter zur Aktualisierung der HSM-Daten für Client-SDK 5 verwendet. Um den `-a` Parameter verwenden zu können, benötigen Sie die IP-Adresse für einen der Parameter HSMs in Ihrem Cluster.

PKCS #11 library

So bootstrappen Sie eine Linux-EC2-Instance für Client-SDK 5

- Verwenden Sie das Konfigurationstool, um die IP-Adresse eines HSM in Ihrem Cluster anzugeben.

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

So bootstrappen Sie eine Windows-EC2-Instance für Client-SDK 5

- Verwenden Sie das Konfigurationstool, um die IP-Adresse eines HSM in Ihrem Cluster anzugeben.

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

OpenSSL Dynamic Engine

So bootstrappen Sie eine Linux-EC2-Instance für Client-SDK 5

- Verwenden Sie das Konfigurationstool, um die IP-Adresse eines HSM in Ihrem Cluster anzugeben.

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

OpenSSL Dynamic Engine Provider

So bootstrappen Sie eine Linux-EC2-Instance für Client-SDK 5

- Verwenden Sie das Konfigurationstool, um die IP-Adresse eines HSM in Ihrem Cluster anzugeben.

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

Key Storage Provider (KSP)

So bootstrappen Sie eine Windows-EC2-Instance für Client-SDK 5

- Verwenden Sie das Konfigurationstool, um die IP-Adresse eines HSM in Ihrem Cluster anzugeben.

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

JCE provider

So bootstrappen Sie eine Linux-EC2-Instance für Client-SDK 5

- Verwenden Sie das Konfigurationstool, um die IP-Adresse eines HSM in Ihrem Cluster anzugeben.

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

So bootstrappen Sie eine Windows-EC2-Instance für Client-SDK 5

- Verwenden Sie das Konfigurationstool, um die IP-Adresse eines HSM in Ihrem Cluster anzugeben.

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

CloudHSM CLI

So bootstrappen Sie eine Linux-EC2-Instance für Client-SDK 5

- Verwenden Sie das Configure-Tool, um die IP-Adresse der HSM(s) in Ihrem Cluster anzugeben.

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

So bootstrappen Sie eine Windows-EC2-Instance für Client-SDK 5

- Verwenden Sie das Configure-Tool, um die IP-Adresse der HSM(s) in Ihrem Cluster anzugeben.

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

Note

Sie können den `--cluster-id`-Parameter anstelle von `-a <HSM_IP_ADDRESSES>` verwenden. Informationen zu den Anforderungen für die Verwendung von `--cluster-id` finden Sie unter [AWS CloudHSM Konfigurationstool für das Client-SDK 5](#).

Weitere Informationen zum Parameter `-a` erhalten Sie unter [the section called "Parameters"](#).

Geben Sie den Cluster, die Region und den Endpunkt für Client-SDK 5 an

Example

In diesem Beispiel wird der `cluster-id`-Parameter verwendet, um das Client-SDK 5 durch einen `DescribeClusters`-Aufruf zu bootstrappen.

PKCS #11 library

So bootstrappen Sie eine Linux-EC2-Instance für Client-SDK 5 mit **cluster-id**

- Verwenden Sie die Cluster-ID `cluster-1234567`, um die IP-Adresse eines HSM in Ihrem Cluster anzugeben.

```
$ sudo /opt/cloudhsm/bin/configure-pkcs11 --cluster-id <cluster-1234567>
```

So bootstrappen Sie eine Windows-EC2-Instance für Client-SDK 5 mit **cluster-id**

- Verwenden Sie die Cluster-ID `cluster-1234567`, um die IP-Adresse eines HSM in Ihrem Cluster anzugeben.

```
PS C:\> & "C:\Program Files\Amazon\CloudHSM\bin\configure-pkcs11.exe" --cluster-id <cluster-1234567>
```

OpenSSL Dynamic Engine

So bootstrappen Sie eine Linux-EC2-Instance für Client-SDK 5 mit **cluster-id**

- Verwenden Sie die Cluster-ID `cluster-1234567`, um die IP-Adresse eines HSM in Ihrem Cluster anzugeben.

```
$ sudo /opt/cloudhsm/bin/configure-dyn --cluster-id <cluster-1234567>
```

Key Storage Provider (KSP)

So bootstrappen Sie eine Windows-EC2-Instance für Client-SDK 5 mit **cluster-id**

- Verwenden Sie die Cluster-ID `cluster-1234567`, um die IP-Adresse eines HSM in Ihrem Cluster anzugeben.

```
PS C:\> & "C:\Program Files\Amazon\CloudHSM\bin\configure-ksp.exe" --cluster-id <cluster-1234567>
```


JCE provider

So bootstrappen Sie eine Linux-EC2-Instance für Client-SDK 5 mit **cluster-id**

- Verwenden Sie die Cluster-ID `cluster-1234567`, um die IP-Adresse eines HSM in Ihrem Cluster anzugeben.

```
$ sudo /opt/cloudhsm/bin/configure-jce --cluster-id <cluster-1234567>
```

So bootstrappen Sie eine Windows-EC2-Instance für Client-SDK 5 mit **cluster-id**

- Verwenden Sie die Cluster-ID `cluster-1234567`, um die IP-Adresse eines HSM in Ihrem Cluster anzugeben.

```
PS C:\> & "C:\Program Files\Amazon\CloudHSM\bin\configure-jce.exe" --cluster-id <cluster-1234567>
```

CloudHSM CLI

So bootstrappen Sie eine Linux-EC2-Instance für Client-SDK 5 mit **cluster-id**

- Verwenden Sie die Cluster-ID `cluster-1234567`, um die IP-Adresse eines HSM in Ihrem Cluster anzugeben.

```
$ sudo /opt/cloudhsm/bin/configure-cli --cluster-id <cluster-1234567>
```

So bootstrappen Sie eine Windows-EC2-Instance für Client-SDK 5 mit **cluster-id**

- Verwenden Sie die Cluster-ID `cluster-1234567`, um die IP-Adresse eines HSM in Ihrem Cluster anzugeben.

```
PS C:\> & "C:\Program Files\Amazon\CloudHSM\bin\configure-cli.exe" --cluster-id <cluster-1234567>
```

Sie können die Parameter `--region` und `--endpoint` in Kombination mit dem `cluster-id`-Parameter verwenden, um anzugeben, wie das System den `DescribeClusters`-Aufruf

durchführt. Wenn sich die Region des Clusters beispielsweise von der Region unterscheidet, die als AWS-CLI-Standard konfiguriert ist, sollten Sie den `--region`-Parameter verwenden, um diese Region zu verwenden. Darüber hinaus haben Sie die Möglichkeit, den AWS CloudHSM API-Endpunkt anzugeben, der für den Anruf verwendet werden soll. Dies kann für verschiedene Netzwerkkonfigurationen erforderlich sein, z. B. für die Verwendung von VPC-Schnittstellenendpunkten, für die nicht den Standard-DNS-Hostnamen verwendet wird. AWS CloudHSM

PKCS #11 library

Um eine Linux-EC2-Instance mit einem benutzerdefinierten Endpunkt und einer benutzerdefinierten Region zu booten

- Verwenden Sie das Konfigurationstool, um die IP-Adresse eines HSM in Ihrem Cluster mit einer benutzerdefinierten Region und einem benutzerdefinierten Endpunkt anzugeben.

```
$ sudo /opt/cloudhsm/bin/configure-pkcs11 --cluster-id <cluster-1234567> --  
region <us-east-1> --endpoint <https://cloudhsmv2.us-east-1.amazonaws.com>
```

So booten Sie eine Windows-EC2-Instance mit einem Endpunkt und einer Region

- Verwenden Sie das Konfigurationstool, um die IP-Adresse eines HSM in Ihrem Cluster mit einer benutzerdefinierten Region und einem benutzerdefinierten Endpunkt anzugeben.

```
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

Um eine Linux-EC2-Instance mit einem benutzerdefinierten Endpunkt und einer benutzerdefinierten Region zu booten

- Verwenden Sie das Konfigurationstool, um die IP-Adresse eines HSM in Ihrem Cluster mit einer benutzerdefinierten Region und einem benutzerdefinierten Endpunkt anzugeben.

```
$ 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)

So booten Sie eine Windows-EC2-Instance mit einem Endpunkt und einer Region

- Verwenden Sie das Konfigurationstool, um die IP-Adresse eines HSM in Ihrem Cluster mit einer benutzerdefinierten Region und einem benutzerdefinierten Endpunkt anzugeben.

```
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

Um eine Linux-EC2-Instance mit einem benutzerdefinierten Endpunkt und einer benutzerdefinierten Region zu booten

- Verwenden Sie das Konfigurationstool, um die IP-Adresse eines HSM in Ihrem Cluster mit einer benutzerdefinierten Region und einem benutzerdefinierten Endpunkt anzugeben.

```
$ sudo /opt/cloudhsm/bin/configure-jce --cluster-id <cluster-1234567> --region <us-east-1> --endpoint <https://cloudhsmv2.us-east-1.amazonaws.com>
```

So booten Sie eine Windows-EC2-Instance mit einem Endpunkt und einer Region

- Verwenden Sie das Konfigurationstool, um die IP-Adresse eines HSM in Ihrem Cluster mit einer benutzerdefinierten Region und einem benutzerdefinierten Endpunkt anzugeben.

```
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

Um eine Linux-EC2-Instance mit einem benutzerdefinierten Endpunkt und einer benutzerdefinierten Region zu booten

- Verwenden Sie das Konfigurationstool, um die IP-Adresse eines HSM in Ihrem Cluster mit einer benutzerdefinierten Region und einem benutzerdefinierten Endpunkt anzugeben.

```
$ sudo /opt/cloudhsm/bin/configure-cli --cluster-id <cluster-1234567> --
region <us-east-1> --endpoint <https://cloudhsmv2.us-east-1.amazonaws.com>
```

So booten Sie eine Windows-EC2-Instance mit einem Endpunkt und einer Region

- Verwenden Sie das Konfigurationstool, um die IP-Adresse eines HSM in Ihrem Cluster mit einer benutzerdefinierten Region und einem benutzerdefinierten Endpunkt anzugeben.

```
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>
```

Weitere Hinweise zu den Parametern `--cluster-id`, `--region` und `--endpoint` finden Sie unter [the section called "Parameters"](#).

Aktualisieren Sie das Client-Zertifikat und den Schlüssel für die gegenseitige Authentifizierung zwischen TLS Client und HSM

Example

Dieses Beispiel zeigt, wie Sie die `--client-key-hsm-tls-file` Parameter `--client-cert-hsm-tls-file` und verwenden, um SSL neu zu konfigurieren, indem Sie einen benutzerdefinierten Schlüssel und ein SSL-Zertifikat für angeben AWS CloudHSM

PKCS #11 library

So verwenden Sie ein benutzerdefiniertes Zertifikat und einen Schlüssel für die gegenseitige Authentifizierung zwischen TLS Client und HSM mit Client SDK 5 unter Linux

1. Kopieren Sie Schlüssel und Zertifikat in das entsprechende Verzeichnis.

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

2. Verwenden Sie das Configure-Tool, um `ssl-client.pem` und `ssl-client.key` anzugeben.

```
$ 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>
```

So verwenden Sie ein benutzerdefiniertes Zertifikat und einen Schlüssel für die gegenseitige Authentifizierung zwischen TLS Client und HSM mit Client SDK 5 unter Windows

1. Kopieren Sie Schlüssel und Zertifikat in das entsprechende Verzeichnis.

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

2. Verwenden Sie mit einem PowerShell Interpreter das Konfigurationstool, um und anzugeben. `ssl-client.pem` `ssl-client.key`

```
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

Um ein benutzerdefiniertes Zertifikat und einen Schlüssel für die gegenseitige Authentifizierung zwischen TLS Client und HSM mit Client SDK 5 unter Linux zu verwenden

1. Kopieren Sie Schlüssel und Zertifikat in das entsprechende Verzeichnis.

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

2. Verwenden Sie das Configure-Tool, um `ssl-client.pem` und `ssl-client.key` anzugeben.

```
$ 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)

So verwenden Sie ein benutzerdefiniertes Zertifikat und einen Schlüssel für die gegenseitige Authentifizierung zwischen TLS Client und HSM mit Client SDK 5 unter Windows

1. Kopieren Sie Schlüssel und Zertifikat in das entsprechende Verzeichnis.

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

2. Verwenden Sie mit einem PowerShell Interpreter das Konfigurationstool, um und anzugeben.
ssl-client.pem ssl-client.key

```
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

Um ein benutzerdefiniertes Zertifikat und einen Schlüssel für die gegenseitige Authentifizierung zwischen TLS Client und HSM mit Client SDK 5 unter Linux zu verwenden

1. Kopieren Sie Schlüssel und Zertifikat in das entsprechende Verzeichnis.

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

2. Verwenden Sie das Configure-Tool, um ssl-client.pem und ssl-client.key anzugeben.

```
$ 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>
```

So verwenden Sie ein benutzerdefiniertes Zertifikat und einen Schlüssel für die gegenseitige Authentifizierung zwischen TLS Client und HSM mit Client SDK 5 unter Windows

1. Kopieren Sie Schlüssel und Zertifikat in das entsprechende Verzeichnis.

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

2. Verwenden Sie mit einem PowerShell Interpreter das Konfigurationstool, um und anzugeben. `ssl-client.pem` `ssl-client.key`

```
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

Um ein benutzerdefiniertes Zertifikat und einen Schlüssel für die gegenseitige Authentifizierung zwischen TLS Client und HSM mit Client SDK 5 unter Linux zu verwenden

1. Kopieren Sie Schlüssel und Zertifikat in das entsprechende Verzeichnis.

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

2. Verwenden Sie das Configure-Tool, um `ssl-client.pem` und `ssl-client.key` anzugeben.

```
$ 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>
```

So verwenden Sie ein benutzerdefiniertes Zertifikat und einen Schlüssel für die gegenseitige Authentifizierung zwischen TLS Client und HSM mit Client SDK 5 unter Windows

1. Kopieren Sie Schlüssel und Zertifikat in das entsprechende Verzeichnis.

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

2. Verwenden Sie mit einem PowerShell Interpreter das Konfigurationstool, um und anzugeben.
ssl-client.pem ssl-client.key

```
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>
```

Weitere Hinweise zu den Parametern `--client-cert-hsm-tls-file` und `--client-key-hsm-tls-file` finden Sie unter [the section called "Parameters"](#).

Deaktivieren Sie die Einstellungen für die Haltbarkeit von Client-Schlüsseln

Example

In diesem Beispiel wird der `--disable-key-availability-check`-Parameter verwendet, um die Einstellungen für die Haltbarkeit von Clientschlüsseln zu deaktivieren. Um einen Cluster mit einem einzigen HSM auszuführen, müssen Sie die Einstellungen für die Haltbarkeit von Client-Schlüsseln deaktivieren.

PKCS #11 library

Um die Haltbarkeit von Client-Schlüsseln für Client-SDK 5 unter Linux zu deaktivieren

- Verwenden Sie das Configure-Tool, um die Einstellungen für die Haltbarkeit von Client-Schlüsseln zu deaktivieren.

```
$ sudo /opt/cloudhsm/bin/configure-pkcs11 --disable-key-availability-check
```

Um die Haltbarkeit von Client-Schlüsseln für Client-SDK 5 unter Windows zu deaktivieren

- Verwenden Sie das Configure-Tool, um die Einstellungen für die Haltbarkeit von Client-Schlüsseln zu deaktivieren.


```
PS C:\> & "C:\Program Files\Amazon\CloudHSM\bin\configure-pkcs11.exe" --disable-key-availability-check
```

OpenSSL Dynamic Engine

Um die Haltbarkeit von Client-Schlüsseln für Client-SDK 5 unter Linux zu deaktivieren

- Verwenden Sie das Configure-Tool, um die Einstellungen für die Haltbarkeit von Client-Schlüsseln zu deaktivieren.

```
$ sudo /opt/cloudhsm/bin/configure-dyn --disable-key-availability-check
```

OpenSSL Dynamic Engine Provider

Um die Haltbarkeit von Client-Schlüsseln für Client-SDK 5 unter Linux zu deaktivieren

- Verwenden Sie das Configure-Tool, um die Einstellungen für die Haltbarkeit von Client-Schlüsseln zu deaktivieren.

```
$ sudo /opt/cloudhsm/bin/configure-openssl-provider --disable-key-availability-check
```

Key Storage Provider (KSP)

Um die Haltbarkeit von Client-Schlüsseln für Client-SDK 5 unter Windows zu deaktivieren

- Verwenden Sie das Configure-Tool, um die Einstellungen für die Haltbarkeit von Client-Schlüsseln zu deaktivieren.

```
PS C:\> & "C:\Program Files\Amazon\CloudHSM\bin\configure-ksp.exe" --disable-key-availability-check
```

JCE provider

Um die Haltbarkeit von Client-Schlüsseln für Client-SDK 5 unter Linux zu deaktivieren

- Verwenden Sie das Configure-Tool, um die Einstellungen für die Haltbarkeit von Client-Schlüsseln zu deaktivieren.

```
$ sudo /opt/cloudhsm/bin/configure-jce --disable-key-availability-check
```

Um die Haltbarkeit von Client-Schlüsseln für Client-SDK 5 unter Windows zu deaktivieren

- Verwenden Sie das Configure-Tool, um die Einstellungen für die Haltbarkeit von Client-Schlüsseln zu deaktivieren.

```
PS C:\> & "C:\Program Files\Amazon\CloudHSM\bin\configure-jce.exe" --disable-key-availability-check
```

CloudHSM CLI

Um die Haltbarkeit von Client-Schlüsseln für Client-SDK 5 unter Linux zu deaktivieren

- Verwenden Sie das Configure-Tool, um die Einstellungen für die Haltbarkeit von Client-Schlüsseln zu deaktivieren.

```
$ sudo /opt/cloudhsm/bin/configure-cli --disable-key-availability-check
```

Um die Haltbarkeit von Client-Schlüsseln für Client-SDK 5 unter Windows zu deaktivieren

- Verwenden Sie das Configure-Tool, um die Einstellungen für die Haltbarkeit von Client-Schlüsseln zu deaktivieren.

```
PS C:\> & "C:\Program Files\Amazon\CloudHSM\bin\configure-cli.exe" --disable-key-availability-check
```

Weitere Informationen zum Parameter `--disable-key-availability-check` erhalten Sie unter [the section called "Parameters"](#).

Protokollierungsoptionen verwalten

Example

Das Client-SDK 5 verwendet die Parameter `log-file`, `log-level`, `log-rotation` und `log-type`, um die Protokollierung zu verwalten.

Note

Um Ihr SDK für serverlose Umgebungen wie AWS Fargate oder AWS Lambda zu konfigurieren, empfehlen wir Ihnen, Ihren AWS CloudHSM Protokolltyp auf zu konfigurieren. term Die Client-Protokolle werden in der für diese Umgebung konfigurierten Protokollgruppe CloudWatch Logs ausgegeben `stderr` und dort erfasst.

PKCS #11 library

Standardspeicherort für Protokollspeicherort

- Wenn Sie keinen Speicherort für die Datei angeben, schreibt das System Protokolle an den folgenden Standardspeicherort:

Linux

```
/opt/cloudhsm/run/cloudhsm-pkcs11.log
```

Windows

```
C:\Program Files\Amazon\CloudHSM\cloudhsm-pkcs11.log
```

Um die Protokollierungsebene zu konfigurieren und andere Protokollierungsoptionen auf Standard zu setzen

- ```
$ sudo /opt/cloudhsm/bin/configure-pkcs11 --log-level info
```

Um Optionen für die Dateiprotokollierung zu konfigurieren

- ```
$ sudo /opt/cloudhsm/bin/configure-pkcs11 --log-type file --log-file <file name with path> --log-rotation daily --log-level info
```

Um Optionen für die Terminalprotokollierung zu konfigurieren

- ```
$ sudo /opt/cloudhsm/bin/configure-pkcs11 --log-type term --log-level info
```

## OpenSSL Dynamic Engine

Standardspeicherort für Protokollspeicherort

- Wenn Sie keinen Speicherort für die Datei angeben, schreibt das System Protokolle an den folgenden Standardspeicherort:

Linux

```
stderr
```

Um die Protokollierungsebene zu konfigurieren und andere Protokollierungsoptionen auf Standard zu setzen

- ```
$ sudo /opt/cloudhsm/bin/configure-dyn --log-level info
```

Um Optionen für die Dateiprotokollierung zu konfigurieren

- ```
$ sudo /opt/cloudhsm/bin/configure-dyn --log-type file --log-file <file name with path> --log-rotation daily --log-level info
```

Um Optionen für die Terminalprotokollierung zu konfigurieren

- ```
$ sudo /opt/cloudhsm/bin/configure-dyn --log-type term --log-level info
```

OpenSSL Dynamic Engine Provider

Standardspeicherort für Protokollspeicherort

- Wenn Sie keinen Speicherort für die Datei angeben, schreibt das System Protokolle an den folgenden Standardspeicherort:

Linux

```
stderr
```

Um die Protokollierungsebene zu konfigurieren und andere Protokollierungsoptionen auf Standard zu setzen

- ```
$ sudo /opt/cloudhsm/bin/configure-openssl-provider --log-level info
```

Um Optionen für die Dateiprotokollierung zu konfigurieren

- ```
$ sudo /opt/cloudhsm/bin/configure-openssl-provider --log-type file --log-file <file name with path> --log-rotation daily --log-level info
```

Um Optionen für die Terminalprotokollierung zu konfigurieren

- ```
$ sudo /opt/cloudhsm/bin/configure-openssl-provider --log-type term --log-level info
```

## Key Storage Provider (KSP)

### Standardspeicherort für Protokollspeicherort

- Wenn Sie keinen Speicherort für die Datei angeben, schreibt das System Protokolle an den folgenden Standardspeicherort:

#### Windows

```
C:\Program Files\Amazon\CloudHSM\cloudhsm-ksp.log
```

Um die Protokollierungsebene zu konfigurieren und andere Protokollierungsoptionen auf Standard zu setzen

- ```
PS C:\> & "C:\Program Files\Amazon\CloudHSM\bin\configure-ksp.exe" --log-level info
```

Um Optionen für die Dateiprotokollierung zu konfigurieren

- ```
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
```

Um Optionen für die Terminalprotokollierung zu konfigurieren

- ```
PS C:\> & "C:\Program Files\Amazon\CloudHSM\bin\configure-ksp.exe" --log-type term --log-level info
```

JCE provider

Standardspeicherort für Protokollspeicherort

- Wenn Sie keinen Speicherort für die Datei angeben, schreibt das System Protokolle an den folgenden Standardspeicherort:

Linux

```
/opt/cloudhsm/run/cloudhsm-jce.log
```

Windows

```
C:\Program Files\Amazon\CloudHSM\cloudhsm-jce.log
```

Um die Protokollierungsebene zu konfigurieren und andere Protokollierungsoptionen auf Standard zu setzen

- ```
$ sudo /opt/cloudhsm/bin/configure-jce --log-level info
```

Um Optionen für die Dateiprotokollierung zu konfigurieren

- ```
$ sudo /opt/cloudhsm/bin/configure-jce --log-type file --log-file <file name  
with path> --log-rotation daily --log-level info
```

Um Optionen für die Terminalprotokollierung zu konfigurieren

- ```
$ sudo /opt/cloudhsm/bin/configure-jce --log-type term --log-level info
```

## CloudHSM CLI

Standardspeicherort für Protokollspeicherort

- Wenn Sie keinen Speicherort für die Datei angeben, schreibt das System Protokolle an den folgenden Standardspeicherort:

Linux

```
/opt/cloudhsm/run/cloudhsm-cli.log
```

Windows

```
C:\Program Files\Amazon\CloudHSM\cloudhsm-cli.log
```

Um die Protokollierungsebene zu konfigurieren und andere Protokollierungsoptionen auf Standard zu setzen

- ```
$ sudo /opt/cloudhsm/bin/configure-cli --log-level info
```

Um Optionen für die Dateiprotokollierung zu konfigurieren

- ```
$ sudo /opt/cloudhsm/bin/configure-cli --log-type file --log-file <file name
with path> --log-rotation daily --log-level info
```

Um Optionen für die Terminalprotokollierung zu konfigurieren

- ```
$ sudo /opt/cloudhsm/bin/configure-cli --log-type term --log-level info
```

Weitere Informationen über die Parameter `log-file`, `log-level`, `log-rotation` und `log-type` finden Sie unter [the section called "Parameters"](#).

Platzieren Sie das ausstellende Zertifikat für das Client-SDK 5

Example

In diesem Beispiel wird der `--hsm-ca-cert`-Parameter verwendet, um den Speicherort des ausstellenden Zertifikats für das Client-SDK 5 zu aktualisieren.

PKCS #11 library

Um das ausstellende Zertifikat für das Client-SDK 5 auf Linux zu platzieren

- Verwenden Sie das Configure-Tool, um einen Speicherort für das ausstellende Zertifikat anzugeben.

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

Um das ausstellende Zertifikat unter Windows für Client-SDK 5 zu platzieren

- Verwenden Sie das Configure-Tool, um einen Speicherort für das ausstellende Zertifikat anzugeben.

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

OpenSSL Dynamic Engine

Um das ausstellende Zertifikat für das Client-SDK 5 auf Linux zu platzieren

- Verwenden Sie das Configure-Tool, um einen Speicherort für das ausstellende Zertifikat anzugeben.


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

OpenSSL Dynamic Engine Provider

Um das ausstellende Zertifikat für das Client-SDK 5 auf Linux zu platzieren

- Verwenden Sie das Configure-Tool, um einen Speicherort für das ausstellende Zertifikat anzugeben.

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

Key Storage Provider (KSP)

Um das ausstellende Zertifikat unter Windows für Client-SDK 5 zu platzieren

- Verwenden Sie das Configure-Tool, um einen Speicherort für das ausstellende Zertifikat anzugeben.

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

JCE provider

Um das ausstellende Zertifikat für das Client-SDK 5 auf Linux zu platzieren

- Verwenden Sie das Configure-Tool, um einen Speicherort für das ausstellende Zertifikat anzugeben.

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

Um das ausstellende Zertifikat unter Windows für Client-SDK 5 zu platzieren

- Verwenden Sie das Configure-Tool, um einen Speicherort für das ausstellende Zertifikat anzugeben.

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

CloudHSM CLI

Um das ausstellende Zertifikat für das Client-SDK 5 auf Linux zu platzieren

- Verwenden Sie das Configure-Tool, um einen Speicherort für das ausstellende Zertifikat anzugeben.

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

Um das ausstellende Zertifikat unter Windows für Client-SDK 5 zu platzieren

- Verwenden Sie das Configure-Tool, um einen Speicherort für das ausstellende Zertifikat anzugeben.

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

Weitere Informationen zum Parameter `--hsm-ca-cert` erhalten Sie unter [the section called "Parameters"](#).

Bootstrap OpenSSL Anbieter

Verwenden Sie das `configure-openssl-provider` Tool, um Ihre OpenSSL Provider-Installation zu booten und sie mit Ihrem AWS CloudHSM Cluster zu verbinden.

Um den OpenSSL-Anbieter zu booten

1. Führen Sie den `configure-openssl-provider` Befehl mit der IP-Adresse eines HSM in Ihrem Cluster aus:

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

<HSM IP address> Ersetzen Sie ihn durch die IP-Adresse eines beliebigen HSM in Ihrem Cluster.

- Überprüfen Sie die Konfiguration, indem Sie überprüfen, ob der OpenSSL-Anbieter eine Verbindung zu Ihrem Cluster herstellen kann:

```
$ openssl list -providers -provider-path /opt/cloudhsm/lib -provider cloudhsm
```

Weitere Hinweise zu den Konfigurationsparametern finden Sie unter [the section called “Parameters”](#).

Erweiterte Konfigurationen für das Client-SDK-5-Configure-Tool

Das AWS CloudHSM Client SDK 5-Konfigurationstool umfasst erweiterte Konfigurationen, die nicht zu den allgemeinen Funktionen gehören, die die meisten Kunden verwenden. Erweiterte Konfigurationen bieten zusätzliche Funktionen.

Important

Nachdem Sie Änderungen an Ihrer Konfiguration vorgenommen haben, müssen Sie Ihre Anwendung neu starten, damit die Änderungen wirksam werden.

- Erweiterte Konfigurationen für PKCS #11
 - [Konfiguration mit mehreren Steckplätzen mit PKCS #11 -Bibliothek für AWS CloudHSM](#)
 - [Wiederholen Sie die Befehle für die PKCS #11 -Bibliothek für AWS CloudHSM](#)
- Erweiterte Konfigurationen für OpenSSL
 - [Befehle für OpenSSL wiederholen für AWS CloudHSM](#)
- Erweiterte Konfigurationen für KSP
 - [SDK3 Kompatibilitätsmodus für Key Storage Provider \(KSP\) für AWS CloudHSM](#)
- Erweiterte Konfigurationen für JCE
 - [Verbindung zu mehreren AWS CloudHSM Clustern mit dem JCE-Anbieter herstellen](#)
 - [Befehle für JCE wiederholen für AWS CloudHSM](#)
 - [Schlüsselextraktion mit JCE für AWS CloudHSM](#)

- Erweiterte Konfigurationen für die AWS CloudHSM Befehlszeilenschnittstelle (CLI)
 - [Verbindung zu mehreren Clustern mit CloudHSM CLI herstellen](#)

AWS CloudHSM Verwandte Themen zum Client-SDK 5

Weitere Informationen zum AWS CloudHSM Client SDK 5 finden Sie in den folgenden verwandten Themen.

- [DescribeClusters](#) API-Operation
- [describe-clusters](#) AWS CLI
- [Get-HSM2Cluster](#) PowerShell Cmdlet
- [Bootstrappen des Client-SDK 5](#)
- [Bootstrap OpenSSL Anbieter](#)
- [AWS CloudHSM VPC-Endpunkte](#)
- [Verwaltung der wichtigsten Haltbarkeitseinstellungen des Client-SDK 5](#)
- [Protokollierung im Client-SDK 5](#)
- [mTLS einrichten \(empfohlen\)](#)

AWS CloudHSM Konfigurationstool für das Client-SDK 3

Verwenden Sie das AWS CloudHSM Client SDK 3-Konfigurationstool, um den Client-Daemon zu booten und das CloudHSM Management Utility (CMU) zu konfigurieren.

Themen

- [AWS CloudHSM Konfigurationssyntax für das Client-SDK 3](#)
- [AWS CloudHSM Konfigurationsparameter für das Client-SDK 3](#)
- [AWS CloudHSM Konfigurationsbeispiele für das Client-SDK 3](#)
- [AWS CloudHSM Themen rund um die Konfiguration des Client-SDK 3](#)

AWS CloudHSM Konfigurationssyntax für das Client-SDK 3

In der folgenden Tabelle wird die Syntax der AWS CloudHSM Konfigurationsdateien für das Client SDK 3 veranschaulicht.

```
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 Konfigurationsparameter für das Client-SDK 3

Im Folgenden finden Sie eine Liste von Parametern für die Konfiguration von AWS CloudHSM Client SDK 3.

-h | --help

Zeigt die Befehlssyntax an.

Erforderlich: Ja

-a **<ENI IP address>**

Fügt die angegebene IP-Adresse der HSM-Elastic Network-Schnittstelle (ENI) zu den Konfigurationsdateien von AWS CloudHSM hinzu. Geben Sie die ENI-IP-Adresse einer beliebigen Person HSMs im Cluster ein. Dabei spielt es keine Rolle, welche Sie auswählen.

Um die ENI-IP-Adressen der HSMs in Ihrem Cluster abzurufen, verwenden Sie den Befehl [DescribeClusters](#) operation, den AWS CLI Befehl [describe-clusters](#) oder das [Get-HSM2Cluster](#) PowerShell Cmdlet.

Note

Stoppen Sie den Client, bevor `-a configure` Sie den Befehl ausführen. AWS CloudHSM Wenn der `-a` Befehl abgeschlossen ist, starten Sie den AWS CloudHSM Client neu. Details finden Sie [in den Beispielen](#).

Dieser Parameter bearbeitet die folgenden Konfigurationsdateien:

- `/opt/cloudhsm/etc/cloudhsm_client.cfg`: Wird von AWS CloudHSM client und [key_mgmt_util](#) verwendet.
- `/opt/cloudhsm/etc/cloudhsm_mgmt_util.cfg`: Wird von [cloudhsm_mgmt_util](#) verwendet.

Wenn der AWS CloudHSM Client gestartet wird, verwendet er die ENI-IP-Adresse in seiner Konfigurationsdatei, um den Cluster abzufragen und die `cluster.info` Datei (`/opt/cloudhsm/daemon/1/cluster.info`) mit den richtigen ENI-IP-Adressen für alle Mitglieder des Clusters zu aktualisieren. HSMs

Erforderlich: Ja

-m

Aktualisiert die HSM-ENI-IP-Adressen in der von CMU verwendeten Konfigurationsdatei.

 Note

Der `-m`-Parameter ist für die Verwendung mit CMU aus Client-SDK 3.2.1 und früheren Versionen vorgesehen. Informationen zu CMU aus dem Client-SDK 3.3.0 und höher finden Sie unter `--cmu`-Parameter, der das Aktualisieren von HSM-Daten für CMU vereinfacht.

Wenn Sie den `-a` Parameter von `aktualisieren configure` und dann den AWS CloudHSM Client starten, fragt der Client-Daemon den Cluster ab und aktualisiert die `cluster.info` Dateien mit den richtigen HSM-IP-Adressen für alle HSMs im Cluster. Durch das Ausführen des Befehls `-m configure` wird die Aktualisierung abgeschlossen, da die HSM-IP-Adressen aus der Datei `cluster.info` in die von `cloudhsm_mgmt_util` verwendete `cloudhsm_mgmt_util.cfg`-Konfigurationsdatei kopiert werden.

Stellen Sie sicher, dass Sie den `-a configure` Befehl ausführen und den AWS CloudHSM Client neu starten, bevor Sie den `-m` Befehl ausführen. Dadurch wird sichergestellt, dass die Daten, die aus `cloudhsm_mgmt_util.cfg` in `cluster.info` kopiert werden, vollständig und korrekt sind.

Erforderlich: Ja

-i

Gibt einen alternativen Client-Daemon an. Der Standardwert stellt den AWS CloudHSM -Client dar.

Standard: 1

Erforderlich: Nein

--ssl

Ersetzt den SSL-Schlüssel und das Zertifikat für den Cluster durch den angegebenen privaten Schlüssel und das Zertifikat. Wenn Sie diesen Parameter verwenden, müssen die Parameter `--pkey` und `--cert` verwendet werden.

Erforderlich: Nein

--pkey

Gibt den neuen privaten Schlüssel an. Geben Sie den Pfad und den Namen der Datei an, die den privaten Schlüssel enthält.

Erforderlich: Ja, `--ssl` wenn angegeben. Andernfalls sollte dieser Wert nicht verwendet werden.

--cert

Gibt das neue Zertifikat an. Geben Sie den Pfad und den Namen der Datei an, die das Zertifikat enthält. Das Zertifikat sollte mit dem Zertifikat `customerCA.crt`, dem selbstsignierten Zertifikat, das zum Initialisieren des Clusters verwendet wurde, verkettet sein. Weitere Informationen finden Sie unter [Initialisieren des Clusters](#).

Erforderlich: Ja, `--ssl` wenn angegeben. Andernfalls sollte dieser Wert nicht verwendet werden.

--cmu *<ENI IP address>*

Kombiniert die Parameter `-a` und `-m` zu einem Parameter. Fügt den AWS CloudHSM Konfigurationsdateien die angegebene HSM elastic network interface (ENI) -IP-Adresse hinzu und aktualisiert dann die CMU-Konfigurationsdatei. Geben Sie eine IP-Adresse von einem beliebigen HSM im Cluster ein. Informationen zum Client-SDK 3.2.1 und früher finden Sie unter [Verwenden von CMU mit Client-SDK 3.2.1](#) und früher.

Erforderlich: Ja

AWS CloudHSM Konfigurationsbeispiele für das Client-SDK 3

Diese Beispiele zeigen, wie das `configure` Tool für AWS CloudHSM Client SDK 3 verwendet wird.

Example: Aktualisieren Sie die HSM-Daten für den AWS CloudHSM Client und `key_mgmt_util`

In diesem Beispiel werden die `-a` Parameter von `configure` zur Aktualisierung der HSM-Daten für den Client und `key_mgmt_util` verwendet. AWS CloudHSM Um den `-a` Parameter verwenden zu

können, benötigen Sie die IP-Adresse für einen der Parameter in Ihrem Cluster. HSMs verwenden Sie entweder die Konsole oder die AWS-CLI, um die IP-Adresse abzurufen.

Um eine IP-Adresse für ein HSM (Konsole) zu erhalten

1. Öffnen Sie die AWS CloudHSM Konsole zu <https://console.aws.amazon.com/cloudhsm/Hause>.
2. Um die AWS-Region zu ändern, verwenden Sie die Regionsauswahl in der oberen rechten Ecke der Seite.
3. Um die Cluster-Detailseite zu öffnen, wählen Sie in der Cluster-Tabelle die Cluster-ID aus.
4. Um die IP-Adresse zu erhalten, gehen Sie zur HSMs Registerkarte. Wählen Sie für IPv4 Cluster eine Adresse aus, die unter IPv4 ENI-Adresse aufgeführt ist. Verwenden Sie für Dual-Stack-Cluster entweder die ENI IPv4 - oder die IPv6 ENI-Adresse.

Um eine IP-Adresse für ein HSM zu erhalten (AWS CLI)

- Rufen Sie die IP-Adresse eines HSM ab, indem Sie den [describe-clusters](#) Befehl von der AWS CLI verwenden. In der Ausgabe des Befehls entspricht die IP-Adresse von den Werten von `EniIp` und `EniIpV6` (falls es HSMs sich um einen Dual-Stack-Cluster handelt).

```
$ 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",
...
      }
    ]
  }
}
```

So aktualisieren Sie die HSM-Daten

1. Stoppen Sie den AWS CloudHSM Client, bevor Sie den `-a` Parameter aktualisieren. Dies verhindert Konflikte, die bei der Bearbeitung der Client-Konfigurationsdatei durch `configure`

auftreten können. Wenn der Client bereits angehalten ist, hat dieser Befehl keine Auswirkungen, sodass Sie ihn in einem Skript verwenden können.

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

- Für Windows-Client 1.1.2 und höher:

```
C:\Program Files\Amazon\CloudHSM>net.exe stop AWSCloudHSMClient
```

- Für Windows-Clients 1.1.1 und früher:

Verwenden Sie Strg + C in dem Befehlsfenster, in dem Sie den AWS CloudHSM Client gestartet haben.

2. Dieser Schritt verwendet den -a-Parameter von configure, um den Konfigurationsdateien die ENI-IP-Adresse 10.0.0.9 hinzuzufügen.

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. Starten Sie als Nächstes den AWS CloudHSM Client neu. Wenn der -Client startet, wird die ENI IP-Adresse aus der Konfigurationsdatei des Clients verwendet, um den Cluster abzufragen. Anschließend werden die ENI-IP-Adressen aller Mitglieder des Clusters HSMs in die `cluster.info` Datei geschrieben.

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

- Für Windows-Client 1.1.2 und höher:

```
C:\Program Files\Amazon\CloudHSM>net.exe start AWSCloudHSMClient
```

- Für Windows-Clients 1.1.1 und früher:

```
C:\Program Files\Amazon\CloudHSM>start "cloudhsm_client" cloudhsm_client.exe  
C:\ProgramData\Amazon\CloudHSM\data\cloudhsm_client.cfg
```

Wenn der Befehl abgeschlossen ist, sind die HSM-Daten, die der AWS CloudHSM Client und `key_mgmt_util` verwenden, vollständig und korrekt.

Example: Aktualisieren Sie die HSM-Daten für CMU aus dem Client-SDK 3.2.1 und früher

In diesem Beispiel wird der `-m configure`-Befehl verwendet, um die aktualisierten HSM-Daten aus der `cluster.info`-Datei in die `cloudhsm_mgmt_util.cfg`-Datei zu kopieren, die `cloudhsm_mgmt_util` verwendet. Verwenden Sie dies mit der CMU, die im Client-SDK 3.2.1 und früher enthalten ist.

- [Bevor Sie den `ausführen-m`, beenden Sie den AWS CloudHSM Client, führen Sie den `-a` Befehl aus und starten Sie den AWS CloudHSM Client neu, wie im vorherigen Beispiel gezeigt.](#) Dadurch wird sichergestellt, dass die Daten, die aus der Datei `cloudhsm_mgmt_util.cfg` in die Datei `cluster.info` kopiert werden, vollständig und korrekt sind.

Linux

```
$ sudo /opt/cloudhsm/bin/configure -m
```

Windows

```
PS C:\> & "C:\Program Files\Amazon\CloudHSM\configure.exe" -m
```

Example: Aktualisieren Sie die HSM-Daten für CMU aus dem Client-SDK 3.3.0 und höher

In diesem Beispiel wird der `--cmu`-Parameter des `configure`-Befehls zur Aktualisierung der HSM-Daten für CMU verwendet. Verwenden Sie dies mit der CMU, die im Client-SDK 3.3.0 und höher enthalten ist. Weitere Informationen zur Verwendung von CMU finden Sie unter [Verwendung des CloudHSM Management Utility \(CMU\) zum Verwalten von Benutzern](#) und [Verwendung des CMU mit Client-SDK 3.2.1 und früher](#).

- Verwenden Sie den `--cmu`-Parameter, um die IP-Adresse eines HSM in Ihrem Cluster zu übergeben.

Linux

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

Windows

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

AWS CloudHSM Themen rund um die Konfiguration des Client-SDK 3

Weitere Informationen zum AWS CloudHSM Client SDK 3 finden Sie in den folgenden verwandten Themen.

- [Richten Sie AWS CloudHSM key_mgmt_util ein](#)

AWS CloudHSM Befehlszeilenschnittstelle (CLI)

CloudHSM CLI hilft Administratoren bei der Verwaltung von Benutzern und Krypto-Benutzern bei der Verwaltung von Schlüsseln in ihrem Cluster in. AWS CloudHSM Die CLI enthält Tools, mit denen Benutzer erstellt, gelöscht und aufgelistet, Benutzerkennwörter geändert und die Benutzer-Multi-Faktor-Authentifizierung (MFA) aktualisiert werden können. Sie umfasst auch Befehle zum

Generieren, Löschen, Importieren und Exportieren von Schlüsseln, zum Abrufen und Festlegen von Attributen, zum Suchen von Schlüsseln und zum Ausführen kryptografischer Operationen.

Eine definierte Liste von CloudHSM-CLI-Benutzern finden Sie unter [HSM-Benutzerverwaltung mit CloudHSM CLI](#). Eine definierte Liste von Schlüsselattributen für die CloudHSM-CLI finden Sie unter. [Schlüsselattribute für CloudHSM-CLI](#) Informationen zur Verwendung der CloudHSM-CLI zur Verwaltung von Schlüsseln finden Sie unter. [Schlüsselverwaltung mit CloudHSM CLI](#)

Informationen zum schnellen Einstieg finden Sie unter [Erste Schritte mit der AWS CloudHSM Befehlszeilenschnittstelle \(CLI\)](#). Detaillierte Informationen über die CloudHSM-CLI-Befehle und Beispiele zu ihrer Verwendung finden Sie unter [Referenz für CloudHSM-CLI-Befehle](#).

Themen

- [AWS CloudHSM Von der Befehlszeilenschnittstelle \(CLI\) unterstützte Plattformen](#)
- [Erste Schritte mit der AWS CloudHSM Befehlszeilenschnittstelle \(CLI\)](#)
- [Befehlsmodi in der CloudHSM-CLI](#)
- [Schlüsselattribute für CloudHSM-CLI](#)
- [Erweiterte Konfigurationen für CloudHSM CLI](#)
- [Referenz für CloudHSM-CLI-Befehle](#)

AWS CloudHSM Von der Befehlszeilenschnittstelle (CLI) unterstützte Plattformen

In diesem Thema werden die Linux- und Windows-Plattformen beschrieben, die von der AWS CloudHSM CLI unterstützt werden.

Linux-Support

Unterstützte Plattformen	X86_64-Architektur	ARM-Architektur
Amazon Linux 2	Ja	Ja
Amazon Linux 2023	Ja	Ja
Red Hat Enterprise Linux 8 (8.3+)	Ja	Ja

Unterstützte Plattformen	X86_64-Architektur	ARM-Architektur
RedHat Enterprise Linux 9 (9,2 und höher)	Ja	Ja
RedHat Enterprise Linux 10 (10.0+)	Ja	Ja
Ubuntu 22.04 LTS	Ja	Ja
Ubuntu 24.04 LTS	Ja	Ja

- SDK 5.16 war die letzte Version, die Unterstützung für die Ubuntu 20.04 LTS-Plattform bot. [Weitere Informationen finden Sie auf der Ubuntu-Website.](#)
- SDK 5.12 war die letzte Version, die CentOS 7 (7.8+) Plattformunterstützung bot. Weitere Informationen finden Sie auf der [CentOS-Website](#).
- SDK 5.12 war die letzte Version, die Unterstützung für die Plattform Red Hat Enterprise Linux 7 (7.8+) bot. Weitere Informationen finden Sie auf der [Red Hat Website](#).
- SDK 5.4.2 war die letzte Version, die CentOS 8-Plattformunterstützung bot. Weitere Informationen finden Sie auf der [CentOS-Website](#).

Windows-Unterstützung

- Microsoft Windows Server 2016
- Microsoft Windows Server 2019
- Microsoft Windows Server 2022
- Microsoft Windows Server 2025

Erste Schritte mit der AWS CloudHSM Befehlszeilenschnittstelle (CLI)

Mit der CloudHSM CLI Command Line Interface (CLI) können Sie Benutzer in Ihrem AWS CloudHSM Cluster verwalten. Verwenden Sie dieses Thema, um mit grundlegenden Benutzerverwaltungsaufgaben des Hardware Security Module (HSM) zu beginnen, z. B. Benutzer erstellen, Benutzer auflisten und CloudHSM CLI mit dem Cluster verbinden.

Themen

- [Installieren Sie die CloudHSM-CLI](#)
- [Verwenden Sie die CloudHSM-CLI](#)

Installieren Sie die CloudHSM-CLI

Verwenden Sie die folgenden Befehle, um die CloudHSM-CLI für herunterzuladen und zu installieren.
AWS CloudHSM

Amazon Linux 2023

Amazon Linux 2023 auf x86_64-Architektur:

```
$ 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
```

Amazon Linux 2023 zur ARM64 Architektur:

```
$ 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

Amazon Linux 2 auf x86_64-Architektur:

```
$ 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
```

Amazon Linux 2 zur ARM64 Architektur:

```
$ 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+)

RHEL 10 auf der x86_64-Architektur:

```
$ 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
```

RHEL 10 zur Architektur: ARM64

```
$ 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+)

RHEL 9 auf der x86_64-Architektur:

```
$ 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
```

RHEL 9 zur Architektur: ARM64

```
$ 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+)

RHEL 8 auf x86_64-Architektur:

```
$ 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
```

RHEL 8 zur ARM64 Architektur:

```
$ 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 24.04 LTS

Ubuntu 24.04 LTS auf der x86_64-Architektur:

```
$ 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
```

Ubuntu 24.04 LTS zur Architektur: ARM64

```
$ 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

Ubuntu 22.04 LTS auf der x86_64-Architektur:

```
$ 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
```

Ubuntu 22.04 LTS zur Architektur: ARM64

```
$ 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

Öffnen Sie Windows Server 2022 auf einer x86_64-Architektur PowerShell als Administrator und führen Sie den folgenden Befehl aus:

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

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

Windows Server 2019

Öffnen Sie Windows Server 2019 auf einer x86_64-Architektur PowerShell als Administrator und führen Sie den folgenden Befehl aus:

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

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

Windows Server 2016

Öffnen Sie Windows Server 2016 auf einer x86_64-Architektur PowerShell als Administrator und führen Sie den folgenden Befehl aus:

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

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

Verwenden Sie die folgenden Befehle, um CloudHSM-CLI zu konfigurieren.

So bootstrappen Sie eine Linux-EC2-Instance für Client-SDK 5

- Verwenden Sie das Configure-Tool, um die IP-Adresse der HSM(s) in Ihrem Cluster anzugeben.

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

So bootstrappen Sie eine Windows-EC2-Instance für Client-SDK 5

- Verwenden Sie das Configure-Tool, um die IP-Adresse der HSM(s) in Ihrem Cluster anzugeben.

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

Verwenden Sie die CloudHSM-CLI

Verwenden Sie die folgenden Befehle, um die CloudHSM-CLI zu starten und zu verwenden.

1. Verwenden Sie den folgenden Befehl, um die CloudHSM-CLI zu starten.

Linux

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

Windows

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

2. Verwenden Sie den login-Befehl, um sich beim Cluster anzumelden. Alle Benutzer können diesen Befehl verwenden.

Mit dem Befehl im folgenden Beispiel wird admin angemeldet, das Standard-[Admin](#)-Konto. Sie legen das Passwort für diesen Benutzer nach der [Aktivierung des Clusters](#) fest.

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

Sie werden vom System aufgefordert, Ihr Passwort einzugeben. Sie geben das Passwort ein und die Ausgabe zeigt, dass der Befehl erfolgreich war.

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

3. Führen Sie den `user list`-Befehl aus, um alle Benutzer im Cluster aufzulisten.

```
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. Verwenden Sie `user create`, um einen CU-Benutzer mit dem Namen **example_user** zu erstellen.

Sie können erstellen CUs , weil Sie sich in einem vorherigen Schritt als Admin-Benutzer angemeldet haben. Nur Admin-Benutzer können Benutzerverwaltungsaufgaben wie das Erstellen und Löschen von Benutzern und das Ändern der Passwörter anderer Benutzer ausführen.

```
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. Verwenden Sie `user list`, um alle Benutzer im Cluster aufzulisten.

```
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. Verwenden Sie den logout Befehl, um sich vom AWS CloudHSM Cluster abzumelden.

```
aws-cloudhsm > logout
{
  "error_code": 0,
  "data": "Logout successful"
}
```

7. Verwenden Sie den quit-Befehl, um die CLI zu stoppen.

```
aws-cloudhsm > quit
```

Befehlsmodi in der CloudHSM-CLI

In der CloudHSM-CLI können Sie Befehle auf zwei verschiedene Arten ausführen: im Einzelbefehlsmodus und im interaktiven Modus. Der interaktive Modus ist für Benutzer konzipiert, und der Einzelbefehlsmodus ist für Skripts konzipiert.

Note

Alle Befehle funktionieren im interaktiven Modus und im Einzelbefehlsmodus.

interaktiver Modus

Verwenden Sie die folgenden Befehle, um den interaktiven CloudHSM-CLI-Modus zu starten

Linux

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

Windows

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

Wenn Sie die CLI im interaktiven Modus verwenden, können Sie sich mit dem login-Befehl bei einem Benutzerkonto anmelden.

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

Um alle CloudHSM-CLI-Befehle auszuführen, führen Sie den folgenden Befehl aus:

```
aws-cloudhsm > help
```

Um die Syntax für einen CloudHSM-CLI-Befehl anzufordern, führen Sie den folgenden Befehl aus:

```
aws-cloudhsm > help <command-name>
```

Um eine Liste der Benutzer auf dem zu erhalten HSMs, geben Sie einuser list.

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

Um Ihre CloudHSM-CLI-Sitzung zu beenden, führen Sie den folgenden Befehl aus:

```
aws-cloudhsm > quit
```

Einzelbefehlsmodus

Note

Wenn Sie den Einzelbefehlsmodus verwenden, müssen Sie alle Sonderzeichen in Umgebungsvariablen und Befehlszeilenargumenten maskieren, die von Ihrer Shell interpretiert werden könnten.

Wenn Sie die CloudHSM-CLI im Einzelbefehlsmodus ausführen, müssen Sie zwei Umgebungsvariablen setzen, um die Anmeldedaten bereitzustellen: CLOUDHSM_PIN und CLOUDHSM_ROLE:

```
$ export CLOUDHSM_ROLE=admin
```

```
$ export CLOUDHSM_PIN=admin_username:admin_password
```


Danach können Sie Befehle mit den in Ihrer Umgebung gespeicherten Anmeldeinformationen ausführen.

```
$ cloudhsm-cli user change-password --username alice --role crypto-user
Enter password:
Confirm password:
{
  "error_code": 0,
  "data": {
    "username": "alice",
    "role": "crypto-user"
  }
}
```

Schlüsselattribute für CloudHSM-CLI

In diesem Thema wird beschrieben, wie Sie die CloudHSM-CLI zum Festlegen von Schlüsselattributen verwenden. Ein Schlüsselattribut in der CloudHSM-CLI kann den Typ eines Schlüssels definieren, wie ein Schlüssel funktionieren kann oder wie ein Schlüssel beschriftet wird. Einige Attribute definieren eindeutige Merkmale (z. B. den Typ eines Schlüssels). Andere Attribute können auf true oder false gesetzt werden. Wenn Sie sie ändern, wird entweder ein Teil der Funktionalität des Schlüssels aktiviert oder deaktiviert.

Beispiele zur Verwendung von Schlüsselattributen finden Sie in den Befehlen, die unter dem übergeordneten Befehl [Die Schlüsselkategorie in CloudHSM CLI](#) aufgeführt sind.

Die folgenden Themen enthalten zusätzliche Informationen zu Schlüsselattributen in der CloudHSM-CLI.

Themen

- [Unterstützte Attribute für CloudHSM CLI](#)
- [Wert in CloudHSM CLI überprüfen](#)
- [Verwandte Themen für CloudHSM CLI](#)

Unterstützte Attribute für CloudHSM CLI

Als bewährte Methode legen Sie nur Werte für Attribute fest, die Sie einschränken möchten. Wenn Sie keinen Wert angeben, verwendet CloudHSM-CLI den in der folgenden Tabelle angegebenen Standardwert.

In der folgenden Tabelle sind die wichtigsten Attribute, möglichen Werte, Standardwerte und zugehörige Hinweise für die CloudHSM-CLI aufgeführt. Eine leere Zelle in der Spalte Wert gibt an, dass dem Attribut kein spezifischer Standardwert zugewiesen ist.

CloudHSM-CLI-Attribut	Wert	Modifizierbar mit key set-attribute	Kann bei der Schlüsselerstellung eingestellt werden
<code>always-sensitive</code>	Der Wert ist True, wenn <code>sensitive</code> immer auf True gesetzt wurde und sich nie geändert hat.	Nein	Nein
<code>check-value</code>	Der Prüfwert des Schlüssels. Weitere Informationen finden Sie unter Weitere Details .	Nein	Nein
<code>class</code>	Die möglichen Werte lauten: <code>secret-key</code> , <code>public-key</code> und <code>private-key</code> .	Nein	Ja
<code>curve</code>	Elliptische Kurve, die zur Generierung des EC-Schlüsselpaars verwendet wird. Gültige Werte: <code>secp224r1</code>, <code>secp256r1</code>, <code>prime256v1</code>, <code>secp384r1</code>, und <code>secp256k1</code>, <code>secp521r1</code> und <code>ed25519</code>	Nein	Einstellbar mit EC, nicht einstellbar mit RSA

CloudHSM-CLI-Attribut	Wert	Modifizierbar mit key set-attribute	Kann bei der Schlüsselerstellung eingestellt werden
	ed25519 wird nur auf hsm2m.medium-Instances im Nicht-FIPS-Modus unterstützt.		
decrypt	Standard: False	Ja	Ja
derive	Standard: False	Derive kann für hsm2m.medium-Instances festgelegt werden. Es kann nicht für RSA-Schlüssel auf hsm1.medium-Instances festgelegt werden.	Ja
destroyable	Standard: True	Ja	Ja
ec-point	Bei EC-Schlüsseln die DER-Kodierung des ANSI ECPoint X9.62-Werts „Q“ in einem Hexadezimalformat. Für andere Schlüsseltypen ist dieses Attribut nicht vorhanden.	Nein	Nein
encrypt	Standard: False	Ja	Ja
extractable	Standard: True	Nein	Ja

CloudHSM-CLI-Attribut	Wert	Modifizierbar mit key set-attribute	Kann bei der Schlüsselerstellung eingestellt werden
<code>id</code>	Standard: leer	Die ID kann auf hsm2m.medium-Instanzen festgelegt werden. Sie kann nicht auf hsm1.medium-Instances festgelegt werden.	Ja
<code>key-length-bytes</code>	Erforderlich für die Generierung eines AES-Schlüssels. Gültige Werte: 16, 24 und 32 Byte.	Nein	Nein
<code>key-type</code>	Die möglichen Werte lauten: aes, rsa und ec.	Nein	Ja
<code>label</code>	Standard: leer	Ja	Ja
<code>local</code>	Standard: <code>True</code> für Schlüssel, die im HSM generiert wurden, <code>False</code> für Schlüssel, die in das HSM importiert wurden.	Nein	Nein
<code>modifiable</code>	Standard: <code>True</code>	Kann von „wahr“ auf „falsch“ geändert werden, aber nicht von „falsch“ auf „wahr“.	Ja

CloudHSM- CLI-Attribut	Wert	Modifizierbar mit key set-attribute	Kann bei der Schlüsselerstellung eingestellt werden
<code>modulus</code>	Der Modulus, der zum Generieren eines RSA-Schlüsselpaars verwendet wurde. Für andere Schlüsseltypen ist dieses Attribut nicht vorhanden.	Nein	Nein
<code>modulus-size-bits</code>	Erforderlich für die Generierung eines RSA-Schlüsselpaars. Der Mindestwert ist 2048.	Nein	Einstellbar mit RSA, nicht einstellbar mit EC
<code>never-extractable</code>	Der Wert ist <code>True</code> , wenn extrahierbar noch nie auf <code>False</code> gesetzt wurde. Der Wert ist <code>False</code> , wenn extrahierbar jemals auf <code>True</code> gesetzt wurde.	Nein	Nein
<code>private</code>	Standard: <code>True</code>	Nein	Ja

CloudHSM-CLI-Attribut	Wert	Modifizierbar mit key set-attribute	Kann bei der Schlüsselerstellung eingestellt werden
<code>public-exponent</code>	Erforderlich für die Generierung eines RSA-Schlüsselpaars. Gültige Werte: Bei diesem Wert muss es sich eine ungerade Zahl gleich oder größer als 65537 handeln.	Nein	Einstellbar mit RSA, nicht einstellbar mit EC
<code>sensitive</code>	Standard: • Der Wert ist <code>True</code> für AES-Schlüssel und private EC- und RSA-Schlüssel. • Der Wert ist <code>False</code> für öffentliche EC- und RSA-Schlüssel.	Nein	Einstellbar mit privaten Schlüsseln, nicht einstellbar mit öffentlichen Schlüsseln.
<code>sign</code>	Standard: • Der Wert ist <code>True</code> für AES-Schlüssel. • Der Wert ist <code>False</code> für RSA- und EC-Schlüssel.	Ja	Ja
<code>token</code>	Standard: <code>True</code>	Kann von falsch zu wahr geändert werden, aber nicht von wahr zu falsch.	Ja

CloudHSM-CLI-Attribut	Wert	Modifizierbar mit key set-attribute	Kann bei der Schlüsselerstellung eingestellt werden
trusted	Standard: False	Nur Admin-Benutzer können diesen Parameter festlegen.	Nein
unwrap	Standard: False	Ja	Ja, außer für öffentliche Schlüssel.
unwrap-template	Für die Werte sollte die Attributvorlage verwendet werden, die auf jeden Schlüssel angewendet wird, der mit diesem Schlüssel zum Packen entpackt wird.	Ja	Nein
verify	Standard: • Der Wert ist True für AES-Schlüssel. • Der Wert ist False für RSA- und EC-Schlüssel.	Ja	Ja
wrap	Standard: False	Ja	Ja, außer für private Schlüssel.

CloudHSM-CLI-Attribut	Wert	Modifizierbar mit key set-attribute	Kann bei der Schlüsselerstellung eingestellt werden
wrap-template	Für die Werte sollte die Attributvorlage verwendet werden, die dem Schlüssel entspricht, der mit diesem Schlüssel zum Packen gepackt wurde.	Ja	Nein
wrap-with-trusted	Standard: False	Ja	Ja

Wert in CloudHSM CLI überprüfen

Der Prüfwert in der CloudHSM-CLI ist ein 3-Byte-Hash oder eine Prüfsumme eines Schlüssels, der generiert wird, wenn das HSM einen Schlüssel importiert oder generiert. Sie können einen Prüfwert auch außerhalb des HSM berechnen, z. B. nachdem Sie einen Schlüssel exportiert haben. Anschließend können Sie die Prüfwerte vergleichen, um die Identität und Integrität des Schlüssels zu bestätigen. Um den Prüfwert eines Schlüssels zu ermitteln, verwenden Sie die [Schlüsselliste](#) mit dem verbose-Flag.

AWS CloudHSM verwendet die folgenden Standardmethoden, um einen Prüfwert zu generieren:

- Symmetrische Schlüssel: Die ersten 3 Byte des Ergebnisses der Verschlüsselung eines Nullblocks mit dem Schlüssel.
- Asymmetrische Schlüsselpaare: Die ersten 3 Byte des SHA-1-Hashs des öffentlichen Schlüssels.
- HMAC-Schlüssel: KCV für HMAC-Schlüssel wird derzeit nicht unterstützt.

Verwandte Themen für CloudHSM CLI

In den folgenden Themen finden Sie weitere Informationen zur CloudHSM-CLI.

- [Die Schlüsselkategorie in CloudHSM CLI](#)

- [Referenz für CloudHSM-CLI-Befehle](#)

Erweiterte Konfigurationen für CloudHSM CLI

Die AWS CloudHSM Befehlszeilenschnittstelle (CLI) umfasst die folgende erweiterte Konfiguration, die nicht Teil der allgemeinen Konfigurationen ist, die die meisten Kunden verwenden. Diese Konfigurationen bieten zusätzliche Funktionen.

- [Verbindung zu mehreren Clustern herstellen](#)

Verbindung zu mehreren Clustern mit CloudHSM CLI herstellen

Mit AWS CloudHSM Client SDK 5 können Sie die CloudHSM-CLI so konfigurieren, dass Verbindungen zu mehreren CloudHSM-Clustern von einer einzigen CLI-Instanz aus möglich sind.

In den folgenden Themen wird beschrieben, wie Sie die CloudHSM CLI Multi-Cluster-Funktionalität verwenden, um eine Verbindung mit mehreren Clustern herzustellen.

Themen

- [Voraussetzungen für mehrere Cluster AWS CloudHSM](#)
- [Konfigurieren Sie die CloudHSM-CLI für Multi-Cluster-Funktionalität](#)
- [Fügen Sie Ihrer Konfiguration einen Cluster hinzu AWS CloudHSM](#)
- [Entfernen Sie einen Cluster aus Ihrer Konfiguration AWS CloudHSM](#)
- [Interagieren Sie mit mehreren Clustern in AWS CloudHSM](#)

Voraussetzungen für mehrere Cluster AWS CloudHSM

Bevor Sie Ihren Cluster so konfigurieren, dass er eine Verbindung AWS CloudHSM zu mehreren Clustern herstellt, müssen Sie die folgenden Voraussetzungen erfüllen:

- Zwei oder mehr AWS CloudHSM Cluster, zu denen Sie eine Verbindung herstellen möchten, zusammen mit ihren Clusterzertifikaten.
- Eine EC2 Instance mit Sicherheitsgruppen, die korrekt konfiguriert sind, um eine Verbindung zu allen oben genannten Clustern herzustellen. Weitere Informationen zum Einrichten eines Clusters und der Client-Instanz finden Sie unter [Erste Schritte mit AWS CloudHSM](#).

- Um die Multi-Cluster-Funktionalität einzurichten, müssen Sie die CloudHSM-CLI bereits heruntergeladen und installiert haben. Wenn Sie dies noch nicht getan haben, lesen Sie die Anweisungen unter [???](#).
- Sie können nicht auf einen Cluster zugreifen, der mit konfiguriert ist, `./configure-cli[.exe]` -a da er nicht mit einem verknüpft ist. `cluster-id` Sie können ihn neu konfigurieren, indem Sie `config-cli add-cluster` wie in diesem Handbuch beschrieben vorgehen.

Konfigurieren Sie die CloudHSM-CLI für Multi-Cluster-Funktionalität

Gehen Sie folgendermaßen vor, um Ihre CloudHSM-CLI für Multi-Cluster-Funktionalität zu konfigurieren:

1. Identifizieren Sie die Cluster, mit denen Sie eine Verbindung herstellen möchten.
2. Fügen Sie diese Cluster mithilfe des `add-cluster` Unterbefehls [configure-cli](#) wie unten beschrieben zu Ihrer CloudHSM CLI-Konfiguration hinzu.
3. Starten Sie alle CloudHSM CLI-Prozesse neu, damit die neue Konfiguration wirksam wird.

Fügen Sie Ihrer Konfiguration einen Cluster hinzu AWS CloudHSM

Wenn Sie eine Verbindung zu mehreren Clustern herstellen, verwenden Sie den `configure-cli add-cluster` Befehl, um Ihrer Konfiguration einen Cluster hinzuzufügen.

Syntax

```
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]
```

Beispiele

Fügen Sie mithilfe des **cluster-id**-Parameters einen Cluster hinzu

Example

Verwenden Sie den `configure-cli add-cluster` zusammen mit dem `cluster-id`-Parameter, um Ihrer Konfiguration einen Cluster (mit der ID von `cluster-1234567`) hinzuzufügen.

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

Wenn die Verwendung von `configure-cli add-cluster` mit dem `cluster-id`-Parameter nicht dazu führt, dass der Cluster hinzugefügt wird, finden Sie im folgenden Beispiel eine längere Version dieses Befehls, die auch die Parameter `--region` und `--endpoint` zur Identifizierung des hinzugefügten Clusters erfordert. Wenn zum Beispiel die Region des Clusters eine andere ist als die, die als Standard für Ihre AWS CLI konfiguriert ist, sollten Sie den `--region`-Parameter verwenden, um die richtige Region zu verwenden. Darüber hinaus haben Sie die Möglichkeit, den AWS CloudHSM API-Endpoint anzugeben, der für den Anruf verwendet werden soll. Dies kann für verschiedene Netzwerkkonfigurationen erforderlich sein, z. B. für die Verwendung von VPC-Schnittstellenendpunkten, für die nicht den Standard-DNS-Hostnamen verwendet wird. AWS CloudHSM

Fügen Sie einen Cluster mit den Parametern **cluster-id**, **endpoint**, und **region** hinzu

Example

Verwenden Sie `configure-cli add-cluster` zusammen mit den Parametern `cluster-id`, `endpoint` und `region`, um Ihrer Konfiguration einen Cluster (mit der ID von `cluster-1234567`) hinzuzufügen.

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>
```

Weitere Hinweise zu den Parametern `--cluster-id`, `--region` und `--endpoint` finden Sie unter [the section called "Parameters"](#).

Parameters

`--cluster-id` <**Cluster ID**>

Führt einen `DescribeClusters`-Aufruf aus, um alle IP-Adressen der HSM-Elastic-Network-Schnittstelle (ENI) im Cluster mit der Cluster-ID zu finden. Das System fügt die ENI-IP-Adressen zu den AWS CloudHSM Konfigurationsdateien hinzu.

Note

Wenn Sie den `--cluster-id` Parameter von einer EC2 Instance innerhalb einer VPC verwenden, die keinen Zugriff auf das öffentliche Internet hat, müssen Sie einen VPC-Schnittstellen-Endpunkt erstellen, mit dem Sie eine Verbindung herstellen können. AWS CloudHSM Weitere Informationen über VPC-Endpunkte finden Sie unter [???](#).

Erforderlich: Ja

`--endpoint` **<Endpoint>**

Geben Sie den AWS CloudHSM API-Endpunkt an, der für den `DescribeClusters` Aufruf verwendet wird. Sie müssen diese Option in Kombination mit `--cluster-id` festlegen.

Erforderlich: Nein

`--hsm-ca-cert` **<HsmCA Certificate Filepath>**

Gibt den Dateipfad zum HSM-CA-Zertifikat an.

Erforderlich: Nein

`--region` **<Region>**

Geben Sie die Region Ihres Clusters an. Sie müssen diese Option in Kombination mit `--cluster-id` festlegen.

Wenn Sie den `--region`-Parameter nicht angeben, wählt das System die Region aus, indem es versucht, die Umgebungsvariablen `AWS_DEFAULT_REGION` oder `AWS_REGION` zu lesen. Wenn diese Variablen nicht festgelegt sind, überprüft das System die Region, die Ihrem Profil in Ihrer AWS-Config-Datei zugeordnet ist (normalerweise `~/.aws/config`), sofern Sie in der `AWS_CONFIG_FILE`-Umgebungsvariable keine andere Datei angegeben haben. Wenn keine der oben genannten Optionen festgelegt ist, verwendet das System standardmäßig die `us-east-1`-Region.

Erforderlich: Nein

`--client-cert-hsm-tls` **<client certificate hsm tls path>**

Pfad zum Client-Zertifikat, das für die gegenseitige Authentifizierung zwischen TLS Client und HSM verwendet wird.

Verwenden Sie diese Option nur, wenn Sie mindestens einen Vertrauensanker auf HSM mit CloudHSM CLI registriert haben. Sie müssen diese Option in Kombination mit `--client-key-hsm-tls-file` festlegen.

Erforderlich: Nein

`--client-key-hsm-tls` **<client key hsm tls path>**

Pfad zum Client-Schlüssel, der für die gegenseitige Authentifizierung zwischen TLS Client und HSM verwendet wird.

Verwenden Sie diese Option nur, wenn Sie mindestens einen Vertrauensanker auf HSM mit CloudHSM CLI registriert haben. Sie müssen diese Option in Kombination mit `--client-cert-hsm-tls-file` festlegen.

Erforderlich: Nein

Entfernen Sie einen Cluster aus Ihrer Konfiguration AWS CloudHSM

Wenn Sie mit der CloudHSM-CLI eine Verbindung zu mehreren Clustern herstellen, verwenden Sie den `configure-cli remove-cluster` Befehl, um einen Cluster aus Ihrer Konfiguration zu entfernen.

Syntax

```
configure-cli remove-cluster [OPTIONS]
    --cluster-id <CLUSTER ID>
    [-h, --help]
```

Beispiele

Entfernen Sie mithilfe des **cluster-id**-Parameters einen Cluster

Example

Verwenden Sie den `configure-cli remove-cluster` zusammen mit dem `cluster-id`-Parameter, um aus Ihrer Konfiguration einen Cluster (mit der ID von `cluster-1234567`) zu entfernen.

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>
```

Weitere Informationen zum Parameter `--cluster-id` erhalten Sie unter [the section called "Parameters"](#).

Parameter

`--cluster-id` **<Cluster ID>**

Die ID des Clusters, der aus der Konfiguration entfernt werden soll.

Erforderlich: Ja

Interagieren Sie mit mehreren Clustern in AWS CloudHSM

Nachdem Sie mehrere Cluster mit der CloudHSM-CLI konfiguriert haben, verwenden Sie den `cloudhsm-cli` Befehl, um mit ihnen zu interagieren.

Beispiele

cluster-id Bei Verwendung des interaktiven Modus einen Standard festlegen

Example

Verwenden Sie den [???](#) zusammen mit dem `cluster-id` Parameter, um einen Standardcluster (mit der ID von `cluster-1234567`) aus Ihrer Konfiguration festzulegen.

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>
```

Einstellung von **cluster-id**, wenn ein einzelner Befehl ausgeführt wird

Example

Verwenden Sie den `cluster-id` Parameter, um den Cluster (mit der ID von `cluster-1234567`) festzulegen, von [???](#) dem aus Sie abrufen möchten.

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>
```

Referenz für CloudHSM-CLI-Befehle

CloudHSM CLI unterstützt Administratoren bei der Verwaltung von Benutzern in ihrem Cluster. AWS CloudHSM CloudHSM-CLI kann in zwei Modi ausgeführt werden: im interaktiven Modus und im Einzelbefehlsmodus. Informationen zum schnellen Einstieg finden Sie unter [Erste Schritte mit der AWS CloudHSM Befehlszeilenschnittstelle \(CLI\)](#).

Um die meisten CloudHSM-CLI-Befehle auszuführen, müssen Sie die CloudHSM-CLI starten und sich beim HSM anmelden. Wenn Sie etwas hinzufügen oder löschen HSMs, aktualisieren Sie die Konfigurationsdateien für CloudHSM CLI. Andernfalls sind die Änderungen, die Sie vornehmen, möglicherweise nicht für alle HSMs Mitglieder des Clusters wirksam.

Die folgenden Themen beschreiben Befehle in CloudHSM-CLI.

Befehl	Description	Benutzertyp
activate	Aktiviert ein CloudHSM-Cluster und bestätigt, dass das Cluster neu ist. Dies muss geschehen, bevor andere Operationen ausgeführt werden können.	Nicht aktivierter Admin
hsm-info	Listen Sie die HSMs in Ihrem Cluster auf.	Alle 1 , auch nicht authentifizierte Benutzer. Eine Anmeldung ist nicht erforderlich.
ecdsa	Generiert eine Signatur mithilfe eines privaten EC-Schlüssels und des ECDSA-Signaturmechanismus.	Crypto-Benutzer (Crypto User, CU)

Befehl	Description	Benutzertyp
ed25519ph	Generiert eine Signatur unter Verwendung eines privaten Ed25519-Schlüssels und des HashEd DSA-Signaturmechanismus.	CU
rsa-pkcs	Generiert eine Signatur unter Verwendung eines privaten RSA-Schlüssels und des RSA-PKCS-Signaturmechanismus.	CU
rsa-pkcs-pss	Generiert eine Signatur unter Verwendung eines privaten RSA-Schlüssels und des Signaturmechanismus. RSA-PKCS-PSS	CU
ecdsa	Bestätigt, dass eine Datei im HSM mit einem bestimmten öffentlichen Schlüssel signiert wurde. Überprüft, ob die Signatur mithilfe des ECDSA-Signaturmechanismus generiert wurde. Vergleicht eine signierte Datei mit einer Quelldatei und ermittelt anhand eines bestimmten öffentlichen ECDSA-Schlüssels und Signaturmechanismus, ob die beiden kryptografisch verwandt sind.	CU
ed25519ph	Überprüft HashEd DSA-Signaturen mithilfe eines öffentlichen Ed25519-Schlüssels.	CU

Befehl	Description	Benutzertyp
rsa-pkcs	Bestätigt, dass eine Datei im HSM mit einem bestimmten öffentlichen Schlüssel signiert wurde. Überprüft, ob die Signatur mithilfe des RSA-PKCS-Signaturmechanismus generiert wurde. Vergleicht eine signierte Datei mit einer Quelldatei und bestimmt anhand eines bestimmten öffentlichen RSA-Schlüssels und Signaturmechanismus, ob die beiden kryptografisch verwandt sind.	CU
rsa-pkcs-pss	Bestätigt, dass eine Datei im HSM mit einem bestimmten öffentlichen Schlüssel signiert wurde. Überprüft, ob die Signatur mithilfe des RSA-PKCS-PSS Signaturmechanismus generiert wurde. Vergleicht eine signierte Datei mit einer Quelldatei und ermittelt anhand eines bestimmten öffentlichen RSA-Schlüssels und Signaturmechanismus, ob die beiden kryptografisch verwandt sind.	CU
key delete	Löscht einen Schlüssel aus Ihrem Cluster. AWS CloudHSM	CU

Befehl	Description	Benutzertyp
key generate-file	Generiert eine Schlüsseldatei in Ihrem AWS CloudHSM Cluster.	CU
Schlüssel generate-asymmetric-pair rsa	Generiert ein asymmetrisches RSA-Schlüsselpaar in Ihrem AWS CloudHSM Cluster.	CU
Schlüssel ec generate-asymmetric-pair	Generiert ein asymmetrisches key pair mit elliptischen Kurven (EC) in Ihrem Cluster. AWS CloudHSM	CU
key generate-symmetric aes	Generiert einen symmetrischen AES-Schlüssel in Ihrem Cluster. AWS CloudHSM	CU
key generate-symmetric generic-secret	Generiert einen symmetrischen Generic Secret-Schlüssel in Ihrem AWS CloudHSM Cluster.	CU
Schlüssel importieren (PEM)	Importiert einen Schlüssel im PEM-Format in ein HSM. Sie können ihn verwenden, um öffentliche und außerhalb des HSM erstellte Schlüssel zu importieren.	CU
key list	Findet alle Schlüssel für den aktuellen Benutzer in Ihrem AWS CloudHSM Cluster.	CU

Befehl	Description	Benutzertyp
Schlüssel replizieren	Replizieren Sie einen Schlüssel von einem Quellcluster auf einen geklonten Zielcluster.	CU
key set-attribute	Legt die Attribute der Schlüssel in Ihrem AWS CloudHSM Cluster fest.	CUs kann diesen Befehl ausführen, Admins können das vertrauenswürdige Attribut festlegen.
key share	Teilt sich einen Schlüssel mit anderen Mitgliedern CUs Ihres AWS CloudHSM Clusters.	CU
key unshare	Macht die gemeinsame Nutzung eines Schlüssels mit anderen CUs in Ihrem AWS CloudHSM Cluster rückgängig.	CU
aes-gcm	Entpackt einen Payload-Schlüssel mithilfe des AES-Wrapping-Schlüssels und des AES-GCM-Entpackungsmechanismus in den Cluster.	CU
aes-no-pad	Entpackt einen Payload-Schlüssel mithilfe des AES-Wrapping-Schlüssels und des Entpackungsmechanismus in den Cluster. AES-NO-PAD	CU

Befehl	Description	Benutzertyp
aes-pkcs5-pad	Entpackt einen Payload-Schlüssel mithilfe des AES-Wrapping-Schlüssels und des AES-PAD-Entpackungsmechanismus. PKCS5	CU
aes-zero-pad	Entpackt einen Payload-Schlüssel mithilfe des AES-Wrapping-Schlüssels und des Entpackungsmechanismus in den Cluster. AES-ZERO-PAD	CU
cloudhsm-aes-gcm	Entpackt mithilfe des AES-Wrapping-Schlüssels und des Entpackungsmechanismus einen Payload-Schlüssel in den Cluster. CLOUDHSM-AES-GCM	CU
rsa-aes	Entpackt einen Payload-Schlüssel mithilfe eines privaten RSA-Schlüssels und des RSA-AES-Entpackungsmechanismus.	CU
rsa-oaep	Entpackt einen Payload-Schlüssel mithilfe des privaten RSA-Schlüssels und des RSA-OAEP-Entpackungsmechanismus.	CU
rsa-pkcs	Entpackt einen Payload-Schlüssel mithilfe des privaten RSA-Schlüssels und des RSA-PKCS-Entpackungsmechanismus.	CU

Befehl	Description	Benutzertyp
aes-gcm	Wrappt einen Payload-Schlüssel mithilfe eines AES-Schlüssels auf dem HSM und des AES-GCM-Wrapping-Mechanismus ein.	CU
aes-no-pad	Umschließt einen Payload-Schlüssel mithilfe eines AES-Schlüssels auf dem HSM und dem Wrapping-Mechanismus. AES-NO-PAD	CU
aes-pkcs5-pad	Umschließt einen Payload-Schlüssel mithilfe eines AES-Schlüssels auf dem HSM und des AES-PAD-Wrapping-Mechanismus. PKCS5	CU
aes-zero-pad	Umschließt einen Payload-Schlüssel mithilfe eines AES-Schlüssels auf dem HSM und dem Wrapping-Mechanismus. AES-ZERO-PAD	CU
cloudhsm-aes-gcm	Umschließt einen Payload-Schlüssel mithilfe eines AES-Schlüssels auf dem HSM und dem Wrapping-Mechanismus. CLOUDHSM-AES-GCM	CUs
rsa-aes	Umschließt einen Payload-Schlüssel mithilfe eines öffentlichen RSA-Schlüssels auf dem HSM und des RSA-AES-Wrapping-Mechanismus.	CU

Befehl	Description	Benutzertyp
rsa-oeap	Umschließt einen Payload-Schlüssel mithilfe eines öffentlichen RSA-Schlüssels auf dem HSM und des RSA-OAEP-Wrapping-Mechanismus.	CU

Befehl	Description	Benutzertyp
Verwenden Sie den <code>key wrap rsa-pkcs</code> Befehl in der CloudHSM-CLI, um einen Payload-Schlüssel mithilfe eines öffentlichen RSA-Schlüssels auf dem Hardware-Sicherheitsmodul (HSM) und dem Wrapping-Mechanismus zu verpacken. RSA-PKCS Das Attribut des Payload-Schlüssels <code>extractable</code> muss auf <code>true</code> gesetzt sein. Nur der Besitzer eines Schlüssels, d. h. der Crypto-Benutzer (CU), der den Schlüssel erstellt hat, kann den Schlüssel umschließen. Benutzer, die den Schlüssel gemeinsam nutzen, können den Schlüssel für kryptografische Operationen verwenden. Um den <code>key wrap rsa-pkcs</code> Befehl verwenden zu können, benötigen Sie zunächst einen RSA-Schlüssel in Ihrem AWS CloudHSM Cluster. Sie können ein RSA-Schlüsselpaar mit dem <code>Die generate-asymmetric-pair</code> Kategorie in CloudHSM CLI Befehl und dem auf <code>true</code> gesetzten <code>wrap</code> Attribut generieren.	Umschließt einen Payload-Schlüssel mithilfe eines öffentlichen RSA-Schlüssels auf dem HSM und des RSA-PKCS-Wrapping-Mechanismus.	CU

Die folgenden Benutzertypen können diesen Befehl

Befehl	Description	Benutzertyp
login	Melden AWS CloudHSM Sie sich bei Ihrem Cluster an.	Administrator, Crypto-Benutzer (CU) und Appliance-Benutzer (AU)
logout	Melden Sie sich von Ihrem AWS CloudHSM Cluster ab.	Administrator, CU und Appliance-Benutzer (AU)
quorum token-sign delete	Löscht ein oder mehrere Token für einen vom Quorum autorisierten Dienst.	Admin.
quorum token-sign generate	Generiert ein Token für einen vom Quorum autorisierten Dienst.	Admin.
quorum token-sign list	Listet alle token-sign-Quorum-Token auf, die in Ihrem CloudHSM-Cluster vorhanden sind.	Alle 1 , auch nicht authentifizierte Benutzer. Eine Anmeldung ist nicht erforderlich.
Quorum-Tokenzeichen list-quorum-values	Listet die in Ihrem CloudHSM-Cluster festgelegten Quorumwerte auf.	Alle 1 , auch nicht authentifizierte Benutzer. Eine Anmeldung ist nicht erforderlich.
Quorum-Tokenzeichen set-quorum-value	Legt einen neuen Quorumwert für einen vom Quorum autorisierten Dienst fest.	Admin.
user change-mfa	Ändert die Strategie zur Multi-Faktor-Authentifizierung (MFA) eines Benutzers.	Admin, CU

Befehl	Description	Benutzertyp
user change-password	Ändert die Passwörter von Benutzern auf dem. HSMs Jeder Benutzer kann das eigene Passwort ändern. Admins können das Passwort jedes Benutzers ändern.	Admin, CU
user create	Erstellt einen Benutzer in Ihrem AWS CloudHSM Cluster.	Admin.
user delete	Löscht einen Benutzer in Ihrem AWS CloudHSM Cluster.	Admin.
user list	Listet die Benutzer in Ihrem AWS CloudHSM Cluster auf.	Alle ¹ , auch nicht authentifizierte Benutzer. Eine Anmeldung ist nicht erforderlich.
user change-quorum token-sign register	Registriert die Quorumstrategie mit Quorum-token-sign für einen Benutzer.	Admin.

Anmerkungen

- [1] Alle Benutzer umfasst alle aufgelisteten Rollen und Benutzer, die nicht angemeldet sind.

Die Cluster-Kategorie in CloudHSM CLI

In der CloudHSM-CLI cluster ist dies eine übergeordnete Kategorie für eine Gruppe von Befehlen, die in Kombination mit der übergeordneten Kategorie einen Befehl erstellen, der spezifisch für Cluster ist. Derzeit besteht die Cluster-Kategorie aus den folgenden Befehlen:

Themen

- [Aktivieren Sie einen Cluster mit CloudHSM CLI](#)
- [Liste HSMs mit CloudHSM CLI](#)
- [Die Cluster-MTLS-Kategorie in der CloudHSM-CLI](#)

Aktivieren Sie einen Cluster mit CloudHSM CLI

Verwenden Sie den `cluster activate` Befehl in der CloudHSM-CLI, um [einen neuen Cluster in zu aktivieren](#). AWS CloudHSM Dieser Befehl muss ausgeführt werden, bevor der Cluster zum Ausführen von kryptografischen Operationen verwendet werden kann.

Benutzertyp

Die folgenden Benutzertypen können diesen Befehl ausführen.

- Nicht aktivierter Admin

Syntax

Dieser Befehl hat keine Parameter.

```
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')

Beispiel

Dieser Befehl aktiviert Ihr Cluster, indem er das anfängliche Passwort für Ihren Admin-Benutzer festlegt.

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

Verwandte Themen

- [user create](#)
- [user delete](#)
- [user change-password](#)

Liste HSMs mit CloudHSM CLI

Verwenden Sie den `cluster hsm-info` Befehl in der CloudHSM-CLI, um die Hardware-Sicherheitsmodule (HSMs) in Ihrem AWS CloudHSM Cluster aufzulisten. Sie müssen nicht bei CloudHSM-CLI angemeldet sein, um diesen Befehl auszuführen.

Note

Wenn Sie hinzufügen oder löschen HSMs, aktualisieren Sie die Konfigurationsdateien, die der AWS CloudHSM Client und die Befehlszeilentools verwenden. Andernfalls sind die Änderungen, die Sie vornehmen, möglicherweise nicht für alle Mitglieder HSMs des Clusters wirksam.

Benutzertyp

Die folgenden Benutzertypen können diesen Befehl ausführen.

- Alle Benutzer. Sie müssen nicht angemeldet sein, um diesen Befehl auszuführen.

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

Beispiel

Dieser Befehl listet die HSMs vorhandenen Elemente in Ihrem AWS CloudHSM Cluster auf.

```
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]"
  }
]
}
}

```

Diese Ausgabe hat die folgenden Attribute:

- Anbieter: Der Name des Anbieters des HSM.
- Modell: Die Modellnummer des HSM.
- Seriennummer: Die Seriennummer des HSM. Dies kann sich aufgrund von Ersetzungen ändern.
- Hardware-version-major: Die wichtigste Hardwareversion.
- Hardware-version-minor: Die kleinere Hardwareversion.
- Firmware-version-major: Die Haupt-Firmware-Version.
- Firmware-version-minor: Die kleinere Firmware-Version.
- Firmware-build-number: Die Build-Nummer der Firmware.
- Firmware-id: Die Firmware-ID, die die Haupt- und Nebenversionen zusammen mit dem Build enthält.
- FIPS-Status: Der FIPS-Modus des Clusters und der darin befindliche HSMs. Im FIPS-Modus lautet die Ausgabe „2 [FIPS-Modus mit Einzelfaktor-Authentifizierung]“. Im Nicht-FIPS-Modus lautet die Ausgabe „0 [Nicht-FIPS-Modus mit Einzelfaktor-Authentifizierung]“.

Verwandte Themen

- [Aktivieren Sie einen Cluster mit CloudHSM CLI](#)

Die Cluster-MTLS-Kategorie in der CloudHSM-CLI

In CloudHSM CLI `cluster mtls` ist dies eine übergeordnete Kategorie für eine Gruppe von Befehlen, die in Kombination mit der übergeordneten Kategorie einen Befehl erstellen, der spezifisch für Cluster ist. AWS CloudHSM. Derzeit besteht diese Kategorie aus den folgenden Befehlen:

Themen

- [Einen Vertrauensanker mit der CloudHSM-CLI deregistrieren](#)
- [Holen Sie sich die mTLS-Durchsetzungsstufe mit der CloudHSM CLI](#)
- [Vertrauensanker mit CloudHSM CLI auflisten](#)
- [Registrieren Sie einen Vertrauensanker mit CloudHSM CLI](#)
- [Legen Sie die mTLS-Durchsetzungsstufe mit der CloudHSM CLI fest](#)

Einen Vertrauensanker mit der CloudHSM-CLI deregistrieren

Verwenden Sie den `cluster mtls deregister-trust-anchor` Befehl in der CloudHSM-CLI, um einen Vertrauensanker für gegenseitiges TLS zwischen Client und abzumelden. AWS CloudHSM.

Benutzertyp

Die folgenden Benutzer können diesen Befehl ausführen.

- Admin.

Voraussetzungen

- Um diesen Befehl auszuführen, müssen Sie als Admin-Benutzer angemeldet sein.

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

Beispiel

Example

Im folgenden Beispiel entfernt dieser Befehl einen Vertrauensanker aus dem HSM.

```
aws-cloudhsm > cluster mtls deregister-trust-anchor --certificate-reference 0x01

{
  "error_code": 0,
  "data": {
    "message": "Trust anchor with reference 0x01 deregistered successfully"
  }
}
```

Anschließend können Sie den list-trust-anchors Befehl ausführen, um zu bestätigen, dass der Vertrauensanker aus dem folgenden Verzeichnis abgemeldet wurde: AWS CloudHSM

```
aws-cloudhsm > cluster mtls list-trust-anchors

{
  "error_code": 0,
  "data": {
    "trust_anchors": []
  }
}
```


Argumente

<CLUSTER_ID>

Die ID des Clusters, auf dem dieser Vorgang ausgeführt werden soll.

Erforderlich: Wenn mehrere Cluster [konfiguriert wurden](#).

<CERTIFICATE_REFERENCE>

Eine hexadezimale oder dezimale Zertifikatsreferenz.

Erforderlich: Ja

Warning

Nachdem Sie einen Vertrauensanker im Cluster deregistriert haben, werden alle vorhandenen mTLS-Verbindungen, die das von diesem Vertrauensanker signierte Client-Zertifikat verwenden, gelöscht.

<APPROVAL>

Gibt den Dateipfad zu einer signierten Quorum-Token-Datei an, um den Vorgang zu genehmigen. Nur erforderlich, wenn der Quorumwert des Quorumclusterdienstes größer als 1 ist.

Verwandte Themen

- [Cluster-MTLS reregister-trust-anchor](#)
- [Cluster-MTLS list-trust-anchors](#)
- [mTLS einrichten \(empfohlen\)](#)

Holen Sie sich die mTLS-Durchsetzungsstufe mit der CloudHSM CLI

Verwenden Sie den `cluster mtls get-enforcement` Befehl in der CloudHSM-CLI, um die Durchsetzungsstufe für die Verwendung von gegenseitigem TLS zwischen Client und zu ermitteln.

AWS CloudHSM

Benutzertyp

Die folgenden Benutzer können diesen Befehl ausführen.

- Admin.
- Krypto-Benutzer () CUs

Voraussetzungen

- Um diesen Befehl ausführen zu können, müssen Sie als Admin-Benutzer oder Crypto-Benutzer (CUs) angemeldet sein.

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

Beispiel

Example

Im folgenden Beispiel listet dieser Befehl die Mtls-Erzwingungsstufe von auf AWS CloudHSM.

```
aws-cloudhsm > cluster mtls get-enforcement
```

```
{
  "error_code": 0,
  "data": {
    "mtls-enforcement-level": "none"
  }
}
```

Argumente

<CLUSTER_ID>

Die ID des Clusters, auf dem dieser Vorgang ausgeführt werden soll.

Erforderlich: Wenn mehrere Cluster [konfiguriert wurden](#).

Verwandte Themen

- [Durchsetzung von Cluster-MTLS-Sets](#)
- [mTLS einrichten \(empfohlen\)](#)

Vertrauensanker mit CloudHSM CLI auflisten

Verwenden Sie den `cluster mtls list-trust-anchors` Befehl in der CloudHSM-CLI, um alle Vertrauensanker aufzulisten, die für gegenseitiges TLS zwischen Client und verwendet werden können. AWS CloudHSM

Benutzertyp

Die folgenden Benutzer können diesen Befehl ausführen.

- Alle Benutzer. Sie müssen nicht angemeldet sein, um diesen Befehl auszuführen.

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
```

Beispiel

Example

Im folgenden Beispiel listet dieser Befehl alle registrierten Vertrauensanker aus dem auf. AWS CloudHSM

```
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"
      }
    ]
  }
}
```

Argumente

<CLUSTER_ID>

Die ID des Clusters, auf dem dieser Vorgang ausgeführt werden soll.

Erforderlich: Wenn mehrere Cluster [konfiguriert wurden](#).

Verwandte Themen

- [Cluster-MTLS reregister-trust-anchor](#)
- [Cluster-MTLS deregister-trust-anchor](#)
- [mTLS einrichten \(empfohlen\)](#)

Registrieren Sie einen Vertrauensanker mit CloudHSM CLI

Verwenden Sie den `cluster mtls register-trust-anchor` Befehl in der CloudHSM-CLI, um einen Vertrauensanker für gegenseitiges TLS zwischen Client und zu registrieren. AWS CloudHSM

Benutzertyp

Die folgenden Benutzer können diesen Befehl ausführen.

- Admin.

Voraussetzungen

Der AWS CloudHSM akzeptiert Vertrauensanker mit den folgenden Schlüsseltypen:

Schlüsseltyp	Description
EC	Die Kurven <code>secp256r1</code> (P-256), <code>secp384r1</code> (P-384) und <code>secp521r1</code> (P-521).
RSA	2048-Bit-, 3072-Bit- und 4096-Bit-RSA-Schlüssel.

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

Beispiel

Example

Im folgenden Beispiel registriert dieser Befehl einen Vertrauensanker auf dem HSM. Die maximale Anzahl von Vertrauensankern, die registriert werden können, beträgt zwei (2).

```
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"
    }
  }
}
```

Sie können dann den `list-trust-anchors` Befehl ausführen, um zu bestätigen, dass der Vertrauensanker registriert wurde auf AWS CloudHSM:

```
aws-cloudhsm > cluster mtls list-trust-anchors

{
  "error_code": 0,
  "data": {
    "trust_anchors": [
      {
        "certificate-reference": "0x01",
        "certificate": "<PEM Encoded Certificate>",
        "cluster-coverage": "full"
      }
    ]
  }
}
```

Argumente

<CLUSTER_ID>

Die ID des Clusters, auf dem dieser Vorgang ausgeführt werden soll.

Erforderlich: Wenn mehrere Cluster [konfiguriert wurden](#).

<PATH>

Dateipfad des zu registrierenden Vertrauensankers.

Erforderlich: Ja

Note

AWS CloudHSM unterstützt die Registrierung von Zwischenzertifikaten als Vertrauensanker. In solchen Fällen muss die gesamte PEM-kodierte Zertifikatskettendatei im HSM registriert werden, wobei die Zertifikate in hierarchischer Reihenfolge angeordnet sind.

AWS CloudHSM unterstützt eine Zertifikatskette von 6980 Byte.

<APPROVAL>

Gibt den Dateipfad zu einer signierten Quorum-Token-Datei an, um den Vorgang zu genehmigen. Nur erforderlich, wenn der Quorumwert des Quorumclusterdienstes größer als 1 ist.

Verwandte Themen

- [Cluster-MTLS deregister-trust-anchor](#)
- [Cluster-MTLS list-trust-anchors](#)
- [mTLS einrichten \(empfohlen\)](#)

Legen Sie die mTLS-Durchsetzungsstufe mit der CloudHSM CLI fest

Verwenden Sie den `cluster mtls set-enforcement` Befehl in der CloudHSM-CLI, um die Durchsetzungsstufe für die Verwendung von gegenseitigem TLS zwischen Client und festzulegen.

AWS CloudHSM

Benutzertyp

Die folgenden Benutzer können diesen Befehl ausführen.

- Admin mit dem Benutzernamen als Admin

Voraussetzungen

Um diesen Befehl auszuführen:

- Mindestens ein Vertrauensanker wurde erfolgreich auf dem registriert AWS CloudHSM.
- Konfigurieren Sie die CloudHSM-CLI mit dem richtigen privaten Schlüssel und dem richtigen Client-Zertifikat und starten Sie die CloudHSM-CLI unter einer gegenseitigen TLS-Verbindung.
- Sie müssen als Standard-Administrator mit dem Benutzernamen „admin“ angemeldet sein. Jeder andere Admin-Benutzer kann diesen Befehl nicht ausführen.

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

Beispiel

Example

Im folgenden Beispiel legt dieser Befehl die MTLS-Erzwingungsstufe des gewünschten AWS CloudHSM Clusters fest. Der Befehl set-enforcement kann nur in einer gegenseitigen TLS-

Verbindung ausgeführt werden, wenn Sie als Admin-Benutzer mit dem Benutzernamen admin angemeldet sind. Weitere Informationen finden Sie unter [mTLS-Erzwingung einrichten](#) für AWS CloudHSM

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

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

Anschließend können Sie den get-enforcement Befehl ausführen, um zu bestätigen, dass die Erzwingungsstufe auf Cluster gesetzt wurde:

```
aws-cloudhsm > cluster mtls get-enforcement

{
  "error_code": 0,
  "data": {
    "mtls-enforcement-level": "cluster"
  }
}
```

Argumente

<CLUSTER_ID>

Die ID des Clusters, auf dem dieser Vorgang ausgeführt werden soll.

Erforderlich: Wenn mehrere Cluster [konfiguriert wurden](#).

<LEVEL>

Stufe, die für MTLS im Cluster festgelegt werden soll.

Zulässige Werte

- Cluster: Erzwingt die Verwendung von gegenseitigem TLS zwischen Client und AWS CloudHSM Cluster.

- none: Erzwingen Sie nicht die Verwendung von gegenseitigem TLS zwischen Client und AWS CloudHSM Cluster.

Erforderlich: Ja

 Warning

Nachdem Sie die mTLS-Nutzung im Cluster erzwungen haben, werden alle bestehenden Nicht-mTLS-Verbindungen gelöscht und Sie können nur mit mTLS-Zertifikaten eine Verbindung zum Cluster herstellen.

<APPROVAL>

Gibt den Dateipfad zu einer signierten Quorum-Token-Datei an, um den Vorgang zu genehmigen. Nur erforderlich, wenn der Quorumwert des Quorumclusterdienstes größer als 1 ist.

Verwandte Themen

- [Erzwingung von Cluster-MTLS](#)
- [mTLS einrichten \(empfohlen\)](#)

Die Krypto-Kategorie in CloudHSM CLI

In der CloudHSM-CLI `crypto` ist dies eine übergeordnete Kategorie für eine Gruppe von Befehlen, die in Kombination mit der übergeordneten Kategorie einen Befehl erstellen, der für kryptografische Operationen spezifisch ist. Derzeit besteht diese Kategorie aus den folgenden Befehlen:

- [sign](#)
 - [ecdsa](#)
 - [ed25519ph](#)
 - [rsa-pkcs](#)
 - [rsa-pkcs-pss](#)
- [verify](#)
 - [ecdsa](#)
 - [ed25519ph](#)

- [rsa-pkcs](#)
- [rsa-pkcs-pss](#)

Die Kategorie Kryptozeichen in CloudHSM CLI

In der CloudHSM-CLI `crypto sign` ist dies eine übergeordnete Kategorie für eine Gruppe von Befehlen, die in Kombination mit der übergeordneten Kategorie einen ausgewählten privaten Schlüssel in Ihrem AWS CloudHSM Cluster verwendet, um eine Signatur zu generieren. `crypto sign` hat die folgenden Unterbefehle:

- [Generieren Sie eine Signatur mit dem ECDSA-Mechanismus in der CloudHSM-CLI](#)
- [Generieren Sie eine Signatur mit dem HashEd DSA-Mechanismus in der CloudHSM-CLI](#)
- [Generieren Sie eine Signatur mit dem RSA-PKCS-Mechanismus in der CloudHSM-CLI](#)
- [Generieren Sie eine Signatur mit dem RSA-PKCS-PSS Mechanismus in CloudHSM CLI](#)

Um sie verwenden zu können `crypto sign`, benötigen Sie einen privaten Schlüssel in Ihrem HSM. Sie können einen privaten Schlüssel mit den folgenden Befehlen generieren:

- [Schlüssel `generate-asymmetric-pair ec`](#)
- [Schlüssel `generate-asymmetric-pair RSA`](#)

Generieren Sie eine Signatur mit dem ECDSA-Mechanismus in der CloudHSM-CLI

Verwenden Sie den `crypto sign ecdsa` Befehl in der CloudHSM-CLI, um eine Signatur mithilfe eines privaten EC-Schlüssels und des ECDSA-Signaturmechanismus zu generieren.

Um den `crypto sign ecdsa` Befehl verwenden zu können, benötigen Sie zunächst einen privaten EC-Schlüssel in Ihrem Cluster. AWS CloudHSM Sie können mit dem [Generieren Sie ein asymmetrisches EC-Schlüsselpaar mit CloudHSM CLI](#) Befehl, dessen `sign` Attribut auf `true` gesetzt ist, einen privaten EC-Schlüssel generieren `true`.

Die resultierende ECDSA-Signatur wird in dem Format generiert `| | s`, in dem die Komponenten R und S als binäre Rohdaten verkettet und im Base64-codierten Format zurückgegeben werden.

Note

Signaturen können mit Unterbefehlen verifiziert werden. AWS CloudHSM [Die Kategorie Crypto Verify in CloudHSM CLI](#)

Benutzertyp

Die folgenden Benutzertypen können diesen Befehl ausführen.

- Krypto-Benutzer () CUs

Voraussetzungen

- Um diesen Befehl auszuführen, müssen Sie als CU angemeldet sein.

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
```

Beispiel

Diese Beispiele zeigen, wie eine Signatur mithilfe des ECDSA-Signaturmechanismus und der SHA256 Hash-Funktion generiert wird. `crypto sign ecdsa` Dieser Befehl verwendet einen privaten Schlüssel im HSM.

Example Beispiel: Generieren Sie eine Signatur für Base-64-kodierte Daten

```
aws-cloudhsm > crypto sign ecdsa --key-filter attr.label=ec-private --hash-function
sha256 --data YWJjMTIz
{
  "error_code": 0,
  "data": {
    "key-reference": "0x000000000007808dd",
    "signature": "4zki+FzjhP7Z/KqoQvh4ueMAxQQVp7FQguZ2w0S3Q5bzk
+Hc5irV5iTkuxQbroPttVFZ8V6FgR2fz+sPegwCw=="
  }
}
```

Example Beispiel: Generieren Sie eine Signatur für eine Datendatei

```
aws-cloudhsm > crypto sign ecdsa --key-filter attr.label=ec-private --hash-function
sha256 --data-path data.txt
{
  "error_code": 0,
  "data": {
    "key-reference": "0x000000000007808dd",
    "signature": "4zki+FzjhP7Z/KqoQvh4ueMAxQQVp7FQguZ2w0S3Q5bzk
+Hc5irV5iTkuxQbroPttVFZ8V6FgR2fz+sPegwCw=="
  }
}
```

Argumente

<CLUSTER_ID>

Die ID des Clusters, auf dem dieser Vorgang ausgeführt werden soll.

Erforderlich: Wenn mehrere Cluster [konfiguriert wurden](#).

<DATA>

Base64-kodierte Daten, die signiert werden sollen.

Erforderlich: Ja (sofern nicht über den Datenpfad angegeben)

<DATA_PATH>

Gibt den Speicherort der zu signierenden Daten an.

Erforderlich: Ja (sofern nicht über den Datenpfad angegeben)

<HASH_FUNCTION>

Gibt die Hash-Funktion an.

Zulässige Werte:

- sha1
- sha224
- sha256
- sha384
- sha512

Erforderlich: Ja

<KEY_FILTER>

Schlüsselreferenz (z. B. `key-reference=0xabc`) oder durch Leerzeichen getrennte Liste von Schlüsselattributen in der Form `attr.key_attribute_name=KEY_ATTRIBUTE_VALUE`, um einen passenden Schlüssel auszuwählen.

Eine Liste der unterstützten CloudHSM-CLI-Schlüsselattribute finden Sie unter Schlüsselattribute für CloudHSM CLI.

Erforderlich: Ja

<APPROVAL>

Gibt den Dateipfad zu einer signierten Quorum-Token-Datei an, um den Vorgang zu genehmigen. Nur erforderlich, wenn der Quorumwert des privaten Schlüssels für den Schlüsselnutzungsdienst größer als 1 ist.

<DATA_TYPE>

Gibt an, ob der Wert des Datenparameters als Teil des Signaturalgorithmus gehasht werden soll. Wird `raw` für Daten ohne Hashwert verwendet; wird `digest` für Digests verwendet, die bereits gehasht wurden.

Zulässige Werte:

- RAW
- Digest

Verwandte Themen

- [Die Kategorie Kryptozeichen in CloudHSM CLI](#)
- [Die Kategorie Crypto Verify in CloudHSM CLI](#)

Generieren Sie eine Signatur mit dem HashEd DSA-Mechanismus in der CloudHSM-CLI

Important

HashEdDSA-Signaturvorgänge werden nur auf `hsm2m.medium`-Instances im Nicht-FIPS-Modus unterstützt.

Verwenden Sie den `crypto sign ed25519ph` Befehl in der CloudHSM-CLI, um eine Signatur mithilfe eines privaten Ed25519-Schlüssels und des HashEd DSA-Signaturmechanismus zu generieren. Weitere Informationen zu HashEd DSA finden Sie unter [NIST SP 186-5](#), Abschnitt 7.8.

Um den `crypto sign ed25519ph` Befehl verwenden zu können, benötigen Sie zunächst einen privaten Ed25519-Schlüssel in Ihrem Cluster. AWS CloudHSM Sie können einen privaten Ed25519-Schlüssel generieren, indem Sie den [Generieren Sie ein asymmetrisches EC-Schlüsselpaar mit CloudHSM CLI](#) Befehl verwenden, wobei der `curve` Parameter auf `ed25519` und das `sign` Attribut auf `true` gesetzt ist.

Note

Signaturen können AWS CloudHSM mit [Die Kategorie Crypto Verify in CloudHSM CLI](#) Unterbefehlen verifiziert werden.

Benutzertyp

Die folgenden Benutzertypen können diesen Befehl ausführen.

- Krypto-Benutzer () CUs

Voraussetzungen

- Um diesen Befehl auszuführen, müssen Sie als CU angemeldet sein.
- HashEdDSA-Signaturvorgänge werden nur auf hsm2m.medium-Instances im Nicht-FIPS-Modus unterstützt.

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
```

Beispiel

Diese Beispiele zeigen, wie eine Signatur mithilfe des `crypto sign ed25519ph` Ed25519PH-Signaturmechanismus und der Hash-Funktion generiert wird. `sha512` Dieser Befehl verwendet einen privaten Schlüssel im HSM.

Example Beispiel: Generieren Sie eine Signatur für Base-64-kodierte Daten

```
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": "0x00000000000401cdf",
    "signature": "mKcCIvC4Ehqp0w+BPWg/gJ5GK0acf/h20UmbuU5trkEx
+FBCRjwqNVogA9BirfWqoQuMYeY2Biqq0RwqJgg0Bg=="
  }
}
```

Example Beispiel: Generieren Sie eine Signatur für eine Datendatei

```
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": "0x00000000000401cdf",
    "signature": "mKcCIvC4Ehqp0w+BPWg/gJ5GK0acf/h20UmbuU5trkEx
+FBCRjwqNVogA9BirfWqoQuMYeY2Biqq0RwqJgg0Bg=="
  }
}
```

Argumente

<CLUSTER_ID>

Die ID des Clusters, auf dem dieser Vorgang ausgeführt werden soll.

Erforderlich: Wenn mehrere Cluster [konfiguriert wurden](#).

<DATA>

Base64-kodierte Daten, die signiert werden sollen.

Erforderlich: Ja (sofern nicht über den Datenpfad angegeben)

<DATA_PATH>

Gibt den Speicherort der zu signierenden Daten an.

Erforderlich: Ja (sofern nicht über den Datenparameter angegeben)

<HASH_FUNCTION>

Gibt die Hash-Funktion an. ED25519Ph unterstützt nur. SHA512

Zulässige Werte:

- sha512

Erforderlich: Ja

<KEY_FILTER>

Schlüsselreferenz (z. B. `key-reference=0xabc`) oder durch Leerzeichen getrennte Liste von Schlüsselattributen in der Form `attr.key_attribute_name=KEY_ATTRIBUTE_VALUE`, um einen passenden Schlüssel auszuwählen.

Eine Liste der unterstützten CloudHSM-CLI-Schlüsselattribute finden Sie unter. [Schlüsselattribute für CloudHSM-CLI](#)

Erforderlich: Ja

<APPROVAL>

Gibt den Dateipfad zu einer signierten Quorum-Token-Datei an, um den Vorgang zu genehmigen. Nur erforderlich, wenn der Quorumwert des privaten Schlüssels für den Schlüsselverwendungsdienst größer als 1 ist.

<DATA_TYPE>

Gibt an, ob der Wert des Datenparameters als Teil des Signaturalgorithmus gehasht werden soll. Wird `raw` für Daten ohne Hashwert verwendet; wird `digest` für Digests verwendet, die bereits gehasht wurden.

Zulässige Werte:

- RAW
- Digest

Erforderlich: Ja

Verwandte Themen

- [Die Kategorie Kryptozeichen in CloudHSM CLI](#)
- [Die Kategorie Crypto Verify in CloudHSM CLI](#)

Generieren Sie eine Signatur mit dem RSA-PKCS-Mechanismus in der CloudHSM-CLI

Verwenden Sie den `crypto sign rsa-pkcs` Befehl in der CloudHSM-CLI, um eine Signatur mithilfe eines privaten RSA-Schlüssels und des RSA-PKCS-Signaturmechanismus zu generieren.

Um den `crypto sign rsa-pkcs` Befehl verwenden zu können, benötigen Sie zunächst einen privaten RSA-Schlüssel in Ihrem Cluster. AWS CloudHSM Sie können mit dem [Generieren Sie ein asymmetrisches RSA-Schlüsselpaar mit CloudHSM CLI](#) Befehl, dessen `sign` Attribut auf `gesetzt` ist, einen privaten RSA-Schlüssel generieren. `true`

Note

Signaturen können AWS CloudHSM mit [Die Kategorie Crypto Verify in CloudHSM CLI](#) Unterbefehlen verifiziert werden.

Benutzertyp

Die folgenden Benutzertypen können diesen Befehl ausführen.

- Krypto-Benutzer () CUs

Voraussetzungen

- Um diesen Befehl auszuführen, müssen Sie als CU angemeldet sein.

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

Beispiel

Diese Beispiele zeigen, wie eine Signatur mithilfe des RSA-PKCS-Signaturmechanismus und SHA256 der Hash-Funktion generiert wird. `crypto sign rsa-pkcs` Dieser Befehl verwendet einen privaten Schlüssel im HSM.

Example Beispiel: Generieren Sie eine Signatur für Base-64-kodierte Daten

```
aws-cloudhsm > crypto sign rsa-pkcs --key-filter attr.label=rsa-private --hash-function sha256 --data YWJjMTIz
{
  "error_code": 0,
  "data": {
    "key-reference": "0x000000000007008db",
    "signature": "XJ7mRyHnDRYrDWTQuuNb
+5mhoXx7VTsPMjg0QW4iMN7E42eNHj2Q0oovMmBdHUEH0F4HYG8FBJ0BhvGuM8J/
z6y41GbowVpUT6WzjnIQs79K9i7i6oR1TYjLnIS3r/zkimuXcS8/ZxyDzru+G09BUT9FFU/
of9cvu40yn6a5+IXuCbKNQs19uASuFARUTZ0a0Ny1CB1MulxUpqGTmI91J6evlP7k/2khwDmJ5E8FEar5/
Cvbn9t21p3Uj561ngTXrYbIZ2KHpef9jQh/cEIvFLG61sexJjQi8EdTxeDA
+I3IT00qrvvESvA9+Sj7kdG2ceIicFS8/8LwyxiIC31UHQ=="
  }
}
```

Example Beispiel: Generieren Sie eine Signatur für eine Datendatei

```
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": "0x000000000007008db",
    "signature": "XJ7mRyHnDRYrDWTQuuNb
+5mhoXx7VTsPMjg0QW4iMN7E42eNHj2Q0oovMmBdHUEH0F4HYG8FBJ0BhvGuM8J/
z6y41GbowVpUT6WzjnIQs79K9i7i6oR1TYjLnIS3r/zkimuXcS8/ZxyDzru+G09BUT9FFU/
of9cvu40yn6a5+IXuCbKNQs19uASuFARUTZ0a0Ny1CB1MulxUpqGTmI91J6evlP7k/2khwDmJ5E8FEar5/
Cvbn9t21p3Uj561ngTXrYbIZ2KHpef9jQh/cEIvFLG61sexJjQi8EdTxeDA
+I3IT00qrvvESvA9+Sj7kdG2ceIicFS8/8LwyxiIC31UHQ=="
  }
}
```

Argumente

<CLUSTER_ID>

Die ID des Clusters, auf dem dieser Vorgang ausgeführt werden soll.

Erforderlich: Wenn mehrere Cluster [konfiguriert wurden](#).

<DATA>

Base64-kodierte Daten, die signiert werden sollen.

Erforderlich: Ja (sofern nicht über den Datenpfad angegeben)

<DATA_PATH>

Gibt den Speicherort der zu signierenden Daten an.

Erforderlich: Ja (sofern nicht in Form von Daten angegeben)

<HASH_FUNCTION>

Spezifiziert die Hash-Funktion.

Zulässige Werte:

- sha1
- sha224
- sha256
- sha384
- sha512

Erforderlich: Ja

<KEY_FILTER>

Schlüsselreferenz (z. B. `key-reference=0xabc`) oder durch Leerzeichen getrennte Liste von Schlüsselattributen in der Form `attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE` zur Auswahl eines passenden Schlüssels.

Eine Liste der unterstützten CloudHSM-CLI-Schlüsselattribute finden Sie unter Schlüsselattribute für CloudHSM CLI.

Erforderlich: Ja

<APPROVAL>

Gibt den Dateipfad zu einer signierten Quorum-Token-Datei an, um den Vorgang zu genehmigen. Nur erforderlich, wenn der Quorumwert des privaten Schlüssels für den Schlüsselnutzungsdienst größer als 1 ist.

<DATA_TYPE>

Gibt an, ob der Wert des Datenparameters als Teil des Signaturalgorithmus gehasht werden soll. Wird `raw` für Daten ohne Hashwert verwendet; wird `digest` für Digests verwendet, die bereits gehasht wurden.

Für RSA-PKCS müssen die Daten im DER-codierten Format übergeben werden, wie in RFC 8017, Abschnitt 9.2 spezifiziert

Zulässige Werte:

- RAW
- Digest

Verwandte Themen

- [Die Kategorie Kryptozeichen in CloudHSM CLI](#)
- [Die Kategorie Crypto Verify in CloudHSM CLI](#)

Generieren Sie eine Signatur mit dem RSA-PKCS-PSS Mechanismus in CloudHSM CLI

Verwenden Sie den `crypto sign rsa-pkcs-pss` Befehl in der CloudHSM-CLI, um mithilfe eines privaten RSA-Schlüssels und des RSA-PKCS-PSS Signaturmechanismus eine Signatur zu generieren.

Um den `crypto sign rsa-pkcs-pss` Befehl verwenden zu können, benötigen Sie zunächst einen privaten RSA-Schlüssel in Ihrem Cluster. AWS CloudHSM Sie können mit dem [Generieren Sie ein asymmetrisches RSA-Schlüsselpaar mit CloudHSM CLI](#) Befehl, dessen `sign` Attribut auf `true` gesetzt ist, einen privaten RSA-Schlüssel generieren.

Note

Signaturen können AWS CloudHSM mit [Die Kategorie Crypto Verify in CloudHSM CLI](#) Unterbefehlen verifiziert werden.

Benutzertyp

Die folgenden Benutzertypen können diesen Befehl ausführen.

- Krypto-Benutzer () CUs

Voraussetzungen

- Um diesen Befehl auszuführen, müssen Sie als CU angemeldet sein.

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:

<code>--cluster-id <CLUSTER_ID></code>	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
<code>--key-filter [<KEY_FILTER>...]</code>	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
<code>--hash-function <HASH_FUNCTION></code>	[possible values: sha1, sha224, sha256, sha384, sha512]
<code>--data-path <DATA_PATH></code>	The path to the file containing the data to be signed
<code>--data <DATA></code>	Base64 Encoded data to be signed
<code>--mgf <MGF></code>	The mask generation function [possible values: mgf1-sha1, mgf1-sha224, mgf1-sha256, mgf1-sha384, mgf1-sha512]
<code>--salt-length <SALT_LENGTH></code>	The salt length
<code>--approval <APPROVAL></code>	Filepath of signed quorum token file to approve operation
<code>--data-type <DATA_TYPE></code>	The type of data passed in, either raw or digest [possible values: raw, digest]
<code>-h, --help</code>	Print help

Beispiel

Diese Beispiele zeigen, wie eine Signatur mithilfe des RSA-PKCS-PSS Signaturmechanismus und der SHA256 Hash-Funktion generiert wird. `crypto sign rsa-pkcs-pss` Dieser Befehl verwendet einen privaten Schlüssel im HSM.

Example Beispiel: Generieren Sie eine Signatur für Base-64-kodierte Daten

```
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": "0x000000000007008db",
    "signature": "H/z1rYVMzNAa31K4amE5MTiwGxDdCTgQXCJXRBKV0Vm7ZuyI0fGE4sT/BUN
+977mQEV2TqtWpTsiF2IpwGM1VfSBRT7h/g4o6YERm1tTQL17q+AJ7uGGK37zCsWQrAo7Vy8NzPSHxekePo/
ZegrB1aHWN1fE8H3IPUKqLuMDI9o1Jq6kM986ExS7Yme0Ic1cZkyykTWqHLQVL2C3+A2bHJZBqRcM5XoIpk8HkPypjpN
+m4FNUds30GAemo0M16asSrEJSthaZWV530BsD0qzA8Rt8JdhXS+GZp3vNLdL10TBELDPweXVgAu4dBX0F0vpw/
gg6sNvuaDK4Y0Bv2fqKg=="
  }
}
```

Example Beispiel: Generieren Sie eine Signatur für eine Datendatei

```
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": "0x000000000007008db",
    "signature": "H/z1rYVMzNAa31K4amE5MTiwGxDdCTgQXCJXRBKV0Vm7ZuyI0fGE4sT/BUN
+977mQEV2TqtWpTsiF2IpwGM1VfSBRT7h/g4o6YERm1tTQL17q+AJ7uGGK37zCsWQrAo7Vy8NzPSHxekePo/
ZegrB1aHWN1fE8H3IPUKqLuMDI9o1Jq6kM986ExS7Yme0Ic1cZkyykTWqHLQVL2C3+A2bHJZBqRcM5XoIpk8HkPypjpN
+m4FNUds30GAemo0M16asSrEJSthaZWV530BsD0qzA8Rt8JdhXS+GZp3vNLdL10TBELDPweXVgAu4dBX0F0vpw/
gg6sNvuaDK4Y0Bv2fqKg=="
  }
}
```

Argumente

<CLUSTER_ID>

Die ID des Clusters, auf dem dieser Vorgang ausgeführt werden soll.

Erforderlich: Wenn mehrere Cluster [konfiguriert wurden](#).

<DATA>

Base64-kodierte Daten, die signiert werden sollen.

Erforderlich: Ja (sofern nicht über den Datenpfad angegeben)

<DATA_PATH>

Gibt den Speicherort der zu signierenden Daten an.

Erforderlich: Ja (sofern nicht in Form von Daten angegeben)

<HASH_FUNCTION>

Spezifiziert die Hash-Funktion.

Zulässige Werte:

- sha1
- sha224
- sha256
- sha384
- sha512

Erforderlich: Ja

<KEY_FILTER>

Schlüsselreferenz (z. B. `key-reference=0xabc`) oder durch Leerzeichen getrennte Liste von Schlüsselattributen in der Form `attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE` zur Auswahl eines passenden Schlüssels.

Eine Liste der unterstützten CloudHSM-CLI-Schlüsselattribute finden Sie unter Schlüsselattribute für CloudHSM CLI.

Erforderlich: Ja

<MGF>

Spezifiziert die Funktion zur Maskengenerierung.

Note

Die Hash-Funktion der Maskengenerierungsfunktion muss mit der Hash-Funktion des Signaturmechanismus übereinstimmen.

Zulässige Werte:

- mgf1-sha1
- mgf1-sha224
- mgf1-sha256
- mgf1-sha384
- mgf1-sha512

Erforderlich: Ja

<SALT_LENGTH>

Gibt die Länge des Salzes an.

Erforderlich: Ja

<APPROVAL>

Gibt den Dateipfad zu einer signierten Quorum-Token-Datei an, um den Vorgang zu genehmigen. Nur erforderlich, wenn der Quorumwert des privaten Schlüssels für den Schlüsselverwendungsdienst größer als 1 ist.

<DATA_TYPE>

Gibt an, ob der Wert des Datenparameters als Teil des Signaturalgorithmus gehasht werden soll. Wird `raw` für Daten ohne Hashwert verwendet; wird `digest` für Digests verwendet, die bereits gehasht wurden.

Zulässige Werte:

- RAW
- Digest

Verwandte Themen

- [Die Kategorie Kryptozeichen in CloudHSM CLI](#)
- [Die Kategorie Crypto Verify in CloudHSM CLI](#)

Verwandte Themen

- [Die Kategorie Crypto Verify in CloudHSM CLI](#)

Die Kategorie Crypto Verify in CloudHSM CLI

In der CloudHSM-CLI crypto verify ist dies eine übergeordnete Kategorie für eine Gruppe von Befehlen, die in Kombination mit der übergeordneten Kategorie bestätigt, ob eine Datei mit einem bestimmten Schlüssel signiert wurde. crypto verify hat die folgenden Unterbefehle:

- [kryptoverifizieren \(ecdsa\)](#)
- [kryptoverifizieren Sie ed25519ph](#)
- [kryptoverifizieren Sie RSA-PKCS](#)
- [kryptoverifizieren rsa-pkcs-pss](#)

Der crypto verify Befehl vergleicht eine signierte Datei mit einer Quelldatei und analysiert anhand eines bestimmten öffentlichen Schlüssels und Signierungsmechanismus, ob sie kryptografisch verwandt sind.

Note

Dateien können AWS CloudHSM mit dem Vorgang angemeldet werden. [Die Kategorie Kryptozeichen in CloudHSM CLI](#)

Überprüfen Sie eine mit dem ECDSA-Mechanismus signierte Signatur in der CloudHSM-CLI

Verwenden Sie den crypto verify ecdsa Befehl in der CloudHSM-CLI, um die folgenden Vorgänge abzuschließen:

- Bestätigen Sie, dass eine Datei im HSM mit einem bestimmten öffentlichen Schlüssel signiert wurde.
- Stellen Sie sicher, dass die Signatur mithilfe des ECDSA-Signaturmechanismus generiert wurde.
- Vergleichen Sie eine signierte Datei mit einer Quelldatei und stellen Sie anhand eines bestimmten öffentlichen ECDSA-Schlüssels und Signaturmechanismus fest, ob die beiden kryptografisch verwandt sind.
- Die ECDSA-Überprüfungsfunktion erwartet die Signatur in dem Format `r | s`, in dem die Komponenten R und S als binäre Rohdaten verkettet werden.

Um den crypto verify ecdsa Befehl verwenden zu können, benötigen Sie zunächst einen öffentlichen EC-Schlüssel in Ihrem Cluster. AWS CloudHSM Sie können einen öffentlichen EC-Schlüssel

importieren, indem Sie den [Importieren Sie einen Schlüssel im PEM-Format mit der CloudHSM-CLI](#) Befehl verwenden, dessen `verify` Attribut auf `gesetzt ist true`.

Note

Sie können in der CloudHSM-CLI eine Signatur mit [Die Kategorie Kryptozeichen in CloudHSM CLI](#) Unterbefehlen generieren.

Benutzertyp

Die folgenden Benutzertypen können diesen Befehl ausführen.

- Crypto-Benutzer () CUs

Voraussetzungen

- Um diesen Befehl auszuführen, müssen Sie als CU angemeldet sein.

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

```

Beispiel

Diese Beispiele zeigen, wie eine Signatur verifiziert werden kann, die mithilfe des ECDSA-Signaturmechanismus und der SHA256 Hash-Funktion generiert wurde. `crypto verify ecdsa` Dieser Befehl verwendet einen öffentlichen Schlüssel im HSM.

Example Beispiel: Überprüfen Sie eine Base64-codierte Signatur mit Base64-codierten Daten

```

aws-cloudhsm > crypto verify ecdsa --hash-function sha256 --key-filter attr.label=ec-
public --data YWJjMTIz --signature 4zki+FzjhP7Z/KqoQvh4ueMAxQQVp7FQguZ2w0S3Q5bzk
+Hc5irV5iTkuxQbropPttVFZ8V6FgR2fz+sPegwCw==
{
  "error_code": 0,
  "data": {
    "message": "Signature verified successfully"
  }
}

```

Example Beispiel: Überprüfen Sie eine Signaturdatei mit einer Datendatei

```

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 Beispiel: Nachweis einer falschen Signaturbeziehung

Mit diesem Befehl wird überprüft, ob die unter befindlichen Daten mit einem öffentlichen Schlüssel mit der Bezeichnung signiert /home/data wurden. Dabei wird der ECDSA-Signaturmechanismus `ecdsa-public` verwendet, um die Signatur zu erzeugen, die sich in befindet. /home/signature Da die angegebenen Argumente keine echte Signaturbeziehung bilden, gibt der Befehl eine Fehlermeldung zurück.

```
aws-cloudhsm > crypto verify ecdsa --hash-function sha256 --  
key-filter attr.label=ec-public --data aW52YWxpZA== --signature  
+ogk7M7S3iTqFg3SndJfd91dZFr5Qo6YixJl8JwcvqqVgsVu06o+VKvTRjz0/V05kf3JJbBLr87Q  
+wLWcMAJfA==  
{  
  "error_code": 1,  
  "data": "Signature verification failed"  
}
```

Argumente

<CLUSTER_ID>

Die ID des Clusters, auf dem dieser Vorgang ausgeführt werden soll.

Erforderlich: Wenn mehrere Cluster [konfiguriert wurden](#).

<DATA>

Base64-kodierte Daten, die signiert werden sollen.

Erforderlich: Ja (sofern nicht über den Datenpfad angegeben)

<DATA_PATH>

Gibt den Speicherort der zu signierenden Daten an.

Erforderlich: Ja (sofern nicht über den Datenpfad angegeben)

<HASH_FUNCTION>

Gibt die Hash-Funktion an.

Zulässige Werte:

- sha1
- sha224

- sha256
- sha384
- sha512

Erforderlich: Ja

<KEY_FILTER>

Schlüsselreferenz (z. B. `key-reference=0xabc`) oder durch Leerzeichen getrennte Liste von Schlüsselattributen in der Form `attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE` zur Auswahl eines passenden Schlüssels.

Eine Liste der unterstützten CloudHSM-CLI-Schlüsselattribute finden Sie unter Schlüsselattribute für CloudHSM CLI.

Erforderlich: Ja

<SIGNATURE>

Base64-kodierte Signatur.

Erforderlich: Ja (sofern nicht über den Signaturpfad angegeben)

<SIGNATURE_PATH>

Gibt den Speicherort der Signatur an.

Erforderlich: Ja (sofern nicht im Signaturpfad angegeben)

<DATA_TYPE>

Gibt an, ob der Wert des Datenparameters als Teil des Signaturalgorithmus gehasht werden soll. Wird `raw` für Daten ohne Hashwert verwendet; wird `digest` für Digests verwendet, die bereits gehasht wurden.

Zulässige Werte:

- RAW
- Digest

Verwandte Themen

- [Die Kategorie Kryptozeichen in CloudHSM CLI](#)
- [Die Kategorie Crypto Verify in CloudHSM CLI](#)

Überprüfen Sie eine mit dem HashEd DSA-Mechanismus signierte Signatur in der CloudHSM-CLI

 Important

HashEdVorgänge zur Überprüfung der DSA-Signatur werden nur auf hsm2m.medium-Instances im Nicht-FIPS-Modus unterstützt.

Verwenden Sie den `crypto verify ed25519ph` Befehl in der CloudHSM-CLI, um die folgenden Vorgänge abzuschließen:

- Überprüfen Sie die Signaturen von Daten oder Dateien mithilfe eines bestimmten öffentlichen Ed25519-Schlüssels.
- Vergewissern Sie sich, dass die Signatur mithilfe des HashEd DSA-Signaturmechanismus generiert wurde. Weitere Informationen zu HashEd DSA finden Sie unter [NIST SP 186-5](#), Abschnitt 7.8.

Um den `crypto verify ed25519ph` Befehl verwenden zu können, benötigen Sie zunächst einen öffentlichen Ed25519-Schlüssel in Ihrem Cluster. AWS CloudHSM Sie können ein Ed25519-Schlüsselpaar mit dem [Generieren Sie ein asymmetrisches EC-Schlüsselpaar mit CloudHSM CLI](#) Befehl generieren, bei dem der `curve` Parameter auf `ed25519` und das `verify` Attribut auf `gesetzt` ist `true`, oder einen öffentlichen Ed25519-Schlüssel mit dem [Importieren Sie einen Schlüssel im PEM-Format mit der CloudHSM-CLI](#) Befehl importieren, bei dem das Attribut auf `gesetzt` ist. `verify true`

 Note

Sie können in der CloudHSM-CLI eine Signatur mit [Die Kategorie Kryptozeichen in CloudHSM CLI](#) Unterbefehlen generieren.

Benutzertyp

Die folgenden Benutzertypen können diesen Befehl ausführen.

- Crypto-Benutzer () CUs

Voraussetzungen

- Um diesen Befehl auszuführen, müssen Sie als CU angemeldet sein.
- HashEdVorgänge zur Überprüfung der DSA-Signatur werden nur auf hsm2m.medium-Instances im Nicht-FIPS-Modus unterstützt.

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

Beispiel

Diese Beispiele zeigen, wie eine Signatur verifiziert wird, `crypto verify ed25519ph` die mit dem Signaturmechanismus und der Hash-Funktion ED25519PH generiert wurde. `sha512` Dieser Befehl verwendet einen öffentlichen Ed25519-Schlüssel im HSM.

Example Beispiel: Überprüfen Sie eine Base64-codierte Signatur mit Base64-codierten Daten

```
aws-cloudhsm > crypto verify ed25519ph \
--hash-function sha512 \
--key-filter attr.label=ed25519-public \
--data-type raw \
--data YWJj \
--signature mKcCIvC4Ehqp0w+BPWg/gJ5GK0acf/h20UmbuU5trkEx
+FBCRjwqNVogA9BirfWqoQuMYeY2Biqq0RwqJgg0Bg==
{
  "error_code": 0,
  "data": {
    "message": "Signature verified successfully"
  }
}
```

Example Beispiel: Überprüfen Sie eine Signaturdatei mit einer Datendatei

```
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"
  }
}
```

Argumente

<CLUSTER_ID>

Die ID des Clusters, auf dem dieser Vorgang ausgeführt werden soll.

Erforderlich: Wenn mehrere Cluster [konfiguriert wurden](#).

<DATA>

Base64-kodierte Daten müssen verifiziert werden.

Erforderlich: Ja (sofern nicht über den Datenpfad angegeben)

<DATA_PATH>

Gibt den Speicherort der zu überprüfenden Daten an.

Erforderlich: Ja (sofern nicht über den Datenparameter angegeben)

<HASH_FUNCTION>

Gibt die Hash-Funktion an. ED25519Ph unterstützt nur. SHA512

Zulässige Werte:

- sha512

Erforderlich: Ja

<KEY_FILTER>

Schlüsselreferenz (z. B. `key-reference=0xabc`) oder durch Leerzeichen getrennte Liste von Schlüsselattributen in der Form `attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE` zur Auswahl eines passenden Schlüssels.

Eine Liste der unterstützten CloudHSM-CLI-Schlüsselattribute finden Sie unter. [Schlüsselattribute für CloudHSM-CLI](#)

Erforderlich: Ja

<SIGNATURE>

Base64-kodierte Signatur.

Erforderlich: Ja (sofern nicht im Signaturpfad angegeben)

<SIGNATURE_PATH>

Gibt den Speicherort der Signatur an.

Erforderlich: Ja (sofern nicht über den Signaturparameter angegeben)

<DATA_TYPE>

Gibt an, ob der Wert des Datenparameters als Teil des Überprüfungsalgorithmus gehasht werden soll. Wird `raw` für Daten ohne Hashwert verwendet; wird `digest` für Digests verwendet, die bereits gehasht wurden.

Zulässige Werte:

- RAW
- Digest

Erforderlich: Ja

Verwandte Themen

- [Die Kategorie Kryptozeichen in CloudHSM CLI](#)
- [Die Kategorie Crypto Verify in CloudHSM CLI](#)
- [Generieren Sie eine Signatur mit dem HashEd DSA-Mechanismus in der CloudHSM-CLI](#)

Überprüfen Sie eine mit dem RSA-PKCS-Mechanismus signierte Signatur in der CloudHSM-CLI

Verwenden Sie den `crypto verify rsa-pkcs` Befehl in der CloudHSM-CLI, um die folgenden Vorgänge abzuschließen:

- Bestätigen Sie, dass eine Datei im HSM mit einem bestimmten öffentlichen Schlüssel signiert wurde.
- Stellen Sie sicher, dass die Signatur mithilfe des RSA-PKCS Signaturmechanismus generiert wurde.
- Vergleicht eine signierte Datei mit einer Quelldatei und ermittelt anhand eines bestimmten öffentlichen RSA-Schlüssels und Signaturmechanismus, ob die beiden kryptografisch verwandt sind.

Um den `crypto verify rsa-pkcs` Befehl verwenden zu können, müssen Sie zunächst über einen öffentlichen RSA-Schlüssel in Ihrem Cluster verfügen. AWS CloudHSM

Note

Sie können mithilfe der CloudHSM-CLI mit den Unterbefehlen eine Signatur generieren. [Die Kategorie Kryptozeichen in CloudHSM CLI](#)

Benutzertyp

Die folgenden Benutzertypen können diesen Befehl ausführen.

- Crypto-Benutzer () CUs

Voraussetzungen

- Um diesen Befehl auszuführen, müssen Sie als CU angemeldet sein.

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

```

Beispiel

Diese Beispiele zeigen, wie eine Signatur verifiziert werden kann, die mithilfe des RSA-PKCS-Signaturmechanismus und SHA256 der Hash-Funktion generiert wurde. `crypto verify rsa-pkcs` Dieser Befehl verwendet einen öffentlichen Schlüssel im HSM.

Example Beispiel: Überprüfen Sie eine Base64-codierte Signatur mit Base64-codierten Daten

```

aws-cloudhsm > crypto verify rsa-pkcs --hash-function sha256 --key-filter
attr.label=rsa-public --data YWJjMTIz --signature XJ7mRyHnDRYrDWTQuuNb
+5mhoXx7VTsPMjg0QW4iMn7E42eNHj2Q0oovMmBdHUEH0F4HYG8FBj0BhvGuM8J/
z6y41GbowVpUT6WzjnIQs79K9i7i6oR1TYjLnIS3r/zkimuXcS8/ZxyDzru+G09BUT9FFU/
of9cvu40yn6a5+IXuCbKNQs19uASuFARUTZ0a0Ny1CB1MulxUpqGTmI91J6evlP7k/2khwDmJ5E8FEar5/
Cvbn9t21p3Uj561ngTXrYbIZ2KHpef9jQh/cEIvFLG61sexJjQi8EdTxeDA
+I3IT00qrvvESvA9+Sj7kdG2ceIicFS8/8LwyxiIC31UHQ==
{
  "error_code": 0,
  "data": {
    "message": "Signature verified successfully"
  }
}

```

Example Beispiel: Verifizieren Sie eine Signaturdatei mit einer Datendatei

```

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 Beispiel: Beweisen Sie die Beziehung zwischen falschen Signaturen

Mit diesem Befehl wird überprüft, ob die ungültigen Daten mit einem öffentlichen Schlüssel mit der Bezeichnung signiert wurden. Dabei wird der RSAKCS-Signaturmechanismus `rsa-public` verwendet, um die Signatur zu erzeugen, die sich in befindet. `/home/signature` Da die angegebenen Argumente keine echte Signaturbeziehung bilden, gibt der Befehl eine Fehlermeldung zurück.

```
aws-cloudhsm > crypto verify rsa-pkcs --hash-function sha256 --key-filter
attr.label=rsa-public --data aW52YWxpZA== --signature XJ7mRyHnDRYrDWTQuuNb
+5mhoXx7VTsPMjg0QW4iMN7E42eNHj2Q0oovMmBdHUEH0F4HYG8FBj0BhvGuM8J/
z6y41GbowVpUT6WzjnIQs79K9i7i6oR1TYjLnIS3r/zkimuXcS8/ZxyDzru+G09BUT9FFU/
of9cvu40yn6a5+IXuCbKNQs19uASuFARUTZ0a0Ny1CB1MulxUpqGTmI91J6ev1P7k/2khwDmJ5E8FEar5/
Cvbn9t21p3Uj561ngTXrYbIZ2KHpef9jQh/cEIvFLG61sexJjQi8EdTxeDA
+I3IT00qrvvESvA9+Sj7kdG2ceIicFS8/8LwyxiIC31UHQ==
{
  "error_code": 1,
  "data": "Signature verification failed"
}
```

Argumente

<CLUSTER_ID>

Die ID des Clusters, auf dem dieser Vorgang ausgeführt werden soll.

Erforderlich: Wenn mehrere Cluster [konfiguriert wurden](#).

<DATA>

Base64-kodierte Daten, die signiert werden sollen.

Erforderlich: Ja (sofern nicht über den Datenpfad angegeben)

<DATA_PATH>

Gibt den Speicherort der zu signierenden Daten an.

Erforderlich: Ja (sofern nicht über den Datenpfad angegeben)

<HASH_FUNCTION>

Gibt die Hash-Funktion an.

Zulässige Werte:

- sha1
- sha224
- sha256
- sha384
- sha512

Erforderlich: Ja

<KEY_FILTER>

Schlüsselreferenz (z. B. `key-reference=0xabc`) oder durch Leerzeichen getrennte Liste von Schlüsselattributen in der Form `attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE` zur Auswahl eines passenden Schlüssels.

Eine Liste der unterstützten CloudHSM-CLI-Schlüsselattribute finden Sie unter Schlüsselattribute für CloudHSM CLI.

Erforderlich: Ja

<SIGNATURE>

Base64-kodierte Signatur.

Erforderlich: Ja (sofern nicht über den Signaturpfad angegeben)

<SIGNATURE_PATH>

Gibt den Speicherort der Signatur an.

Erforderlich: Ja (sofern nicht über den Signaturpfad angegeben)

<DATA_TYPE>

Gibt an, ob der Wert des Datenparameters als Teil des Signaturalgorithmus gehasht werden soll. Wird `raw` für Daten ohne Hashwert verwendet; wird `digest` für Digests verwendet, die bereits gehasht wurden.

[Für RSA-PKCS müssen die Daten im DER-codierten Format übergeben werden, wie in RFC 8017, Abschnitt 9.2 spezifiziert](#)

Zulässige Werte:

- RAW

- Digest

Verwandte Themen

- [Die Kategorie Kryptozeichen in CloudHSM CLI](#)
- [Die Kategorie Crypto Verify in CloudHSM CLI](#)

Verifizieren Sie eine Signatur, die mit dem RSA-PKCS-PSS Mechanismus in der CloudHSM-CLI signiert wurde

Verwenden Sie den `crypto sign rsa-pkcs-pss` Befehl in der CloudHSM-CLI, um die folgenden Vorgänge abzuschließen.

- Bestätigen Sie, dass eine Datei im HSM mit einem bestimmten öffentlichen Schlüssel signiert wurde.
- Stellen Sie sicher, dass die Signatur mithilfe des RSA-PKCS-PSS Signaturmechanismus generiert wurde.
- Vergleicht eine signierte Datei mit einer Quelldatei und ermittelt anhand eines bestimmten öffentlichen RSA-Schlüssels und Signaturmechanismus, ob die beiden kryptografisch verwandt sind.

Um den `crypto verify rsa-pkcs-pss` Befehl verwenden zu können, müssen Sie zunächst über einen öffentlichen RSA-Schlüssel in Ihrem Cluster verfügen. AWS CloudHSM Sie können einen öffentlichen RSA-Schlüssel mit dem Befehl `key import pem (ADD UNWRAP LINK HERE)` importieren, wobei das `verify` Attribut auf gesetzt ist. `true`

Note

Sie können mithilfe der CloudHSM-CLI mit den Unterbefehlen eine Signatur generieren. [Die Kategorie Kryptozeichen in CloudHSM CLI](#)

Benutzertyp

Die folgenden Benutzertypen können diesen Befehl ausführen.

- Crypto-Benutzer () CUs

Voraussetzungen

- Um diesen Befehl auszuführen, müssen Sie als CU angemeldet sein.

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

Beispiel

Diese Beispiele zeigen, wie eine Signatur verifiziert werden kann, die mithilfe des RSA-PKCS-PSS Signaturmechanismus und der SHA256 Hash-Funktion generiert wurde. `crypto verify rsa-pkcs-pss` Dieser Befehl verwendet einen öffentlichen Schlüssel im HSM.

Example Beispiel: Überprüfen Sie eine Base64-codierte Signatur mit Base64-codierten Daten

```
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/z1rYVMzNAa31K4amE5MTiwGxDdCTgQXCJXRBKV0Vm7ZuyI0fGE4sT/BUN
+977mQEV2TqtWpTsiF2IpwGM1VfSBrt7h/g4o6YERm1tTQL17q+AJ7uGGK37zCsWQrAo7Vy8NzPSHxekePo/
ZegrB1aHWN1fE8H3IPUKqLuMDI9o1Jq6kM986ExS7Yme0IclcZkyykTWqHLQVL2C3+A2bHJZBqRcM5XoIpk8HkPypjpN
+m4FNUds30GAemo0M16asSrEJSthaZWV530BsD0qzA8Rt8JdhXS+GZp3vNLdL10TBELDPweXVgAu4dBX0F0vpw/
gg6sNvuaDK4Y0Bv2fqKg==
{
  "error_code": 0,
  "data": {
    "message": "Signature verified successfully"
  }
}
```

Example Beispiel: Verifizieren Sie eine Signaturdatei mit einer Datendatei

```
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
signature-file
{
  "error_code": 0,
  "data": {
    "message": "Signature verified successfully"
  }
}
```

Example Beispiel: Nachweis einer falschen Signaturbeziehung

Mit diesem Befehl wird überprüft, ob die ungültigen Daten mit einem öffentlichen Schlüssel mit der Bezeichnung signiert wurden. Dabei wird der RSAPKCS-PSS-Signaturmechanismus `rsa-public` verwendet, um die Signatur zu erzeugen, die sich in befindet. `/home/signature` Da die

angegebenen Argumente keine echte Signaturbeziehung bilden, gibt der Befehl eine Fehlermeldung zurück.

```
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/z1rYVMzNAa31K4amE5MTiwGxDdCTgQXCJXRBKV0Vm7ZuyI0fGE4sT/BUN
+977mQEV2TqtWpTsiF2IpwGM1VfSBrt7h/g4o6YERm1tTQL17q+AJ7uGGK37zCsWQrAo7Vy8NzPSHxekePo/
ZegrB1aHWN1fE8H3IPUKqLuMDI9o1Jq6kM986ExS7Yme0Ic1cZkyykTWqHLQVL2C3+A2bHJZBqRcM5XoIpk8HkPypjpN
+m4FNuds30GAemo0M16asSrEJSthaZWV530BsD0qzA8Rt8JdhXS+GZp3vNLdL10TBELDPweXVgAu4dBX0F0vpw/
gg6sNvuaDK4Y0Bv2fqKg==
{
  "error_code": 1,
  "data": "Signature verification failed"
}
```

Argumente

<CLUSTER_ID>

Die ID des Clusters, auf dem dieser Vorgang ausgeführt werden soll.

Erforderlich: Wenn mehrere Cluster [konfiguriert wurden](#).

<DATA>

Base64-kodierte Daten, die signiert werden sollen.

Erforderlich: Ja (sofern nicht über den Datenpfad angegeben)

<DATA_PATH>

Gibt den Speicherort der zu signierenden Daten an.

Erforderlich: Ja (sofern nicht über den Datenpfad angegeben)

<HASH_FUNCTION>

Gibt die Hash-Funktion an.

Zulässige Werte:

- sha1
- sha224
- sha256

- sha384
- sha512

Erforderlich: Ja

<KEY_FILTER>

Schlüsselreferenz (z. B. `key-reference=0xabc`) oder durch Leerzeichen getrennte Liste von Schlüsselattributen in der Form `attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE` zur Auswahl eines passenden Schlüssels.

Eine Liste der unterstützten CloudHSM-CLI-Schlüsselattribute finden Sie unter Schlüsselattribute für CloudHSM CLI.

Erforderlich: Ja

<MGF>

Spezifiziert die Funktion zur Maskengenerierung.

Note

Die Hash-Funktion der Maskengenerierungsfunktion muss mit der Hash-Funktion des Signaturmechanismus übereinstimmen.

Zulässige Werte:

- mgf1-sha1
- mgf1-sha224
- mgf1-sha256
- mgf1-sha384
- mgf1-sha512

Erforderlich: Ja

<SIGNATURE>

Base64-kodierte Signatur.

Erforderlich: Ja (sofern nicht über den Signaturpfad angegeben)

<SIGNATURE_PATH>

Gibt den Speicherort der Signatur an.

Erforderlich: Ja (sofern nicht über den Signaturpfad angegeben)

<DATA_TYPE>

Gibt an, ob der Wert des Datenparameters als Teil des Signaturalgorithmus gehasht werden soll. Wird `raw` für Daten ohne Hashwert verwendet; wird `digest` für Digests verwendet, die bereits gehasht wurden.

Zulässige Werte:

- RAW
- Digest

Verwandte Themen

- [Die Kategorie Kryptozeichen in CloudHSM CLI](#)
- [Die Kategorie Crypto Verify in CloudHSM CLI](#)

Die Schlüsselkategorie in CloudHSM CLI

In der CloudHSM-CLI key ist dies eine übergeordnete Kategorie für eine Gruppe von Befehlen, die in Kombination mit der übergeordneten Kategorie einen schlüsselspezifischen Befehl erstellen. Derzeit besteht diese Kategorie aus den folgenden Befehlen:

- [delete](#)
- [generate-file](#)
- [key generate-asymmetric-pair](#)
 - [Schlüssel rsa generate-asymmetric-pair](#)
 - [Schlüssel usw. generate-asymmetric-pair](#)
- [key generate-symmetric](#)
 - [key generate-symmetric aes](#)
 - [key generate-symmetric generic-secret](#)
- [PEM importieren](#)
- [list](#)

- [Replikat](#)
- [set-attribute](#)
- [share](#)
- [unshare](#)
- [auspacken](#)
- [umwickeln](#)

Löschen Sie einen Schlüssel mit CloudHSM CLI

Verwenden Sie den `key delete` Befehl in der CloudHSM-CLI, um einen Schlüssel aus einem AWS CloudHSM Cluster zu löschen. Sie können nur jeweils einen Schlüssel löschen. Das Löschen eines Schlüssels in einem Schlüsselpaar hat keine Auswirkungen auf den anderen Schlüssel in diesem Paar.

Nur der CU, der den Schlüssel erstellt hat und somit Eigentümer des Schlüssels ist, kann den Schlüssel löschen. Benutzer, die den Schlüssel gemeinsam nutzen, ihn aber nicht besitzen, können den Schlüssel für kryptografische Operationen verwenden, ihn aber nicht löschen.

Benutzertyp

Die folgenden Benutzertypen können diesen Befehl ausführen.

- Crypto-Benutzer () CUs

Voraussetzungen

- Um diesen Befehl auszuführen, müssen Sie als CU angemeldet sein.

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
```

Beispiel

```
aws-cloudhsm > key delete --filter attr.label="ec-test-public-key"
{
  "error_code": 0,
  "data": {
    "message": "Key deleted successfully"
  }
}
```

Argumente

<CLUSTER_ID>

Die ID des Clusters, auf dem dieser Vorgang ausgeführt werden soll.

Erforderlich: Wenn mehrere Cluster [konfiguriert wurden](#).

<FILTER>

Schlüsselreferenz (z. B. key-reference=0xabc) oder durch Leerzeichen getrennte Liste von Schlüsselattributen in der Form attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE, dass ein passender Schlüssel zum Löschen ausgewählt werden soll.

Eine Liste der unterstützten CloudHSM-CLI-Schlüsselattribute finden Sie unter [Schlüsselattribute für CloudHSM-CLI](#)

Erforderlich: Ja

Verwandte Themen

- [Schlüssel für einen Benutzer mit CloudHSM CLI auflisten](#)
- [Exportieren Sie einen asymmetrischen Schlüssel mit CloudHSM CLI](#)
- [Freigabe eines Schlüssels mithilfe der CloudHSM-CLI rückgängig machen](#)
- [Schlüsselattribute für CloudHSM-CLI](#)
- [Schlüssel mithilfe der CloudHSM-CLI filtern](#)

Exportieren Sie einen asymmetrischen Schlüssel mit CloudHSM CLI

Verwenden Sie den `key generate-file` Befehl in der CloudHSM-CLI, um einen asymmetrischen Schlüssel aus dem Hardware-Sicherheitsmodul (HSM) zu exportieren. Wenn das Ziel ein privater Schlüssel ist, wird der Verweis auf den privaten Schlüssel im gefälschten PEM-Format exportiert. Wenn das Ziel ein öffentlicher Schlüssel ist, werden die Bytes des öffentlichen Schlüssels im PEM-Format exportiert.

Die gefälschte PEM-Datei, die nicht das eigentliche Material des privaten Schlüssels enthält, sondern stattdessen auf den privaten Schlüssel im HSM verweist, kann verwendet werden, um die Übertragung von Ihrem SSL/TLS Webserver auf zu einrichten. AWS CloudHSM Weitere Informationen finden Sie unter [SSL/TLS-Offloading](#).

Benutzertyp

Die folgenden Benutzertypen können diesen Befehl ausführen.

- Krypto-Benutzer () CUs

Voraussetzungen

Um diesen Befehl auszuführen, müssen Sie als CU angemeldet sein.

Syntax

```
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)

--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')

Beispiel

Dieses Beispiel zeigt, wie Sie key generate-file damit eine Schlüsseldatei in Ihrem AWS CloudHSM Cluster generieren können.

Example

```
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"
  }
}
```

Argumente

<CLUSTER_ID>

Die ID des Clusters, auf dem dieser Vorgang ausgeführt werden soll.

Erforderlich: Wenn mehrere Cluster [konfiguriert wurden](#).

<FILTER>

Schlüsselreferenz (z. B. key-reference=0xabc) oder durch Leerzeichen getrennte Liste von Schlüsselattributen in der Form attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE, dass ein passender Schlüssel zum Löschen ausgewählt werden soll.

Eine Liste der unterstützten CloudHSM-CLI-Schlüsselattribute finden Sie unter [Schlüsselattribute für CloudHSM-CLI](#).

Erforderlich: Nein

<ENCODING>

Gibt das Kodierungsformat für die Schlüsseldatei an

Erforderlich: Ja

<PATH>

Gibt den Dateipfad an, in den die Schlüsseldatei geschrieben wird

Erforderlich: Ja

Generieren von KSP-Schlüsselreferenzen (Windows)

 Note

Diese Funktion ist nur in SDK-Version 5.16.0 und höher verfügbar.

Voraussetzungen

- Sie können KSP-Schlüsselreferenzen nur auf Windows-Plattformen generieren.
- Sie müssen sich als Crypto-Benutzer (CU) anmelden.

Speicherort der Datei

Standardmäßig speichert AWS CloudHSM generierte Dateien in: C:\Users\Default\AppData \Roaming\Microsoft\Crypto\CaviumKSP\GlobalPartition

Verwenden Sie den Parameter, um einen anderen Speicherort anzugeben. --path

Syntax

```
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')

Beispiel — Generieren Sie eine KSP-Schlüsselreferenz mithilfe eines Attributfilters für einen privaten Schlüssel

Das folgende Beispiel generiert eine KSP-Schlüsselreferenz für einen privaten Schlüssel mit einer bestimmten Bezeichnung.

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"
  }
}
```

Beispiel — Generieren Sie KSP-Schlüsselreferenzen für alle Schlüsselpaare

Im folgenden Beispiel werden KSP-Schlüsselreferenzen für alle Schlüsselpaare in Ihrem Cluster generiert.

Example

```
aws-cloudhsm > key generate-file --encoding ksp-key-reference --all
{
  "error_code": 0,
  "data": {
    "message": "Successfully generated key file"
  }
}
```

Verwandte Themen

- [Schlüsselattribute für CloudHSM-CLI](#)
- [Schlüssel mithilfe der CloudHSM-CLI filtern](#)
- [Die generate-asymmetric-pair Kategorie in CloudHSM CLI](#)
- [Die Kategorie „Symmetrisch generieren“ in CloudHSM CLI](#)

Die generate-asymmetric-pair Kategorie in CloudHSM CLI

In der CloudHSM-CLI `key generate-asymmetric-pair` ist dies eine übergeordnete Kategorie für eine Gruppe von Befehlen, die in Kombination mit der übergeordneten Kategorie einen Befehl erstellen, der asymmetrische Schlüsselpaare generiert. Derzeit besteht diese Kategorie aus den folgenden Befehlen:

- [Schlüssel, ec generate-asymmetric-pair](#)
- [Schlüssel generate-asymmetric-pair RSA](#)

Generieren Sie ein asymmetrisches EC-Schlüsselpaar mit CloudHSM CLI

Verwenden Sie den `key asymmetric-pair ec` Befehl in der CloudHSM-CLI, um ein asymmetrisches EC-Schlüsselpaar (Elliptic Curve) in Ihrem Cluster zu generieren. AWS CloudHSM

Benutzertyp

Die folgenden Benutzertypen können diesen Befehl ausführen.

- Krypto-Benutzer () CUs

Voraussetzungen

Um diesen Befehl auszuführen, müssen Sie als CU angemeldet sein.

Syntax

```
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](#)

Beispiele

Diese Beispiele zeigen, wie Sie mit diesem `key generate-asymmetric-pair ec`-Befehl ein EC-Schlüsselpaar erstellen.

Example Beispiel: Ein EC-Schlüsselpaar erstellen

```
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": "0x0000000000012000b",
      "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": "0x0000000000012000c",
  "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 Beispiel: Erstellen Sie ein EC-Schlüsselpaar mit optionalen Attributen

```

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": "0x00000000000280c82",
  "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 Beispiel: Erstellen Sie ein EC-Schlüsselpaar mit Quorumwerten

Bei der Generierung eines Schlüssels mit Quorumsteuerungen muss der Schlüssel einer Mindestanzahl von Benutzern zugeordnet werden, die dem größten Schlüsselquorumwert entspricht. Zu den assoziierten Benutzern gehören der Schlüsselinhaber und Crypto-Benutzer, mit denen der Schlüssel geteilt wird. Um die Mindestanzahl der Benutzer zu ermitteln, mit denen der Schlüssel geteilt werden soll, ermitteln Sie den größten Quorumwert zwischen dem Quorumwert für die Schlüsselnutzung und dem Quorumwert für die Schlüsselverwaltung und subtrahieren Sie 1, um den Schlüsselbesitzer zu berücksichtigen, der standardmäßig mit dem Schlüssel verknüpft ist. Verwenden Sie den Befehl, um den Schlüssel mit mehreren Benutzern zu teilen. [Einen Schlüssel mit CloudHSM CLI teilen](#)

```
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": "0x0000000000002806eb",
      "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": "0x00000000000280c82",
  "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"
}
}
}

```

```
}
```

Argumente

<CLUSTER_ID>

Die ID des Clusters, auf dem dieser Vorgang ausgeführt werden soll.

Erforderlich: Wenn mehrere Cluster [konfiguriert wurden](#).

<CURVE>

Gibt die ID für die elliptische Kurve an.

- prime256v1
- secp256r1
- secp224r1
- secp384r1
- secp256k1
- secp521r1
- ed25519 (wird nur auf hsm2m.medium-Instances im Nicht-FIPS-Modus unterstützt)

Erforderlich: Ja

<PUBLIC_KEY_ATTRIBUTES>

Gibt eine durch Leerzeichen getrennte Liste von Schlüsselattributen an, die für den generierten öffentlichen EC-Schlüssel festgelegt werden sollen, in der Form von KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE (z. B. verify=true)

Eine Liste der unterstützten Schlüsselattribute finden Sie unter [Schlüsselattribute für CloudHSM-CLI](#).

Erforderlich: Nein

<PUBLIC_LABEL>

Gibt eine benutzerdefinierte Bezeichnung für den öffentlichen Schlüssel an. Die maximal zulässige Größe für Client SDK 5.11 und höher label beträgt 127 Zeichen. Das Client-SDK 5.10 und früher ist auf 126 Zeichen begrenzt.

Erforderlich: Ja

<PRIVATE_KEY_ATTRIBUTES>

Gibt eine durch Leerzeichen getrennte Liste von Schlüsselattributen an, die für den generierten privaten EC-Schlüssel festgelegt werden sollen, in der Form von KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE (z. B. sign=true)

Eine Liste der unterstützten Schlüsselattribute finden Sie unter [Schlüsselattribute für CloudHSM-CLI](#).

Erforderlich: Nein

<PRIVATE_LABEL>

Gibt eine benutzerdefinierte Bezeichnung für den private-key an. Die maximal zulässige Größe für Client-SDK 5.11 und höher label beträgt 127 Zeichen. Das Client-SDK 5.10 und früher ist auf 126 Zeichen begrenzt.

Erforderlich: Ja

<SESSION>

Erstellt einen Schlüssel, der nur in der aktuellen Sitzung existiert. Der Schlüssel kann nach Ende der Sitzung nicht wiederhergestellt werden.

Verwenden Sie diesen Parameter, wenn Sie einen Schlüssel zum Packen nur für kurze Zeit benötigen, z. B. einen Schlüssel, der einen anderen Schlüssel verschlüsselt und dann schnell entschlüsselt. Verwenden Sie keinen Sitzungsschlüssel, um Daten zu verschlüsseln, die Sie nach dem Ende der Sitzung möglicherweise entschlüsseln müssen.

Standardmäßig handelt es sich bei den generierten Schlüsseln um persistente (Token-)Schlüssel. Das Übergeben von <SESSION> ändert dies und stellt sicher, dass es sich bei einem mit diesem Argument generierten Schlüssel um einen (kurzlebigen) Sitzungsschlüssel handelt.

Erforderlich: Nein

<SHARE_CRYPTO_USERS>

Gibt eine durch Leerzeichen getrennte Liste von Crypto-Benutzernamen an, mit denen der private EC-Schlüssel gemeinsam genutzt werden soll

Erforderlich: Nein

<MANAGE_PRIVATE_KEY_QUORUM_VALUE>

Der Quorumwert für die Schlüsselverwaltungsvorgänge des privaten Schlüssels. Dieser Wert muss kleiner oder gleich der Anzahl der Benutzer sein, denen der Schlüssel zugeordnet ist. Dazu gehören Benutzer, mit denen der Schlüssel geteilt wird, und der Schlüsselbesitzer. Maximaler Wert von 8.

Erforderlich: Nein

<USE_PRIVATE_KEY_QUORUM_VALUE>

Der Quorumwert für die Schlüsselverwendungsvorgänge des privaten Schlüssels. Dieser Wert muss kleiner oder gleich der Anzahl der Benutzer sein, denen der Schlüssel zugeordnet ist. Dazu gehören Benutzer, mit denen der Schlüssel geteilt wird, und der Schlüsselbesitzer. Maximaler Wert von 8.

Erforderlich: Nein

Verwandte Themen

- [Schlüsselattribute für CloudHSM-CLI](#)
- [Schlüssel mithilfe der CloudHSM-CLI filtern](#)

Generieren Sie ein asymmetrisches RSA-Schlüsselpaar mit CloudHSM CLI

Verwenden Sie den `key generate-asymmetric-pair rsa` Befehl in der CloudHSM-CLI, um ein asymmetrisches RSA-Schlüsselpaar in Ihrem Cluster zu generieren. AWS CloudHSM

Benutzertyp

Die folgenden Benutzertypen können diesen Befehl ausführen.

- Krypto-Benutzer () CUs

Voraussetzungen

Um diesen Befehl auszuführen, müssen Sie als CU angemeldet sein.

Syntax

```
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

Beispiele

Diese Beispiele verdeutlichen, wie mit `key generate-asymmetric-pair rsa` ein RSA-Schlüsselpaar erstellt wird.

Example Beispiel: Ein RSA-Schlüsselpaar erstellen

```
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": "0x000000000000160010",
      "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": "0x00000000000160011",
    "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 Beispiel: Erstellen eines RSA-Schlüsselpaars mit optionalen Attributen

```

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": "0x00000000000280cc8",
      "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 Beispiel: Erstellen Sie ein RSA-Schlüsselpaar mit Quorumwerten

Beim Generieren eines Schlüssels mit Quorumsteuerelementen muss der Schlüssel einer Mindestanzahl von Benutzern zugeordnet werden, die dem größten Schlüsselquorumwert entspricht. Zu den assoziierten Benutzern gehören der Schlüsselinhaber und Crypto-Benutzer, mit denen der Schlüssel geteilt wird. Um die Mindestanzahl der Benutzer zu ermitteln, mit denen der Schlüssel geteilt werden soll, ermitteln Sie den größten Quorumwert zwischen dem Quorumwert für die Schlüsselnutzung und dem Quorumwert für die Schlüsselverwaltung und subtrahieren Sie 1, um den Schlüsselbesitzer zu berücksichtigen, der standardmäßig mit dem Schlüssel verknüpft ist. Verwenden Sie den Befehl, um den Schlüssel mit mehreren Benutzern zu teilen. [Einen Schlüssel mit CloudHSM CLI teilen](#)

```
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": "0x00000000000280cc8",
      "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
  }
}
}
}

```

Argumente

<CLUSTER_ID>

Die ID des Clusters, auf dem dieser Vorgang ausgeführt werden soll.

Erforderlich: Wenn mehrere Cluster [konfiguriert wurden](#).

<MODULUS_SIZE_BITS>

Gibt die Länge des Moduls in Bits an. Der minimale Wert beträgt 2048.

Erforderlich: Ja

<PRIVATE_KEY_ATTRIBUTES>

Gibt eine durch Leerzeichen getrennte Liste von Schlüsselattributen an, die für den generierten privaten RSA-Schlüssel festgelegt werden sollen, in der Form von KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE (z. B. sign=true)

Eine Liste der unterstützten Schlüsselattribute finden Sie unter [Schlüsselattribute für CloudHSM-CLI](#).

Erforderlich: Nein

<PRIVATE_LABEL>

Gibt eine benutzerdefinierte Bezeichnung für den private-key an. Die maximal zulässige Größe für Client SDK 5.11 und höher label beträgt 127 Zeichen. Das Client-SDK 5.10 und früher ist auf 126 Zeichen begrenzt.

Erforderlich: Ja

<PUBLIC_EXPONENT>

Gibt den öffentlichen Exponenten an. Bei diesem Wert muss es sich eine ungerade Zahl gleich oder größer als 65537 handeln.

Erforderlich: Ja

<PUBLIC_KEY_ATTRIBUTES>

Gibt eine durch Leerzeichen getrennte Liste von Schlüsselattributen an, die für den generierten öffentlichen RSA-Schlüssel festgelegt werden sollen, in der Form von KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE (z. B. `verify=true`)

Eine Liste der unterstützten Schlüsselattribute finden Sie unter [Schlüsselattribute für CloudHSM-CLI](#).

Erforderlich: Nein

<PUBLIC_LABEL>

Gibt eine benutzerdefinierte Bezeichnung für den öffentlichen Schlüssel an. Die maximal zulässige Größe für Client-SDK 5.11 und höher `label` beträgt 127 Zeichen. Das Client-SDK 5.10 und früher ist auf 126 Zeichen begrenzt.

Erforderlich: Ja

<SESSION>

Erstellt einen Schlüssel, der nur in der aktuellen Sitzung existiert. Der Schlüssel kann nach Ende der Sitzung nicht wiederhergestellt werden.

Verwenden Sie diesen Parameter, wenn Sie einen Schlüssel zum Packen nur für kurze Zeit benötigen, z. B. einen Schlüssel, der einen anderen Schlüssel verschlüsselt und dann schnell entschlüsselt. Verwenden Sie keinen Sitzungsschlüssel, um Daten zu verschlüsseln, die Sie nach dem Ende der Sitzung möglicherweise entschlüsseln müssen.

Standardmäßig handelt es sich bei den generierten Schlüsseln um persistente (Token-)Schlüssel. Das Übergeben von `<SESSION>` ändert dies und stellt sicher, dass es sich bei einem mit diesem Argument generierten Schlüssel um einen (kurzlebigen) Sitzungsschlüssel handelt.

Erforderlich: Nein

<SHARE_CRYPTO_USERS>

Gibt eine durch Leerzeichen getrennte Liste von Crypto-Benutzernamen an, mit denen der private RSA-Schlüssel gemeinsam genutzt werden soll

Erforderlich: Nein

<MANAGE_PRIVATE_KEY_QUORUM_VALUE>

Der Quorumwert für die Schlüsselverwaltungsvorgänge des privaten Schlüssels. Dieser Wert muss kleiner oder gleich der Anzahl der Benutzer sein, denen der Schlüssel zugeordnet ist. Dazu gehören Benutzer, mit denen der Schlüssel geteilt wird, und der Schlüsselbesitzer. Maximaler Wert von 8.

Erforderlich: Nein

<USE_PRIVATE_KEY_QUORUM_VALUE>

Der Quorumwert für die Schlüsselverwendungsvorgänge des privaten Schlüssels. Dieser Wert muss kleiner oder gleich der Anzahl der Benutzer sein, denen der Schlüssel zugeordnet ist. Dazu gehören Benutzer, mit denen der Schlüssel geteilt wird, und der Schlüsselbesitzer. Maximaler Wert von 8.

Erforderlich: Nein

Verwandte Themen

- [Schlüsselattribute für CloudHSM-CLI](#)
- [Schlüssel mithilfe der CloudHSM-CLI filtern](#)

Die Kategorie „Symmetrisch generieren“ in CloudHSM CLI

In der CloudHSM-CLI `key generate-symmetric` ist dies eine übergeordnete Kategorie für eine Gruppe von Befehlen, die in Kombination mit der übergeordneten Kategorie einen Befehl erstellen, der symmetrische Schlüssel generiert. Derzeit besteht diese Kategorie aus den folgenden Befehlen:

- [key generate-symmetric aes](#)
- [key generate-symmetric generic-secret](#)

Generieren Sie einen symmetrischen AES-Schlüssel mit CloudHSM CLI

Verwenden Sie den `key generate-symmetric aes` Befehl in der CloudHSM-CLI, um einen symmetrischen AES-Schlüssel in Ihrem Cluster zu generieren. AWS CloudHSM

Benutzertyp

Die folgenden Benutzertypen können diesen Befehl ausführen.

- Crypto-Benutzer () CUs

Voraussetzungen

Um diesen Befehl auszuführen, müssen Sie als CU angemeldet sein.

Syntax

```
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
```

Beispiele

Diese Beispiele zeigen, wie Sie den `key generate-symmetric aes`-Befehl zum Erstellen eines AES-Schlüssels verwenden.

Example Beispiel: Einen AES-Schlüssel erstellen

```
aws-cloudhsm > key generate-symmetric aes \
--label example-aes \
--key-length-bytes 24
{
  "error_code": 0,
  "data": {
    "key": {
      "key-reference": "0x000000000002e06bf",
      "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 Beispiel: Erstellen eines AES-Schlüssels mit optionalen Attributen

```

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": "0x000000000002e06bf",
      "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 Beispiel: Erstellen Sie einen AES-Schlüssel mit Quorumwerten

Beim Generieren eines Schlüssels mit Quorumsteuerungen muss der Schlüssel einer Mindestanzahl von Benutzern zugeordnet werden, die dem größten Schlüsselquorumwert entspricht. Zu den assoziierten Benutzern gehören der Schlüsselinhaber und Crypto-Benutzer, mit denen der Schlüssel geteilt wird. Um die Mindestanzahl der Benutzer zu ermitteln, mit denen der Schlüssel geteilt werden soll, ermitteln Sie den größten Quorumwert zwischen dem Quorumwert für die Schlüsselnutzung und dem Quorumwert für die Schlüsselverwaltung und subtrahieren Sie 1, um den Schlüsselbesitzer zu berücksichtigen, der standardmäßig mit dem Schlüssel verknüpft ist. Verwenden Sie den Befehl, um den Schlüssel mit mehreren Benutzern zu teilen. [Einen Schlüssel mit CloudHSM CLI teilen](#)

```
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": "0x000000000002e06bf",
      "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
  }
}
}
}

```

Argumente

<CLUSTER_ID>

Die ID des Clusters, auf dem dieser Vorgang ausgeführt werden soll.

Erforderlich: Wenn mehrere Cluster [konfiguriert wurden](#).

<KEY_ATTRIBUTES>

Gibt eine durch Leerzeichen getrennte Liste von Schlüsselattributen an, die für den generierten AES-Schlüssel festgelegt werden sollen, in der Form von KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE (z. B. sign=true).

Eine Liste der unterstützten Schlüsselattribute finden Sie unter [Schlüsselattribute für CloudHSM-CLI](#).

Erforderlich: Nein

<KEY-LENGTH-BYTES>

Gibt die Schlüssellänge in Byte an.

Zulässige Werte:

- 16, 24 und 32

Erforderlich: Ja

<LABEL>

Gibt eine benutzerdefinierte Bezeichnung für den AES-Schlüssel an. Die maximal zulässige Größe für Client SDK 5.11 und höher `label` beträgt 127 Zeichen. Das Client-SDK 5.10 und früher ist auf 126 Zeichen begrenzt.

Erforderlich: Ja

<SESSION>

Erstellt einen Schlüssel, der nur in der aktuellen Sitzung existiert. Der Schlüssel kann nach Ende der Sitzung nicht wiederhergestellt werden.

Verwenden Sie diesen Parameter, wenn Sie einen Schlüssel zum Packen nur für kurze Zeit benötigen, z. B. einen Schlüssel, der einen anderen Schlüssel verschlüsselt und dann schnell entschlüsselt. Verwenden Sie keinen Sitzungsschlüssel, um Daten zu verschlüsseln, die Sie nach dem Ende der Sitzung möglicherweise entschlüsseln müssen.

Standardmäßig handelt es sich bei den generierten Schlüsseln um persistente (Token-)Schlüssel. Das Übergeben von `<SESSION>` ändert dies und stellt sicher, dass es sich bei einem mit diesem Argument generierten Schlüssel um einen (kurzlebigen) Sitzungsschlüssel handelt.

Erforderlich: Nein

<SHARE_CRYPTO_USERS>

Gibt eine durch Leerzeichen getrennte Liste von Crypto-Benutzernamen an, mit denen der AES-Schlüssel gemeinsam genutzt werden soll

Erforderlich: Nein

<MANAGE_KEY_QUORUM_VALUE>

Der Quorumwert für Schlüsselverwaltungsvorgänge. Dieser Wert muss kleiner oder gleich der Anzahl der Benutzer sein, denen der Schlüssel zugeordnet ist. Dazu gehören Benutzer, mit denen der Schlüssel geteilt wird, und der Schlüsselbesitzer. Maximaler Wert von 8.

Erforderlich: Nein

<USE_KEY_QUORUM_VALUE>

Der Quorumwert für Operationen zur Verwendung von Schlüsseln. Dieser Wert muss kleiner oder gleich der Anzahl der Benutzer sein, denen der Schlüssel zugeordnet ist. Dazu gehören Benutzer, mit denen der Schlüssel geteilt wird, und der Schlüsselbesitzer. Maximaler Wert von 8.

Erforderlich: Nein

Verwandte Themen

- [Schlüsselattribute für CloudHSM-CLI](#)
- [Schlüssel mithilfe der CloudHSM-CLI filtern](#)

Generieren Sie einen symmetrischen Generic Secret Key mit CloudHSM CLI

Verwenden Sie den `key generate-symmetric generic-secret` Befehl in der CloudHSM-CLI, um einen symmetrischen Generic Secret-Schlüssel in Ihrem Cluster zu generieren. AWS CloudHSM

Benutzertyp

Die folgenden Benutzertypen können diesen Befehl ausführen.

- Crypto-Benutzer () CUs

Voraussetzungen

Um diesen Befehl auszuführen, müssen Sie als CU angemeldet sein.

Syntax

```
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

Beispiele

Diese Beispiele zeigen, wie Sie den `key generate-symmetric generic-secret`-Befehl verwenden, um einen generischen geheimen Schlüssel zu erstellen.

Example Beispiel: Erstellen Sie einen generischen geheimen Schlüssel

```
aws-cloudhsm > key generate-symmetric generic-secret \
--label example-generic-secret \
--key-length-bytes 256
{
  "error_code": 0,
  "data": {
    "key": {
      "key-reference": "0x000000000002e08fd",
      "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 Beispiel: Erstellen Sie einen generischen geheimen Schlüssel mit optionalen Attributen

```

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": "0x000000000002e08fd",
      "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 Beispiel: Erstellen Sie einen generischen geheimen Schlüssel mit Quorumwerten

Beim Generieren eines Schlüssels mit Quorumsteuerelementen muss der Schlüssel einer Mindestanzahl von Benutzern zugeordnet werden, die dem größten Schlüsselquorumwert entspricht. Zu den assoziierten Benutzern gehören der Schlüsselinhaber und Crypto-Benutzer, mit denen der Schlüssel geteilt wird. Um die Mindestanzahl der Benutzer zu ermitteln, mit denen der Schlüssel geteilt werden soll, ermitteln Sie den größten Quorumwert zwischen dem Quorumwert für die Schlüsselnutzung und dem Quorumwert für die Schlüsselverwaltung und subtrahieren Sie 1, um den Schlüsselbesitzer zu berücksichtigen, der standardmäßig mit dem Schlüssel verknüpft ist. Verwenden Sie den Befehl, um den Schlüssel mit mehreren Benutzern zu teilen. [Einen Schlüssel mit CloudHSM CLI teilen](#)

```

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": "0x000000000002e08fd",
      "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
}
}
}
}

```

Argumente

<CLUSTER_ID>

Die ID des Clusters, auf dem dieser Vorgang ausgeführt werden soll.

Erforderlich: Wenn mehrere Cluster [konfiguriert wurden](#).

<KEY_ATTRIBUTES>

Gibt eine durch Leerzeichen getrennte Liste von Schlüsselattributen an, die für den generierten AES-Schlüssel festgelegt werden sollen, in der Form von KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE (z. B. sign=true).

Eine Liste der unterstützten Schlüsselattribute finden Sie unter [Schlüsselattribute für CloudHSM-CLI](#).

Erforderlich: Nein

<KEY-LENGTH-BYTES>

Gibt die Schlüssellänge in Byte an.

Zulässige Werte:

- 1 bis 800

Erforderlich: Ja

<LABEL>

Gibt eine benutzerdefinierte Bezeichnung für den generischen geheimen Schlüssel an. Die maximal zulässige Größe für Client SDK 5.11 und höher label beträgt 127 Zeichen. Das Client-SDK 5.10 und früher ist auf 126 Zeichen begrenzt.

Erforderlich: Ja

<SESSION>

Erstellt einen Schlüssel, der nur in der aktuellen Sitzung existiert. Der Schlüssel kann nach Ende der Sitzung nicht wiederhergestellt werden.

Verwenden Sie diesen Parameter, wenn Sie einen Schlüssel zum Packen nur für kurze Zeit benötigen, z. B. einen Schlüssel, der einen anderen Schlüssel verschlüsselt und dann schnell entschlüsselt. Verwenden Sie keinen Sitzungsschlüssel, um Daten zu verschlüsseln, die Sie nach dem Ende der Sitzung möglicherweise entschlüsseln müssen.

Standardmäßig handelt es sich bei den generierten Schlüsseln um persistente (Token-)Schlüssel. Das Übergeben von <SESSION> ändert dies und stellt sicher, dass es sich bei einem mit diesem Argument generierten Schlüssel um einen (kurzlebigen) Sitzungsschlüssel handelt.

Erforderlich: Nein

<SHARE_CRYPTO_USERS>

Durch Leerzeichen getrennte Liste von Crypto-Benutzernamen, mit denen der generische geheime Schlüssel geteilt werden soll

Erforderlich: Nein

<MANAGE_KEY_QUORUM_VALUE>

Der Quorumwert für Schlüsselverwaltungsvorgänge. Dieser Wert muss kleiner oder gleich der Anzahl der Benutzer sein, denen der Schlüssel zugeordnet ist. Dazu gehören Benutzer, mit denen der Schlüssel geteilt wird, und der Schlüsselbesitzer. Maximaler Wert von 8.

Erforderlich: Nein

<USE_KEY_QUORUM_VALUE>

Der Quorumwert für Operationen zur Verwendung von Schlüsseln. Dieser Wert muss kleiner oder gleich der Anzahl der Benutzer sein, denen der Schlüssel zugeordnet ist. Dazu gehören Benutzer, mit denen der Schlüssel geteilt wird, und der Schlüsselbesitzer. Maximaler Wert von 8.

Erforderlich: Nein

Verwandte Themen

- [Schlüsselattribute für CloudHSM-CLI](#)
- [Schlüssel mithilfe der CloudHSM-CLI filtern](#)

Importieren Sie einen Schlüssel im PEM-Format mit der CloudHSM-CLI

Verwenden Sie den `key import pem` Befehl in AWS CloudHSM, um einen Schlüssel im PEM-Format in ein Hardware-Sicherheitsmodul (HSM) zu importieren. Sie können ihn verwenden, um öffentliche und außerhalb des HSM erstellte Schlüssel zu importieren.

Note

Verwenden Sie den [Exportieren Sie einen asymmetrischen Schlüssel mit CloudHSM CLI](#) Befehl, um eine Standard-PEM-Datei aus einem öffentlichen Schlüssel oder eine Referenz-PEM-Datei aus einem privaten Schlüssel zu erstellen.

Benutzertyp

Die folgenden Benutzertypen können diesen Befehl ausführen.

- Crypto-Benutzer () CUs

Voraussetzungen

- Um diesen Befehl auszuführen, müssen Sie als CU angemeldet sein.

Syntax

```
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](#)

Beispiele

Dieses Beispiel zeigt, wie der `key import pem` Befehl verwendet wird, um einen öffentlichen RSA-Schlüssel aus einer Datei im PEM-Format zu importieren.

Example Beispiel: Importieren Sie einen öffentlichen RSA-Schlüssel

```
aws-cloudhsm > key import pem --path /home/example --label example-imported-key --key-  
type-class rsa-public  
{  
  "error_code": 0,  
  "data": {  
    "key": {  
      "key-reference": "0x000000000001e08e3",  
      "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":
"0x8e9c172c37aa22ed1ce25f7c3a7c936dadcd532201400128b044ebb4b96#..3e4930ab910df5a2896eae8853cfe
    "modulus-size-bits": 2048
  }
},
  "message": "Successfully imported key"
}
}

```

Example Beispiel: Importieren Sie einen öffentlichen RSA-Schlüssel mit optionalen Attributen

```

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": "0x000000000001e08e3",
      "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":
      "0x8e9c172c37aa22ed1ce25f7c3a7c936dad532201400128b044ebb4b96#..3e4930ab910df5a2896eae8853cfe"
    "modulus-size-bits": 2048
  }
},
"message": "Successfully imported key"
}
}

```

Argumente

<CLUSTER_ID>

Die ID des Clusters, auf dem dieser Vorgang ausgeführt werden soll.

Erforderlich: Wenn mehrere Cluster [konfiguriert wurden](#).

<PATH>

Gibt den Dateipfad an, in dem sich die Schlüsseldatei befindet.

Erforderlich: Ja

<LABEL>

Gibt eine benutzerdefinierte Bezeichnung für den importierten Schlüssel an. Die maximal zulässige Größe für label beträgt 126 Zeichen.

Erforderlich: Ja

<KEY_TYPE_CLASS>

Schlüsseltyp und Klasse des umschlossenen Schlüssels.

Mögliche Werte:

- ec-public
- rsa-öffentlich

Erforderlich: Ja

<IMPORT_KEY_ATTRIBUTES>

Gibt eine durch Leerzeichen getrennte Liste von Schlüsselattributen an, die für den importierten Schlüssel festgelegt werden sollen, in der Form von KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE (z. B. sign=true). Eine Liste der unterstützten Schlüsselattribute finden Sie unter [Schlüsselattribute für CloudHSM-CLI](#).

Erforderlich: Nein

Verwandte Themen

- [Die Kategorie Kryptozeichen in CloudHSM CLI](#)
- [Die Kategorie Crypto Verify in CloudHSM CLI](#)

Schlüssel für einen Benutzer mit CloudHSM CLI auflisten

Verwenden Sie den key list Befehl in der CloudHSM-CLI, um alle Schlüssel für den aktuellen Benutzer in Ihrem AWS CloudHSM Cluster zu finden. Die Ausgabe umfasst Schlüssel, die der Benutzer besitzt und die für ihn freigegeben sind, sowie alle öffentlichen Schlüssel im CloudHSM-Cluster.

Benutzertyp

Die folgenden Benutzertypen können diesen Befehl ausführen.

- Administratoren () COs
- Krypto-Benutzer () CUs

Syntax

```
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

Beispiele

Die folgenden Beispiele zeigen die verschiedenen Möglichkeiten, wie Sie den key list-Befehl ausführen. Die folgenden Beispiele zeigen die Ausgaben eines Krypto-Benutzers.

Example Beispiel: Finde alle Schlüssel — Standard

Dieser Befehl listet die Schlüssel des angemeldeten Benutzers auf, der sich im AWS CloudHSM Cluster befindet.

Note

Standardmäßig werden nur 10 Schlüssel des aktuell angemeldeten Benutzers angezeigt, und nur die Schlüssel `key-reference` und `label` werden als Ausgabe angezeigt. Verwenden Sie die entsprechenden Paginierungsoptionen, um mehr oder weniger Schlüssel als Ausgabe anzuzeigen.

```
aws-cloudhsm > key list
{
  "error_code": 0,
  "data": {
    "matched_keys": [
      {
        "key-reference": "0x000000000000003d5",
        "attributes": {
          "label": "test_label_1"
        }
      },
      {
        "key-reference": "0x00000000000000626",
        "attributes": {
          "label": "test_label_2"
        }
      },
      ...8 keys later...
    ],
    "total_key_count": 56,
    "returned_key_count": 10,
    "next_token": "10"
  }
}
```

Example Beispiel: Finde alle Schlüssel — ausführlich

Die Ausgabe enthält Schlüssel, die der Benutzer besitzt und mit anderen teilt, sowie alle öffentlichen Schlüssel in der HSMs.

 Note

Hinweis: Standardmäßig werden nur 10 Schlüssel des aktuell angemeldeten Benutzers angezeigt. Verwenden Sie die entsprechenden Paginierungsoptionen, um mehr oder weniger Schlüssel als Ausgabe anzuzeigen.

```
aws-cloudhsm > key list --verbose
{
  "error_code": 0,
  "data": {
    "matched_keys": [
      {
        "key-reference": "0x00000000000012000c",
        "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": "0x0000000000012000d",
  "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 Beispiel: Paginated return

Das folgende Beispiel zeigt eine paginierte Teilmenge der Schlüssel, die nur zwei Schlüssel enthält. Das Beispiel bietet dann einen nachfolgenden Aufruf, um die nächsten beiden Schlüssel anzuzeigen.

```

aws-cloudhsm > key list --verbose --max-items 2
{
  "error_code": 0,
  "data": {
    "matched_keys": [
      {
        "key-reference": "0x000000000000000030",
        "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": "0x00000000000000042",
    "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"
}
}

```

Um die nächsten 2 Schlüssel anzuzeigen, kann ein nachfolgender Aufruf getätigt werden:

```
aws-cloudhsm > key list --verbose --max-items 2 --starting-token 2
{
  "error_code": 0,
  "data": {
    "matched_keys": [
      {
        "key-reference": "0x00000000000000081",
        "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": "0x000000000000000089",
    "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"
}
}

```

Weitere Beispiele, die zeigen, wie der wichtige Filtrationsmechanismus in der CloudHSM-CLI funktioniert, finden Sie unter [Schlüssel mithilfe der CloudHSM-CLI filtern](#).

Argumente

<CLUSTER_ID>

Die ID des Clusters, auf dem dieser Vorgang ausgeführt werden soll.

Erforderlich: Wenn mehrere Cluster [konfiguriert wurden](#).

<FILTER>

Schlüsselreferenz (z. B. `key-reference=0xabc`) oder durch Leerzeichen getrennte Liste von Schlüsselattributen in der Form `tr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE`, dass übereinstimmende Schlüssel für die Liste ausgewählt werden sollen.

Eine Liste der unterstützten CloudHSM-CLI-Schlüsselattribute finden Sie unter [Schlüsselattribute für CloudHSM-CLI](#).

Erforderlich: Nein

<MAX_ITEMS>

Die Gesamtzahl der Elemente, die in der Ausgabe des Befehls zurückgegeben werden sollen. Ist die Gesamtzahl der verfügbaren Elemente größer als der angegebene Wert, wird ein next-token in der Ausgabe des Befehls bereitgestellt. Um die Seitennummerierung fortzusetzen, geben Sie den next-token-Wert im starting-token-Argument eines nachfolgenden Befehls an.

Erforderlich: Nein

<STARTING_TOKEN>

Ein Token für den Beginn der Seitennummerierung. Dies ist das next-token aus einer zuvor abgeschnittenen Antwort.

Erforderlich: Nein

<VERBOSE>

Falls enthalten, werden alle Attribute und Schlüsselinformationen für jeden übereinstimmenden Schlüssel gedruckt. Standardmäßig zeigt jeder übereinstimmende Schlüssel nur seine Schlüsselreferenz und sein Labelattribut an. Wenn diese Markierung von Admins verwendet wird, hat sie keine Wirkung.

Erforderlich: Nein

Verwandte Themen

- [Löschen Sie einen Schlüssel mit CloudHSM CLI](#)
- [Exportieren Sie einen asymmetrischen Schlüssel mit CloudHSM CLI](#)
- [Freigabe eines Schlüssels mithilfe der CloudHSM-CLI rückgängig machen](#)
- [Schlüsselattribute für CloudHSM-CLI](#)
- [Schlüssel mithilfe der CloudHSM-CLI filtern](#)

Einen Schlüssel mit CloudHSM CLI replizieren

Verwenden Sie den key replicate Befehl in der CloudHSM-CLI, um einen Schlüssel von einem AWS CloudHSM Quellcluster auf einen Zielcluster zu replizieren. AWS CloudHSM

Benutzertyp

Die folgenden Benutzertypen können diesen Befehl ausführen.

- Administratoren () COs
- Krypto-Benutzer () CUs

Note

Crypto-Benutzer müssen den Schlüssel besitzen, um diesen Befehl verwenden zu können.

Voraussetzungen

- Die Quell- und Zielcluster müssen Klone sein. Das bedeutet, dass einer aus einem Backup des anderen erstellt wurde oder dass beide aus einem gemeinsamen Backup erstellt wurden. Weitere Informationen finden Sie unter [Cluster aus Sicherungen erstellen](#).
- Der Besitzer des Schlüssels muss auf dem Zielcluster vorhanden sein. Wenn der Schlüssel mit anderen Benutzern geteilt wird, müssen diese Benutzer außerdem auch auf dem Zielcluster vorhanden sein.
- Um diesen Befehl ausführen zu können, müssen Sie sowohl im Quell- als auch im Zielcluster als Crypto-Benutzer oder Administrator angemeldet sein.
- Im Einzelbefehlsmodus verwendet der Befehl die Umgebungsvariablen CLOUDHSM_PIN und CLOUDHSM_ROLE, um sich im Quellcluster zu authentifizieren. Weitere Informationen finden Sie unter [Einzelbefehlsmodus](#). Um Anmeldeinformationen für den Zielcluster bereitzustellen, müssen Sie zwei zusätzliche Umgebungsvariablen festlegen: DESTINATION_CLOUDHSM_PIN und DESTINATION_CLOUDHSM_ROLE:

```
$ export DESTINATION_CLOUDHSM_ROLE=<role>
```

```
$ export DESTINATION_CLOUDHSM_PIN=<username:password>
```

- Im interaktiven Modus müssen sich Benutzer explizit sowohl beim Quell- als auch beim Zielcluster anmelden.

Syntax

```
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

```

Beispiele

Example Beispiel: Schlüssel replizieren

Dieser Befehl repliziert einen Schlüssel von einem Quellcluster auf einen geklonten Zielcluster. Das folgende Beispiel zeigt die Ausgabe, wenn Sie als Crypto-Benutzer auf beiden Clustern angemeldet sind.

```

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": "0x000000000003000006",
      "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"
}
}

```

Argumente

<FILTER>

Schlüsselreferenz (z. B. `key-reference=0xabc`) oder durch Leerzeichen getrennte Liste von Schlüsselattributen in der Form `attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE` zur Auswahl eines passenden Schlüssels auf dem Quellcluster.

Eine Liste der unterstützten CloudHSM-CLI-Schlüsselattribute finden Sie unter [Schlüsselattribute für CloudHSM-CLI](#).

Erforderlich: Ja

<SOURCE_CLUSTER_ID>

Die Quellcluster-ID.

Erforderlich: Ja

<DESTINATION_CLUSTER_ID>

Die Zielcluster-ID.

Erforderlich: Ja

Verwandte Themen

- [Verbindung zu mehreren Clustern mit CloudHSM CLI herstellen](#)

Legen Sie die Attribute von Schlüsseln mit der CloudHSM-CLI fest

Verwenden Sie den `key set-attribute` Befehl in der CloudHSM-CLI, um die Attribute der Schlüssel in Ihrem AWS CloudHSM Cluster festzulegen. Nur der CU, der den Schlüssel erstellt hat und folglich Eigentümer des Schlüssels ist, kann die Attribute des Schlüssels ändern.

Eine Liste der wichtigsten Attribute, die in der CloudHSM-CLI verwendet werden können, finden Sie unter [Schlüsselattribute für CloudHSM-CLI](#).

Benutzertyp

Die folgenden Benutzertypen können diesen Befehl ausführen.

- Crypto-Benutzer (CUs) können diesen Befehl ausführen.
- Administratoren können das vertrauenswürdige Attribut festlegen.

Voraussetzungen

Um diesen Befehl auszuführen, müssen Sie als CU angemeldet sein. Um das vertrauenswürdige Attribut festzulegen, müssen Sie als Admin-Benutzer angemeldet sein.

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

```

Beispiel: Einstellung eines identifizierenden Attributs

Das folgende Beispiel demonstriert, wie Sie mit dem key set-attribute-Befehl das Label setzen.

Example

1. Verwenden Sie den Schlüssel mit dem Label my_key, wie hier gezeigt:

```

aws-cloudhsm > key set-attribute --filter attr.label=my_key --name encrypt --value
false
{
  "error_code": 0,
  "data": {
    "message": "Attribute set successfully"
  }
}

```

2. Bestätigen Sie mit dem key list-Befehl, dass sich das encrypt-Attribut geändert hat:

```

aws-cloudhsm > key list --filter attr.label=my_key --verbose
{
  "error_code": 0,
  "data": {
    "matched_keys": [
      {
        "key-reference": "0x000000000006400ec",
        "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
}
}

```

Argumente

<CLUSTER_ID>

Die ID des Clusters, auf dem dieser Vorgang ausgeführt werden soll.

Erforderlich: Wenn mehrere Cluster [konfiguriert wurden](#).

<KEY_ATTRIBUTE>

Gibt den Namen des Attributs des Schlüssels an.

Erforderlich: Ja

<FILTER>

Schlüsselreferenz (z. B. `key-reference=0xabc`) oder durch Leerzeichen getrennte Liste von Schlüsselattributen in der Form `tr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE`, dass ein passender Schlüssel zum Löschen ausgewählt werden soll.

Eine Liste der unterstützten CloudHSM-CLI-Schlüsselattribute finden Sie unter [Schlüsselattribute für CloudHSM-CLI](#).

Erforderlich: Nein

<KEY_ATTRIBUTE_VALUE>

Gibt den Wert des Attributs des Schlüssels an.

Erforderlich: Ja

<KEY_REFERENCE>

Eine hexadezimale oder dezimale Darstellung des Schlüssels. (z. B. ein Tastenkürzel).

Erforderlich: Nein

<APPROVAL>

Gibt den Dateipfad zu einer signierten Quorum-Token-Datei an, um den Vorgang zu genehmigen. Nur erforderlich, wenn der Quorumwert des Schlüssels für den Schlüsselverwaltungsdienst größer als 1 ist.

Verwandte Themen

- [Schlüssel mithilfe der CloudHSM-CLI filtern](#)

- [Schlüsselattribute für CloudHSM-CLI](#)

Einen Schlüssel mit CloudHSM CLI teilen

Verwenden Sie den `key share` Befehl in der CloudHSM-CLI, um einen Schlüssel mit anderen CUs in Ihrem AWS CloudHSM Cluster zu teilen.

Nur der CU, der den Schlüssel erstellt hat und somit Eigentümer des Schlüssels ist, kann den Schlüssel teilen. Benutzer, mit denen ein Schlüssel geteilt wird, können den Schlüssel für kryptografische Operationen verwenden, aber sie können den Schlüssel nicht löschen, exportieren, teilen oder das Teilen aufheben. Darüber hinaus können diese Benutzer [Schlüsselattribute](#) nicht ändern.

Benutzertyp

Die folgenden Benutzertypen können diesen Befehl ausführen.

- Crypto-Benutzer () CUs

Voraussetzungen

Um diesen Befehl auszuführen, müssen Sie als CU angemeldet sein.

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')

Beispiel: Teilen Sie sich einen Schlüssel mit einem anderen CU

Das folgende Beispiel zeigt, wie Sie den key share-Befehl verwenden, um einen Schlüssel mit CU alice zu teilen.

Example

1. Führen Sie den key share-Befehl aus, um den Schlüssel mit alice zu teilen.

```
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. Führen Sie den Befehl key list aus.

```
aws-cloudhsm > key list --filter attr.label="rsa_key_to_share" attr.class=private-
key --verbose
{
  "error_code": 0,
  "data": {
    "matched_keys": [
      {
        "key-reference": "0x000000000001c0686",
```

```
"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
}
}

```

- Überprüfen Sie in der obigen Liste, ob `alice` in der Liste von `shared-users` ist.

Argumente

<CLUSTER_ID>

Die ID des Clusters, auf dem dieser Vorgang ausgeführt werden soll.

Erforderlich: Wenn mehrere Cluster [konfiguriert wurden](#).

<FILTER>

Schlüsselreferenz (z. B. `key-reference=0xabc`) oder durch Leerzeichen getrennte Liste von Schlüsselattributen in der Form `tr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE`, dass ein passender Schlüssel zum Löschen ausgewählt werden soll.

Eine Liste der unterstützten Schlüsselattribute finden Sie unter [Schlüsselattribute für CloudHSM-CLI](#).

Erforderlich: Ja

<USERNAME>

Gibt einen Anzeigenamen für den Benutzer an. Die maximale Länge beträgt 31 Zeichen. Das einzige zulässige Sonderzeichen ist ein Unterstrich (`_`). Beim Benutzernamen wird in diesem Befehl nicht zwischen Groß- und Kleinschreibung unterschieden, der Benutzername wird immer in Kleinbuchstaben angezeigt.

Erforderlich: Ja

<ROLE>

Gibt die diesem Benutzer zugewiesene Rolle an. Dieser Parameter muss angegeben werden. Um die Rolle des Benutzers zu erfahren, verwenden Sie den Befehl `user list`. Ausführliche Informationen zu den Benutzertypen in einem HSM finden Sie unter [HSM-Benutzertypen für CloudHSM CLI](#).

Erforderlich: Ja

<APPROVAL>

Gibt den Dateipfad zu einer signierten Quorum-Token-Datei an, um den Vorgang zu genehmigen. Nur erforderlich, wenn der Quorumwert des Schlüssels für den Schlüsselverwaltungsdienst größer als 1 ist.

Verwandte Themen

- [Schlüssel mithilfe der CloudHSM-CLI filtern](#)
- [Schlüsselattribute für CloudHSM-CLI](#)

Freigabe eines Schlüssels mithilfe der CloudHSM-CLI rückgängig machen

Verwenden Sie den `key unshare` Befehl in der CloudHSM-CLI, um die gemeinsame Nutzung eines Schlüssels mit anderen CUs in Ihrem Cluster aufzuheben. AWS CloudHSM

Nur der CU, der den Schlüssel erstellt hat und somit Eigentümer des Schlüssels ist, kann das Teilen des Schlüssels aufheben. Benutzer, mit denen ein Schlüssel geteilt wird, können den Schlüssel für kryptografische Operationen verwenden, aber sie können den Schlüssel nicht löschen, exportieren, teilen oder das Teilen aufheben. Darüber hinaus können diese Benutzer [Schlüsselattribute](#) nicht ändern.

Benutzertyp

Die folgenden Benutzertypen können diesen Befehl ausführen.

- Crypto-Benutzer () CUs

Voraussetzungen

Um diesen Befehl auszuführen, müssen Sie als CU angemeldet sein.

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')

Beispiel: Die gemeinsame Nutzung eines Schlüssels mit einer anderen CU rückgängig machen

Das folgende Beispiel zeigt, wie Sie den key unshare-Befehl verwenden, um einen Schlüssel mit CU alice nicht mehr zu teilen.

Example

1. Führen Sie den key list-Befehl aus und filtern Sie nach dem bestimmten Schlüssel, mit dem Sie das Teilen mit alice beenden möchten.

```
aws-cloudhsm > key list --filter attr.label="rsa_key_to_share" attr.class=private-
key --verbose
{
  "error_code": 0,
  "data": {
    "matched_keys": [
      {
        "key-reference": "0x000000000001c0686",
        "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
}
}

```

- Bestätigen Sie, dass `alice` in der `shared-users`-Ausgabe enthalten ist, und führen Sie den folgenden `key unshare`-Befehl aus, um das Teilen des Schlüssels mit `alice` aufzuheben.

```

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"
  }
}

```

- Führen Sie den `key list`-Befehl erneut aus, um zu bestätigen, dass das Teilen des Schlüssels mit `alice` aufgehoben wurde.

```

aws-cloudhsm > key list --filter attr.label="rsa_key_to_share" attr.class=private-
key --verbose
{
  "error_code": 0,
  "data": {

```

```
"matched_keys": [
  {
    "key-reference": "0x000000000001c0686",
    "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
}
}

```

Argumente

<CLUSTER_ID>

Die ID des Clusters, auf dem dieser Vorgang ausgeführt werden soll.

Erforderlich: Wenn mehrere Cluster [konfiguriert wurden](#).

<FILTER>

Schlüsselreferenz (z. B. `key-reference=0xabc`) oder durch Leerzeichen getrennte Liste von Schlüsselattributen in der Form `tr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE`, dass ein passender Schlüssel zum Löschen ausgewählt werden soll.

Eine Liste der unterstützten Schlüsselattribute finden Sie unter [Schlüsselattribute für CloudHSM-CLI](#).

Erforderlich: Ja

<USERNAME>

Gibt einen Anzeigenamen für den Benutzer an. Die maximale Länge beträgt 31 Zeichen. Das einzige zulässige Sonderzeichen ist ein Unterstrich (`_`). Beim Benutzernamen wird in diesem Befehl nicht zwischen Groß- und Kleinschreibung unterschieden, der Benutzername wird immer in Kleinbuchstaben angezeigt.

Erforderlich: Ja

<ROLE>

Gibt die diesem Benutzer zugewiesene Rolle an. Dieser Parameter muss angegeben werden. Um die Rolle des Benutzers zu erfahren, verwenden Sie den Befehl `user list`. Ausführliche Informationen zu den Benutzertypen in einem HSM finden Sie unter [HSM-Benutzertypen für CloudHSM CLI](#).

Erforderlich: Ja

<APPROVAL>

Gibt den Dateipfad zu einer signierten Quorum-Token-Datei an, um den Vorgang zu genehmigen. Nur erforderlich, wenn der Quorumwert des Schlüssels für den Schlüsselverwaltungsdienst größer als 1 ist.

Verwandte Themen

- [Schlüssel mithilfe der CloudHSM-CLI filtern](#)
- [Schlüsselattribute für CloudHSM-CLI](#)

Der Befehl zum Entpacken von Schlüsseln in der CloudHSM-CLI

Der `key unwrap` übergeordnete Befehl in der CloudHSM-CLI importiert einen verschlüsselten (verpackten) symmetrischen oder asymmetrischen privaten Schlüssel aus einer Datei in das HSM. Dieser Befehl dient zum Import verschlüsselter Schlüssel, die durch den [Der Key Wrap-Befehl in der CloudHSM-CLI](#) Befehl umschlossen wurden. Er kann aber auch zum Entpacken von Schlüsseln verwendet werden, die mit anderen Tools umschlossen wurden. In diesen Situationen empfehlen wir jedoch, die Softwarebibliotheken PKCS#11 oder JCE zu verwenden, um den Schlüssel zu entpacken.

- [aes-gcm](#)
- [aes-no-pad](#)
- [aes-pkcs5-pad](#)
- [aes-zero-pad](#)
- [cloudhsm-aes-gcm](#)
- [rsa-aes](#)
- [rsa-oaep](#)
- [rsa-pkcs](#)

Entpacken Sie einen Schlüssel mit AES-GCM mithilfe der CloudHSM-CLI

Verwenden Sie den `key unwrap aes-gcm` Befehl in der CloudHSM-CLI, um einen Payload-Schlüssel mithilfe des AES-Wrapping-Schlüssels und des Entpackungsmechanismus in den Cluster zu entpacken. AES-GCM

Unverpackte Schlüssel können auf die gleiche Weise verwendet werden wie die Schlüssel, die von generiert wurden. AWS CloudHSM Um anzuzeigen, dass sie nicht lokal generiert wurden, ist ihr `local` Attribut auf `false` gesetzt.

Um den `key unwrap aes-gcm` Befehl verwenden zu können, müssen Sie den AES-Wrapping-Schlüssel in Ihrem AWS CloudHSM Cluster haben und sein `unwrap` Attribut muss auf `true` gesetzt sein.

Benutzertyp

Die folgenden Benutzertypen können diesen Befehl ausführen.

- Crypto-Benutzer (CUs)

Voraussetzungen

- Um diesen Befehl auszuführen, müssen Sie als CU angemeldet sein.

Syntax

```
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

Beispiele

Diese Beispiele zeigen, wie der `key unwrap aes-gcm` Befehl mithilfe eines AES-Schlüssels verwendet wird, dessen `unwrap` Attributwert auf `gesetzt` ist `true`.

Example Beispiel: Entpacken Sie einen Payload-Schlüssel aus Base64-kodierten verpackten Schlüsseldaten

```
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 xvslgrtg8kHrzrvekny97tLSieokpPwV8
```

```
{
  "error_code": 0,
  "data": {
    "key": {
      "key-reference": "0x0000000000001808e4",
      "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 Beispiel: Entpacken Sie einen Payload-Schlüssel, der über einen Datenpfad bereitgestellt wird

```

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": "0x000000000001808e4",
      "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
}
}
}
}

```

Argumente

<CLUSTER_ID>

Die ID des Clusters, auf dem dieser Vorgang ausgeführt werden soll.

Erforderlich: Wenn mehrere Cluster [konfiguriert wurden](#).

<FILTER>

Schlüsselreferenz (z. B. `key-reference=0xabc`) oder durch Leerzeichen getrennte Liste von Schlüsselattributen in der Form `tr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE`, dass ein Schlüssel zum Entpacken ausgewählt werden soll.

Erforderlich: Ja

<DATA_PATH>

Pfad zur Binärdatei, die die umschlossenen Schlüsseldaten enthält.

Erforderlich: Ja (sofern nicht über Base64-kodierte Daten bereitgestellt)

<DATA>

Base64-kodierte umhüllte Schlüsseldaten.

Erforderlich: Ja (sofern nicht über den Datenpfad angegeben)

<ATTRIBUTES>

Durch Leerzeichen getrennte Liste von Schlüsselattributen in der Form von `KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE` für den umschlossenen Schlüssel.

Erforderlich: Nein

<AAD>

AES GCM-Wert für zusätzliche authentifizierte Daten (AAD) in Hexadezimalzahl.

Erforderlich: Nein

<TAG_LENGTH_BITS>

Länge des Aes GCM-Tags in Bit.

Erforderlich: Ja

<KEY_TYPE_CLASS>

Schlüsseltyp und Klasse des umschlossenen Schlüssels [mögliche Werte: `aes`, `des3`, `ec-private`, `generic-secret`, `rsa-private`].

Erforderlich: Ja

<LABEL>

Bezeichnung für den entpackten Schlüssel.

Erforderlich: Ja

<SESSION>

Erstellt einen Sitzungsschlüssel, der nur in der aktuellen Sitzung existiert. Der Schlüssel kann nach Ende der Sitzung nicht wiederhergestellt werden.

Erforderlich: Nein

<IV>

Anfangswert, der zum Umschließen des Schlüssels verwendet wird, in Hexadezimalform.

Erforderlich: Nein

<APPROVAL>

Gibt den Dateipfad zu einer signierten Quorum-Token-Datei an, um den Vorgang zu genehmigen. Nur erforderlich, wenn der Quorumwert für den Schlüsselverwaltungsdienst des Entpackungsschlüssels größer als 1 ist.

Verwandte Themen

- [Der Key Wrap-Befehl in der CloudHSM-CLI](#)
- [Der Befehl zum Entpacken von Schlüsseln in der CloudHSM-CLI](#)

Entpacken Sie einen Schlüssel AES-NO-PAD mithilfe der CloudHSM-CLI

Verwenden Sie den `key unwrap aes-no-pad` Befehl in der CloudHSM-CLI, um einen Payload-Schlüssel mithilfe des AES-Wrapping-Schlüssels und des Entpackungsmechanismus in den AWS CloudHSM Cluster zu entpacken. AES-NO-PAD

Unverpackte Schlüssel können auf die gleiche Weise verwendet werden wie die Schlüssel, die von generiert wurden. AWS CloudHSM Um anzuzeigen, dass sie nicht lokal generiert wurden, ist ihr `local` Attribut auf `false` gesetzt.

Um den `key unwrap aes-no-pad` Befehl verwenden zu können, müssen Sie den AES-Wrapping-Schlüssel in Ihrem AWS CloudHSM Cluster haben und sein `unwrap` Attribut muss auf `gesetzt` sein.

Benutzertyp

Die folgenden Benutzertypen können diesen Befehl ausführen.

- Crypto-Benutzer (CUs)

Voraussetzungen

- Um diesen Befehl auszuführen, müssen Sie als CU angemeldet sein.

Syntax

```
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

```

Beispiele

Diese Beispiele zeigen, wie der `key unwrap aes-no-pad` Befehl mithilfe eines AES-Schlüssels verwendet wird, dessen `unwrap` Attributwert auf `gesetzt ist true`.

Example Beispiel: Entpacken Sie einen Payload-Schlüssel aus Base64-kodierten umschlossenen Schlüsseldaten

```

aws-cloudhsm > key unwrap aes-no-pad --key-type-class aes --label aes-unwrapped --
filter attr.label=aes-example --data eXK3PMA0nKM9y3YX6brbhtMoC060E0H9
{
  "error_code": 0,
  "data": {
    "key": {
      "key-reference": "0x000000000001c08ec",
      "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 Beispiel: Entpacken Sie einen Payload-Schlüssel, der über einen Datenpfad bereitgestellt wird

```

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": "0x000000000001c08ec",
      "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
}
}
}
}

```

Argumente

<CLUSTER_ID>

Die ID des Clusters, auf dem dieser Vorgang ausgeführt werden soll.

Erforderlich: Wenn mehrere Cluster [konfiguriert wurden](#).

<FILTER>

Schlüsselreferenz (z. B. `key-reference=0xabc`) oder durch Leerzeichen getrennte Liste von Schlüsselattributen in der Form `tr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE`, dass ein Schlüssel zum Entpacken ausgewählt werden soll.

Erforderlich: Ja

<DATA_PATH>

Pfad zur Binärdatei, die die umschlossenen Schlüsseldaten enthält.

Erforderlich: Ja (sofern nicht über Base64-kodierte Daten bereitgestellt)

<DATA>

Base64-kodierte umhüllte Schlüsseldaten.

Erforderlich: Ja (sofern nicht über den Datenpfad angegeben)

<ATTRIBUTES>

Durch Leerzeichen getrennte Liste von Schlüsselattributen in der Form von `KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE` für den umschlossenen Schlüssel.

Erforderlich: Nein

<KEY_TYPE_CLASS>

Schlüsseltyp und Klasse des umschlossenen Schlüssels [mögliche Werte: `aesdes3`, `ec-private`, `generic-secret`, `rsa-private`].

Erforderlich: Ja

<LABEL>

Bezeichnung für den entpackten Schlüssel.

Erforderlich: Ja

<SESSION>

Erstellt einen Sitzungsschlüssel, der nur in der aktuellen Sitzung existiert. Der Schlüssel kann nach Ende der Sitzung nicht wiederhergestellt werden.

Erforderlich: Nein

<APPROVAL>

Gibt den Dateipfad zu einer signierten Quorum-Token-Datei an, um den Vorgang zu genehmigen. Nur erforderlich, wenn der Quorumwert des Entpackungsschlüssels für den Schlüsselverwaltungsdienst größer als 1 ist.

Verwandte Themen

- [Der Key Wrap-Befehl in der CloudHSM-CLI](#)
- [Der Befehl zum Entpacken von Schlüsseln in der CloudHSM-CLI](#)

Entpacken Sie einen Schlüssel mit AES- PKCS5 -PAD mithilfe der CloudHSM-CLI

Verwenden Sie den `key unwrap aes-pkcs5-pad` Befehl in der CloudHSM-CLI, um einen Payload-Schlüssel mithilfe des AES-Wrapping-Schlüssels und des Entpackungsmechanismus zu entpacken.
AES-PKCS5-PAD

Unverpackte Schlüssel können auf die gleiche Weise verwendet werden wie die Schlüssel, die von generiert wurden. AWS CloudHSM Um anzuzeigen, dass sie nicht lokal generiert wurden, ist ihr `local` Attribut auf `false` gesetzt.

Um den `key unwrap aes-pkcs5-pad` Befehl verwenden zu können, müssen Sie den AES-Wrapping-Schlüssel in Ihrem AWS CloudHSM Cluster haben und sein `unwrap` Attribut muss auf `gesetzt sein true`.

Benutzertyp

Die folgenden Benutzertypen können diesen Befehl ausführen.

- Crypto-Benutzer (CUs)

Voraussetzungen

- Um diesen Befehl auszuführen, müssen Sie als CU angemeldet sein.

Syntax

```
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

Beispiele

Diese Beispiele zeigen, wie der `key unwrap aes-pkcs5-pad` Befehl mithilfe eines AES-Schlüssels verwendet wird, dessen `unwrap` Attributwert auf `gesetzt` ist `true`.

Example Beispiel: Entpacken Sie einen Payload-Schlüssel aus Base64-kodierten umschlossenen Schlüsseldaten

```
aws-cloudhsm > key unwrap aes-pkcs5-pad --key-type-class aes --label aes-unwrapped --
filter attr.label=aes-example --data MbuYNresf0KyGNnxKwen88nSfX+uUE/0qmGofSisicY=
{
  "error_code": 0,
  "data": {
    "key": {
      "key-reference": "0x000000000001c08e3",
      "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 Beispiel: Entpacken Sie einen Payload-Schlüssel, der über einen Datenpfad bereitgestellt wird

```

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": "0x000000000001c08e3",
      "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
  }
}
}
}

```

Argumente

<CLUSTER_ID>

Die ID des Clusters, auf dem dieser Vorgang ausgeführt werden soll.

Erforderlich: Wenn mehrere Cluster [konfiguriert wurden](#).

<FILTER>

Schlüsselreferenz (z. B. key-reference=0xabc) oder durch Leerzeichen getrennte Liste von Schlüsselattributen in der Form `tr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE`, dass ein Schlüssel zum Entpacken ausgewählt werden soll.

Erforderlich: Ja

<DATA_PATH>

Pfad zur Binärdatei, die die umschlossenen Schlüsseldaten enthält.

Erforderlich: Ja (sofern nicht über Base64-kodierte Daten bereitgestellt)

<DATA>

Base64-kodierte umhüllte Schlüsseldaten.

Erforderlich: Ja (sofern nicht über den Datenpfad angegeben)

<ATTRIBUTES>

Durch Leerzeichen getrennte Liste von Schlüsselattributen in der Form von KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE für den umschlossenen Schlüssel.

Erforderlich: Nein

<KEY_TYPE_CLASS>

Schlüsseltyp und Klasse des umschlossenen Schlüssels [mögliche Werte: aesdes3,ec-private,generic-secret,,rsa-private].

Erforderlich: Ja

<LABEL>

Bezeichnung für den entpackten Schlüssel.

Erforderlich: Ja

<SESSION>

Erstellt einen Sitzungsschlüssel, der nur in der aktuellen Sitzung existiert. Der Schlüssel kann nach Ende der Sitzung nicht wiederhergestellt werden.

Erforderlich: Nein

<APPROVAL>

Gibt den Dateipfad zu einer signierten Quorum-Token-Datei an, um den Vorgang zu genehmigen. Nur erforderlich, wenn der Quorumwert des Entpackungsschlüssels für den Schlüsselverwaltungsdienst größer als 1 ist.

Verwandte Themen

- [Der Key Wrap-Befehl in der CloudHSM-CLI](#)
- [Der Befehl zum Entpacken von Schlüsseln in der CloudHSM-CLI](#)

Entpacken Sie einen Schlüssel AES-ZERO-PAD mithilfe der CloudHSM-CLI

Verwenden Sie den `key unwrap aes-zero-pad` Befehl in der CloudHSM-CLI, um einen Payload-Schlüssel mithilfe des AES-Wrapping-Schlüssels und des Entpackungsmechanismus in den AWS CloudHSM Cluster zu entpacken. AES-ZERO-PAD

Unverpackte Schlüssel können auf die gleiche Weise verwendet werden wie die Schlüssel, die von generiert wurden. AWS CloudHSM Um anzuzeigen, dass sie nicht lokal generiert wurden, ist ihr `local` Attribut auf `false` gesetzt.

Um den `key unwrap aes-no-pad` Befehl verwenden zu können, müssen Sie den AES-Wrapping-Schlüssel in Ihrem AWS CloudHSM Cluster haben und sein `unwrap` Attribut muss auf `gesetzt` sein `true`.

Benutzertyp

Die folgenden Benutzertypen können diesen Befehl ausführen.

- Crypto-Benutzer (CUs)

Voraussetzungen

- Um diesen Befehl auszuführen, müssen Sie als CU angemeldet sein.

Syntax

```
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

```

Beispiele

Diese Beispiele zeigen, wie der key unwrap aes-zero-pad Befehl mithilfe eines AES-Schlüssels verwendet wird, dessen unwrap Attributwert auf gesetzt ist true.

Example Beispiel: Entpacken Sie einen Payload-Schlüssel aus Base64-kodierten umschlossenen Schlüsseldaten

```

aws-cloudhsm > key unwrap aes-zero-pad --key-type-class aes --label aes-unwrapped --
filter attr.label=aes-example --data L1wV1L/YeBNVAw6Mpk3owFJZXBzDL0nt
{
  "error_code": 0,
  "data": {

```

```

"key": {
  "key-reference": "0x000000000001c08e7",
  "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 Beispiel: Entpacken Sie einen Payload-Schlüssel, der über einen Datenpfad bereitgestellt wird

```
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": "0x000000000001c08e7",
      "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
  }
}
}
}

```

Argumente

<CLUSTER_ID>

Die ID des Clusters, auf dem dieser Vorgang ausgeführt werden soll.

Erforderlich: Wenn mehrere Cluster [konfiguriert wurden](#).

<FILTER>

Schlüsselreferenz (z. B. `key-reference=0xabc`) oder durch Leerzeichen getrennte Liste von Schlüsselattributen in der Form `tr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE`, dass ein Schlüssel zum Entpacken ausgewählt werden soll.

Erforderlich: Ja

<DATA_PATH>

Pfad zur Binärdatei, die die umschlossenen Schlüsseldaten enthält.

Erforderlich: Ja (sofern nicht über Base64-kodierte Daten bereitgestellt)

<DATA>

Base64-kodierte umhüllte Schlüsseldaten.

Erforderlich: Ja (sofern nicht über den Datenpfad angegeben)

<ATTRIBUTES>

Durch Leerzeichen getrennte Liste von Schlüsselattributen in der Form von `KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE` für den umschlossenen Schlüssel.

Erforderlich: Nein

<KEY_TYPE_CLASS>

Schlüsseltyp und Klasse des umschlossenen Schlüssels [mögliche Werte: aesdes3,ec-private,generic-secret,,rsa-private].

Erforderlich: Ja

<LABEL>

Bezeichnung für den entpackten Schlüssel.

Erforderlich: Ja

<SESSION>

Erstellt einen Sitzungsschlüssel, der nur in der aktuellen Sitzung existiert. Der Schlüssel kann nach Ende der Sitzung nicht wiederhergestellt werden.

Erforderlich: Nein

<APPROVAL>

Gibt den Dateipfad zu einer signierten Quorum-Token-Datei an, um den Vorgang zu genehmigen. Nur erforderlich, wenn der Quorumwert des Entpackungsschlüssels für den Schlüsselverwaltungsdienst größer als 1 ist.

Verwandte Themen

- [Der Key Wrap-Befehl in der CloudHSM-CLI](#)
- [Der Befehl zum Entpacken von Schlüsseln in der CloudHSM-CLI](#)

Entpacken Sie einen Schlüssel CLOUDHSM-AES-GCM mithilfe der CloudHSM-CLI

Verwenden Sie den `key unwrap cloudhsm-aes-gcm` Befehl in der CloudHSM-CLI, um einen Payload-Schlüssel mithilfe des AES-Wrapping-Schlüssels und des Entpackungsmechanismus in den AWS CloudHSM Cluster zu entpacken. CLOUDHSM-AES-GCM

Unverpackte Schlüssel können auf die gleiche Weise verwendet werden wie die Schlüssel, die von generiert wurden. AWS CloudHSM Um anzuzeigen, dass sie nicht lokal generiert wurden, ist ihr `local` Attribut auf `false` gesetzt.

Um den `key unwrap cloudhsm-aes-gcm` Befehl verwenden zu können, müssen Sie den AES-Wrapping-Schlüssel in Ihrem AWS CloudHSM Cluster haben und sein `unwrap` Attribut muss auf `true` gesetzt sein.

Benutzertyp

Die folgenden Benutzertypen können diesen Befehl ausführen.

- Krypto-Benutzer (CUs)

Voraussetzungen

- Um diesen Befehl auszuführen, müssen Sie als CU angemeldet sein.

Syntax

```
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

```

Beispiele

Diese Beispiele zeigen, wie der `key unwrap cloudhsm-aes-gcm` Befehl mithilfe eines AES-Schlüssels verwendet wird, dessen `unwrap` Attributwert auf `true` gesetzt ist.

Example Beispiel: Entpacken Sie einen Payload-Schlüssel aus Base64-kodierten umschlossenen Schlüsseldaten

```

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
6Rn8nkjEriDYlnP3P8nPkYQ8hp10EJ899zsrF+aTB0i/fI1Z
{
  "error_code": 0,
  "data": {
    "key": {
      "key-reference": "0x000000000001408e8",
      "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 Beispiel: Entpacken Sie einen Payload-Schlüssel, der über einen Datenpfad bereitgestellt wird

```

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": "0x000000000001408e8",
    "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
    }
  }
}
```

```
}  
}
```

Argumente

<CLUSTER_ID>

Die ID des Clusters, auf dem dieser Vorgang ausgeführt werden soll.

Erforderlich: Wenn mehrere Cluster [konfiguriert wurden](#).

<FILTER>

Schlüsselreferenz (z. B. `key-reference=0xabc`) oder durch Leerzeichen getrennte Liste von Schlüsselattributen in der Form `tr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE`, dass ein Schlüssel zum Entpacken ausgewählt werden soll.

Erforderlich: Ja

<DATA_PATH>

Pfad zur Binärdatei, die die umschlossenen Schlüsseldaten enthält.

Erforderlich: Ja (sofern nicht über Base64-kodierte Daten bereitgestellt)

<DATA>

Base64-kodierte umhüllte Schlüsseldaten.

Erforderlich: Ja (sofern nicht über den Datenpfad angegeben)

<ATTRIBUTES>

Durch Leerzeichen getrennte Liste von Schlüsselattributen in der Form von `KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE` für den umschlossenen Schlüssel.

Erforderlich: Nein

<AAD>

AES GCM-Wert für zusätzliche authentifizierte Daten (AAD) in Hexadezimalzahl.

Erforderlich: Nein

<TAG_LENGTH_BITS>

Länge des Aes GCM-Tags in Bit.

Erforderlich: Ja

<KEY_TYPE_CLASS>

Schlüsseltyp und Klasse des umschlossenen Schlüssels [mögliche Werte:aes,des3,ec-private,generic-secret,rsa-private].

Erforderlich: Ja

<LABEL>

Bezeichnung für den entpackten Schlüssel.

Erforderlich: Ja

<SESSION>

Erstellt einen Sitzungsschlüssel, der nur in der aktuellen Sitzung existiert. Der Schlüssel kann nach Ende der Sitzung nicht wiederhergestellt werden.

Erforderlich: Nein

<APPROVAL>

Gibt den Dateipfad zu einer signierten Quorum-Token-Datei an, um den Vorgang zu genehmigen. Nur erforderlich, wenn der Quorumwert des Entpackungsschlüssels für den Schlüsselverwaltungsdienst größer als 1 ist.

Verwandte Themen

- [Der Key Wrap-Befehl in der CloudHSM-CLI](#)
- [Der Befehl zum Entpacken von Schlüsseln in der CloudHSM-CLI](#)

Entpacken Sie einen Schlüssel mit RSA-AES mithilfe der CloudHSM-CLI

Verwenden Sie den `key unwrap rsa-aes` Befehl in der CloudHSM-CLI, um einen Payload-Schlüssel mithilfe eines privaten RSA-Schlüssels und des Entpackungsmechanismus zu entpacken. RSA-AES

Unverpackte Schlüssel können auf die gleiche Weise verwendet werden wie die Schlüssel, die von generiert wurden. AWS CloudHSM Um anzuzeigen, dass sie nicht lokal generiert wurden, ist ihr `local` Attribut auf `false` gesetzt.

Um den verwenden zu könnenkey unwrap rsa-aes, müssen Sie den privaten RSA-Schlüssel oder den öffentlichen RSA-Wrapping-Schlüssel in Ihrem AWS CloudHSM Cluster haben, und sein unwrap Attribut muss auf gesetzt sein. true

Note

Dieser Befehl ist nur mit CloudHSM CLI 5.11+ verfügbar.

Benutzertyp

Die folgenden Benutzertypen können diesen Befehl ausführen.

- Crypto-Benutzer () CUs

Voraussetzungen

- Um diesen Befehl auszuführen, müssen Sie als CU angemeldet sein.

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

```

Beispiel

Diese Beispiele zeigen, wie der `key unwrap rsa-aes` Befehl unter Verwendung des privaten RSA-Schlüssels mit dem `unwrap` Attributwert auf `true` verwendet wird.

Example Beispiel: Entpacken Sie einen Payload-Schlüssel aus Base64-kodierten verpackten Schlüsseldaten

```

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
+NcerbAJ08DJ11XPygZcI21vIFSZJuWMEiWpe1R9D/5WSYgxLVKex30xCFqebtEzxbKuv4D0mU4meSofqREYvtb3EoIKwjy
+RL5WGxKe4nAboAkC5G07veI5yHL1SaKlssSJtTL/CFpbSLsAFuYbv/NUCWwMY5mwyVTCs1w+HlgKK
+5TH1MzBaSi8fpfyepLT8sHy2Q/VR16ifb49p6m0KQFbRVvz/0WUd614d97BdgtaEz6ueg==
{
  "error_code": 0,
  "data": {

```



```

"key": {
  "key-reference": "0x000000000001808e2",
  "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 Beispiel: Entpacken Sie einen Payload-Schlüssel, der über einen Datenpfad bereitgestellt wird

```
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": "0x000000000001808e2",
      "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  
  }  
}  
}
```

Argumente

<CLUSTER_ID>

Die ID des Clusters, auf dem dieser Vorgang ausgeführt werden soll.

Erforderlich: Wenn mehrere Cluster [konfiguriert wurden](#).

<FILTER>

Schlüsselreferenz (z. B. `key-reference=0xabc`) oder durch Leerzeichen getrennte Liste von Schlüsselattributen in der Form `tr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE`, dass ein Schlüssel zum Entpacken ausgewählt werden soll.

Erforderlich: Ja

<DATA_PATH>

Pfad zur Binärdatei, die die umschlossenen Schlüsseldaten enthält.

Erforderlich: Ja (sofern nicht über Base64-kodierte Daten bereitgestellt)

<DATA>

Base64-kodierte umhüllte Schlüsseldaten.

Erforderlich: Ja (sofern nicht über den Datenpfad angegeben)

<ATTRIBUTES>

Durch Leerzeichen getrennte Liste von Schlüsselattributen in der Form von `KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE` für den umschlossenen Schlüssel.

Erforderlich: Nein

<KEY_TYPE_CLASS>

Schlüsseltyp und Klasse des umschlossenen Schlüssels [mögliche Werte: aesdes3,ec-private,generic-secret,,rsa-private].

Erforderlich: Ja

<HASH_FUNCTION>

Spezifiziert die Hash-Funktion.

Zulässige Werte:

- sha1
- sha224
- sha256
- sha384
- sha512

Erforderlich: Ja

<MGF>

Gibt die Funktion zur Maskengenerierung an.

Note

Die Hash-Funktion der Maskengenerierungsfunktion muss mit der Hash-Funktion des Signaturmechanismus übereinstimmen.

Zulässige Werte:

- mgf1-sha1
- mgf1-sha224
- mgf1-sha256
- mgf1-sha384
- mgf1-sha512

Erforderlich: Ja

<LABEL>

Etikett für den unverpackten Schlüssel.

Erforderlich: Ja

<SESSION>

Erstellt einen Sitzungsschlüssel, der nur in der aktuellen Sitzung existiert. Der Schlüssel kann nach Ende der Sitzung nicht wiederhergestellt werden.

Erforderlich: Nein

<APPROVAL>

Gibt den Dateipfad zu einer signierten Quorum-Token-Datei an, um den Vorgang zu genehmigen. Nur erforderlich, wenn der Quorumwert des Entpackungsschlüssels für den Schlüsselverwaltungsdienst größer als 1 ist.

Verwandte Themen

- [Der Key Wrap-Befehl in der CloudHSM-CLI](#)
- [Der Befehl zum Entpacken von Schlüsseln in der CloudHSM-CLI](#)

Entpacken Sie einen Schlüssel mit RSA-OAEP mithilfe der CloudHSM-CLI

Verwenden Sie den `key unwrap rsa-oaep` Befehl in der CloudHSM-CLI, um einen Payload-Schlüssel mithilfe des privaten RSA-Schlüssels und des Entpackungsmechanismus zu entpacken. RSA-OAEP

Unverpackte Schlüssel können auf die gleiche Weise verwendet werden wie die Schlüssel, die von generiert wurden. AWS CloudHSM Um anzuzeigen, dass sie nicht lokal generiert wurden, ist ihr `local` Attribut auf `false` gesetzt.

Um den `key unwrap rsa-oaep` Befehl verwenden zu können, müssen Sie den privaten RSA-Schlüssel oder den öffentlichen RSA-Wrapping-Schlüssel in Ihrem AWS CloudHSM Cluster haben, und sein `unwrap` Attribut muss auf `true` gesetzt sein.

Benutzertyp

Die folgenden Benutzertypen können diesen Befehl ausführen.

- Crypto-Benutzer () CUs

Voraussetzungen

- Um diesen Befehl auszuführen, müssen Sie als CU angemeldet sein.

Syntax

```
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

```

Beispiele

Diese Beispiele zeigen, wie der `key unwrap rsa-oaep` Befehl unter Verwendung des privaten RSA-Schlüssels mit dem `unwrap` Attributwert auf `true` verwendet wird.

Example Beispiel: Entpacken Sie einen Payload-Schlüssel aus Base64-kodierten verpackten Schlüsseldaten

```

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
OjJe4msobPLz9TuSAdULEu17T5rMDWtS1LyBSkLbaZnYzzpdrhsbGLbwZJCtB/jGkDNdB4qyTA0QwEpggGf6v
+Yx6JcesNeKKNU8XZa1/YBoHC8noTGUSDi2qr+u2tDc84NPv6d+F2K00NXsSxMhmzzzNG/
gzTVIJh0uy/B1yHjGP4m0XoDZf5+7f5M1CjxBmz4Vva/wrWHGCSG0y0aWb1Ev0iHAIIt3UBdyKmU+/
My4xjfJv7WGGu3DFUUIZ06TihRtKQhUYU1M9u6NPf9riJJfHsk6QCUSZ9yWThDT9as6i7e3htnyDhIhGWaoK8JU855cN/
YNKAUqkNpC4FPL3iw==
{
  "data": {
    "key": {
      "key-reference": "0x0000000000001808e9",
      "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 Beispiel: Entpacken Sie einen Payload-Schlüssel, der über einen Datenpfad bereitgestellt wird

```

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": "0x000000000001808e9",
      "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
}
}
}
}

```

Argumente

<CLUSTER_ID>

Die ID des Clusters, auf dem dieser Vorgang ausgeführt werden soll.

Erforderlich: Wenn mehrere Cluster [konfiguriert wurden](#).

<FILTER>

Schlüsselreferenz (z. B. `key-reference=0xabc`) oder durch Leerzeichen getrennte Liste von Schlüsselattributen in der Form `tr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE`, dass ein Schlüssel zum Entpacken ausgewählt werden soll.

Erforderlich: Ja

<DATA_PATH>

Pfad zur Binärdatei, die die umschlossenen Schlüsseldaten enthält.

Erforderlich: Ja (sofern nicht über Base64-kodierte Daten bereitgestellt)

<DATA>

Base64-kodierte umhüllte Schlüsseldaten.

Erforderlich: Ja (sofern nicht über den Datenpfad angegeben)

<ATTRIBUTES>

Durch Leerzeichen getrennte Liste von Schlüsselattributen in der Form von `KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE` für den umschlossenen Schlüssel.

Erforderlich: Nein

<KEY_TYPE_CLASS>

Schlüsseltyp und Klasse des umschlossenen Schlüssels [mögliche Werte: `aesdes3`, `ec-private`, `generic-secret`, `rsa-private`].

Erforderlich: Ja

<HASH_FUNCTION>

Spezifiziert die Hash-Funktion.

Zulässige Werte:

- sha1
- sha224
- sha256
- sha384
- sha512

Erforderlich: Ja

<MGF>

Gibt die Funktion zur Maskengenerierung an.

Note

Die Hash-Funktion der Maskengenerierungsfunktion muss mit der Hash-Funktion des Signaturmechanismus übereinstimmen.

Zulässige Werte:

- mgf1-sha1
- mgf1-sha224
- mgf1-sha256
- mgf1-sha384
- mgf1-sha512

Erforderlich: Ja

<LABEL>

Etikett für den unverpackten Schlüssel.

Erforderlich: Ja

<SESSION>

Erstellt einen Sitzungsschlüssel, der nur in der aktuellen Sitzung existiert. Der Schlüssel kann nach Ende der Sitzung nicht wiederhergestellt werden.

Erforderlich: Nein

<APPROVAL>

Gibt den Dateipfad zu einer signierten Quorum-Token-Datei an, um den Vorgang zu genehmigen. Nur erforderlich, wenn der Quorumwert des Entpackungsschlüssels für den Schlüsselverwaltungsdienst größer als 1 ist.

Verwandte Themen

- [Der Key Wrap-Befehl in der CloudHSM-CLI](#)
- [Der Befehl zum Entpacken von Schlüsseln in der CloudHSM-CLI](#)

Entpacken Sie einen Schlüssel mit RSA-PKCS mithilfe der CloudHSM-CLI

Verwenden Sie den `key unwrap rsa-pkcs` Befehl in der CloudHSM-CLI, um einen Payload-Schlüssel mithilfe des privaten RSA-Schlüssels und des Entpackungsmechanismus zu entpacken. RSA-PKCS

Unverpackte Schlüssel können auf die gleiche Weise verwendet werden wie die Schlüssel, die von generiert wurden. AWS CloudHSM Um anzuzeigen, dass sie nicht lokal generiert wurden, ist ihr `local` Attribut auf `false` gesetzt.

Um den `unwrap rsa-pkcs` Befehl `key` verwenden zu können, müssen Sie den privaten RSA-Schlüssel oder den öffentlichen RSA-Wrapping-Schlüssel in Ihrem AWS CloudHSM Cluster haben, und sein `unwrap` Attribut muss auf `true` gesetzt sein.

Benutzertyp

Die folgenden Benutzertypen können diesen Befehl ausführen.

- Crypto-Benutzer () CUs

Voraussetzungen

- Um diesen Befehl auszuführen, müssen Sie als CU angemeldet sein.

Syntax

```
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

Beispiele

Diese Beispiele zeigen, wie der key unwrap rsa-oaep Befehl mithilfe eines AES-Schlüssels verwendet wird, dessen unwrap Attributwert auf gesetzt ist true.

Example Beispiel: Entpacken Sie einen Payload-Schlüssel aus Base64-kodierten umschlossenen Schlüsseldaten

```
aws-cloudhsm > key unwrap rsa-pkcs --key-type-class aes --label
aes-unwrapped --filter attr.label=rsa-private-key-example --data
am0Nc7+YE8FWs+5HvU7sIBcXVb24QA0165nbNAD+1bK+e18BpSfnaI3P+r8Dp+pLu1ofouy/
vtzRjZoCiDofcz4EqCFnG14GdcJ1/3W/5WRvMatCa2d7cx02swaeZcjKsermPXYR011G1fq6NskwMeeTkV8R7Rx9artFrs1
c3XdFJ2+0Bo94c6og/
yfPcp00obJlITCoXhtMRepSd040ggYq/6nUDuHCtJ86pPGnNahyr7+sAaSI3a5ECQLUjwaIARUCyoRh7EFK3qPXcg==
{
  "error_code": 0,
  "data": {
    "key": {
      "key-reference": "0x000000000001c08ef",
      "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 Beispiel: Entpacken Sie einen Payload-Schlüssel, der über einen Datenpfad bereitgestellt wird

```

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": "0x000000000001c08ef",
      "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
  }
}
}
}

```

Argumente

<CLUSTER_ID>

Die ID des Clusters, auf dem dieser Vorgang ausgeführt werden soll.

Erforderlich: Wenn mehrere Cluster [konfiguriert wurden](#).

<FILTER>

Schlüsselreferenz (z. B. key-reference=0xabc) oder durch Leerzeichen getrennte Liste von Schlüsselattributen in der Form `tr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE`, dass ein Schlüssel zum Entpacken ausgewählt werden soll.

Erforderlich: Ja

<DATA_PATH>

Pfad zur Binärdatei, die die umschlossenen Schlüsseldaten enthält.

Erforderlich: Ja (sofern nicht über Base64-kodierte Daten bereitgestellt)

<DATA>

Base64-kodierte umhüllte Schlüsseldaten.

Erforderlich: Ja (sofern nicht über den Datenpfad angegeben)

<ATTRIBUTES>

Durch Leerzeichen getrennte Liste von Schlüsselattributen in der Form von KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE für den umschlossenen Schlüssel.

Erforderlich: Nein

<KEY_TYPE_CLASS>

Schlüsseltyp und Klasse des umschlossenen Schlüssels [mögliche Werte: aesdes3,ec-private,generic-secret,,rsa-private].

Erforderlich: Ja

<LABEL>

Bezeichnung für den entpackten Schlüssel.

Erforderlich: Ja

<SESSION>

Erstellt einen Sitzungsschlüssel, der nur in der aktuellen Sitzung existiert. Der Schlüssel kann nach Ende der Sitzung nicht wiederhergestellt werden.

Erforderlich: Nein

<APPROVAL>

Gibt den Dateipfad zu einer signierten Quorum-Token-Datei an, um den Vorgang zu genehmigen. Nur erforderlich, wenn der Quorumwert des Entpackungsschlüssels für den Schlüsselverwaltungsdienst größer als 1 ist.

Verwandte Themen

- [Der Key Wrap-Befehl in der CloudHSM-CLI](#)
- [Der Befehl zum Entpacken von Schlüsseln in der CloudHSM-CLI](#)

Der Key Wrap-Befehl in der CloudHSM-CLI

Der key wrap Befehl in der CloudHSM-CLI exportiert eine verschlüsselte Kopie eines symmetrischen oder asymmetrischen privaten Schlüssels aus dem Hardware-Sicherheitsmodul (HSM) in eine Datei. Bei der Ausführung key wrap geben Sie zwei Dinge an: den zu exportierenden Schlüssel und die Ausgabedatei. Der zu exportierende Schlüssel ist ein Schlüssel auf dem HSM, der den Schlüssel, den Sie exportieren möchten, verschlüsselt (umhüllt).

Der key wrap Befehl entfernt den Schlüssel nicht aus dem HSM und verhindert auch nicht, dass Sie ihn für kryptografische Operationen verwenden. Sie können den gleichen Schlüssel mehrmals exportieren. Um den verschlüsselten Schlüssel wieder in das HSM zu importieren, verwenden Sie. [Der Befehl zum Entpacken von Schlüsseln in der CloudHSM-CLI](#) Nur der Besitzer eines Schlüssels, d. h. der Crypto-Benutzer (CU), der den Schlüssel erstellt hat, kann den Schlüssel umschließen. Benutzer, mit denen der Schlüssel gemeinsam genutzt wird, können den Schlüssel nur für kryptografische Operationen verwenden.

Der key wrap Befehl besteht aus den folgenden Unterbefehlen:

- [aes-gcm](#)
- [aes-no-pad](#)
- [aes-pkcs5-pad](#)
- [aes-zero-pad](#)
- [cloudhsm-aes-gcm](#)
- [rsa-aes](#)
- [rsa-oaep](#)
- [rsa-pkcs](#)

Umschließen eines Schlüssels mit AES-GCM mithilfe der CloudHSM-CLI

Verwenden Sie den key wrap aes-gcm Befehl in der CloudHSM-CLI, um einen Payload-Schlüssel mithilfe eines AES-Schlüssels auf dem Hardware-Sicherheitsmodul (HSM) und dem Wrapping-Mechanismus zu umschließen. AES-GCM Das extractable Attribut des Payload-Schlüssels muss auf gesetzt sein. `true`

Nur der Besitzer eines Schlüssels, d. h. der Crypto-Benutzer (CU), der den Schlüssel erstellt hat, kann den Schlüssel umschließen. Benutzer, die den Schlüssel gemeinsam nutzen, können den Schlüssel für kryptografische Operationen verwenden.

Um den key wrap aes-gcm Befehl verwenden zu können, benötigen Sie zunächst einen AES-Schlüssel in Ihrem AWS CloudHSM Cluster. Sie können einen AES-Schlüssel zum Umschließen generieren, indem Sie den [Generieren Sie einen symmetrischen AES-Schlüssel mit CloudHSM CLI](#) Befehl verwenden und das wrap Attribut auf gesetzt habent true.

Benutzertyp

Die folgenden Benutzertypen können diesen Befehl ausführen.

- Crypto-Benutzer (CUs)

Voraussetzungen

- Um diesen Befehl auszuführen, müssen Sie als CU angemeldet sein.

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
```

Beispiel

Dieses Beispiel zeigt, wie der key wrap aes-gcm Befehl mit einem AES-Schlüssel verwendet wird.

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": "0x000000000001c08f1",
    "wrapping_key_reference": "0x000000000001c08ea",
    "iv": "0xf90613bb8e337ec0339aad21",
    "wrapped_key_data": "xvslgrtg8kHrzvekn97tLSIeokpPwV8"
  }
}
```

Argumente

<CLUSTER_ID>

Die ID des Clusters, auf dem dieser Vorgang ausgeführt werden soll.

Erforderlich: Wenn mehrere Cluster [konfiguriert wurden](#).

<PAYLOAD_FILTER>

Schlüsselreferenz (z. B. key-reference=0xabc) oder durch Leerzeichen getrennte Liste von Schlüsselattributen in der Form „attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE“. Zur Auswahl eines Payload-Schlüssels.

Erforderlich: Ja

<PATH>

Pfad zur Binärdatei, in der die verpackten Schlüsseldaten gespeichert werden.

Erforderlich: Nein

<WRAPPING_FILTER>

Schlüsselreferenz (z. B. `key-reference=0xabc`) oder durch Leerzeichen getrennte Liste von Schlüsselattributen in der Form „`attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE`“ Zur Auswahl eines Umbruchschlüssels“.

Erforderlich: Ja

<AAD>

AES-GCM-Wert für zusätzliche authentifizierte Daten (AAD) in Hexadezimalzahl.

Erforderlich: Nein

<TAG_LENGTH_BITS>

Länge des AES-GCM-Tags in Bit.

Erforderlich: Ja

<WRAPPING_APPROVALR>

Gibt den Dateipfad zu einer signierten Quorum-Token-Datei an, um den Vorgang zum Umschließen des Schlüssels zu genehmigen. Nur erforderlich, wenn der Quorumwert für den Schlüsselverwaltungsdienst beim Umschließen des Schlüssels größer als 1 ist.

<PAYLOAD_APPROVALR>

Gibt den Dateipfad zu einer signierten Quorum-Token-Datei an, um den Vorgang für den Payload-Schlüssel zu genehmigen. Nur erforderlich, wenn der Quorumwert für den Schlüsselverwaltungsdienst des Payload-Schlüssels größer als 1 ist.

Verwandte Themen

- [Der Key Wrap-Befehl in der CloudHSM-CLI](#)
- [Der Befehl zum Entpacken von Schlüsseln in der CloudHSM-CLI](#)

Einen Schlüssel AES-NO-PAD mithilfe der CloudHSM-CLI umschließen

Verwenden Sie den `key wrap aes-no-pad` Befehl in der CloudHSM-CLI, um einen Payload-Schlüssel mithilfe eines AES-Schlüssels auf dem Hardware-Sicherheitsmodul (HSM) und dem Wrapping-Mechanismus zu umschließen. AES-NO-PAD Das `extractable` Attribut des Payload-Schlüssels muss auf `true` gesetzt sein.

Nur der Besitzer eines Schlüssels, d. h. der Crypto-Benutzer (CU), der den Schlüssel erstellt hat, kann den Schlüssel umschließen. Benutzer, die den Schlüssel gemeinsam nutzen, können den Schlüssel für kryptografische Operationen verwenden.

Um den `key wrap aes-no-pad` Befehl verwenden zu können, benötigen Sie zunächst einen AES-Schlüssel in Ihrem AWS CloudHSM Cluster. Sie können mithilfe des [Generieren Sie einen symmetrischen AES-Schlüssel mit CloudHSM CLI](#) Befehls einen AES-Schlüssel für das Umschließen generieren, und das `wrap` Attribut ist auf `gesetzttrue`.

Benutzertyp

Die folgenden Benutzertypen können diesen Befehl ausführen.

- Crypto-Benutzer (CUs)

Voraussetzungen

- Um diesen Befehl auszuführen, müssen Sie als CU angemeldet sein.

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_APPROVAL>
```

```

    File path of signed quorum token file to approve operation for wrapping key
--payload-approval <PAYLOAD_APPROVAL>
    File path of signed quorum token file to approve operation for payload key
-h, --help
    Print help

```

Beispiel

Dieses Beispiel zeigt, wie der key wrap aes-no-pad Befehl mit einem AES-Schlüssel verwendet wird, dessen wrap Attributwert auf gesetzt ist `true`.

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": "0x000000000001c08f1",
    "wrapping_key_reference": "0x000000000001c08ea",
    "wrapped_key_data": "eXK3PMA0nKM9y3YX6brbhtMoC060E0H9"
  }
}

```

Argumente

<CLUSTER_ID>

Die ID des Clusters, auf dem dieser Vorgang ausgeführt werden soll.

Erforderlich: Wenn mehrere Cluster [konfiguriert wurden](#).

<PAYLOAD_FILTER>

Schlüsselreferenz (z. B. `key-reference=0xabc`) oder durch Leerzeichen getrennte Liste von Schlüsselattributen in der Form „`attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE`“ Zur Auswahl eines Payload-Schlüssels“.

Erforderlich: Ja

<PATH>

Pfad zur Binärdatei, in der die verpackten Schlüsseldaten gespeichert werden.

Erforderlich: Nein

<WRAPPING_FILTER>

Schlüsselreferenz (z. B. `key-reference=0xabc`) oder durch Leerzeichen getrennte Liste von Schlüsselattributen in der Form „`attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE`“ Zur Auswahl eines Umbruchschlüssels“.

Erforderlich: Ja

<WRAPPING_APPROVALR>

Gibt den Dateipfad zu einer signierten Quorum-Token-Datei an, um den Vorgang zum Umschließen des Schlüssels zu genehmigen. Nur erforderlich, wenn der Quorumwert für den Schlüsselverwaltungsdienst beim Umschließen des Schlüssels größer als 1 ist.

<PAYLOAD_APPROVALR>

Gibt den Dateipfad zu einer signierten Quorum-Token-Datei an, um den Vorgang für den Payload-Schlüssel zu genehmigen. Nur erforderlich, wenn der Quorumwert für den Schlüsselverwaltungsdienst des Payload-Schlüssels größer als 1 ist.

Verwandte Themen

- [Der Key Wrap-Befehl in der CloudHSM-CLI](#)
- [Der Befehl zum Entpacken von Schlüsseln in der CloudHSM-CLI](#)

Umschließen Sie einen Schlüssel mit AES- PKCS5 -PAD mithilfe der CloudHSM-CLI

Verwenden Sie den `key wrap aes-pkcs5-pad` Befehl in der CloudHSM-CLI, um einen Payload-Schlüssel mithilfe eines AES-Schlüssels auf dem Hardware-Sicherheitsmodul (HSM) und dem Wrapping-Mechanismus zu umschließen. AES-PKCS5-PAD Das `extractable` Attribut des Payload-Schlüssels muss auf `true` gesetzt sein.

Nur der Besitzer eines Schlüssels, d. h. der Crypto-Benutzer (CU), der den Schlüssel erstellt hat, kann den Schlüssel umschließen. Benutzer, die den Schlüssel gemeinsam nutzen, können den Schlüssel für kryptografische Operationen verwenden.

Um den `key wrap aes-pkcs5-pad` Befehl verwenden zu können, benötigen Sie zunächst einen AES-Schlüssel in Ihrem AWS CloudHSM Cluster. Sie können mithilfe des [Generieren Sie einen symmetrischen AES-Schlüssel mit CloudHSM CLI](#) Befehls einen AES-Schlüssel für das Umschließen generieren, und das `wrap` Attribut ist auf `true` gesetzt.

Benutzertyp

Die folgenden Benutzertypen können diesen Befehl ausführen.

- Crypto-Benutzer (CUs)

Voraussetzungen

- Um diesen Befehl auszuführen, müssen Sie als CU angemeldet sein.

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

Beispiel

Dieses Beispiel zeigt, wie der key wrap aes-pkcs5-pad Befehl mit einem AES-Schlüssel verwendet wird, dessen wrap Attributwert auf gesetzt ist true.

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": "0x000000000001c08f1",
    "wrapping_key_reference": "0x000000000001c08ea",
    "wrapped_key_data": "MbuYNresf0KyGNnxKWen88nSfX+uUE/0qmGofSisicY="
  }
}
```

Argumente

<CLUSTER_ID>

Die ID des Clusters, auf dem dieser Vorgang ausgeführt werden soll.

Erforderlich: Wenn mehrere Cluster [konfiguriert wurden](#).

<PAYLOAD_FILTER>

Schlüsselreferenz (z. B. `key-reference=0xabc`) oder durch Leerzeichen getrennte Liste von Schlüsselattributen in der Form „`attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE`“ Zur Auswahl eines Payload-Schlüssels“.

Erforderlich: Ja

<PATH>

Pfad zur Binärdatei, in der die verpackten Schlüsseldaten gespeichert werden.

Erforderlich: Nein

<WRAPPING_FILTER>

Schlüsselreferenz (z. B. `key-reference=0xabc`) oder durch Leerzeichen getrennte Liste von Schlüsselattributen in der Form „`attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE`“ Zur Auswahl eines Umbruchschlüssels“.

Erforderlich: Ja

<WRAPPING_APPROVALR>

Gibt den Dateipfad zu einer signierten Quorum-Token-Datei an, um den Vorgang zum Umschließen des Schlüssels zu genehmigen. Nur erforderlich, wenn der Quorumwert für den Schlüsselverwaltungsdienst beim Umschließen des Schlüssels größer als 1 ist.

<PAYLOAD_APPROVALR>

Gibt den Dateipfad zu einer signierten Quorum-Token-Datei an, um den Vorgang für den Payload-Schlüssel zu genehmigen. Nur erforderlich, wenn der Quorumwert für den Schlüsselverwaltungsdienst des Payload-Schlüssels größer als 1 ist.

Verwandte Themen

- [Der Key Wrap-Befehl in der CloudHSM-CLI](#)
- [Der Befehl zum Entpacken von Schlüsseln in der CloudHSM-CLI](#)

Einen Schlüssel AES-ZERO-PAD mithilfe der CloudHSM-CLI umschließen

Verwenden Sie den `key wrap aes-zero-pad` Befehl in der CloudHSM-CLI, um einen Payload-Schlüssel mithilfe eines AES-Schlüssels auf dem Hardware-Sicherheitsmodul (HSM) und dem Wrapping-Mechanismus zu umschließen. AES-ZERO-PAD Das `extractable` Attribut des Payload-Schlüssels muss auf `true` gesetzt sein.

Nur der Besitzer eines Schlüssels, d. h. der Crypto-Benutzer (CU), der den Schlüssel erstellt hat, kann den Schlüssel umschließen. Benutzer, die den Schlüssel gemeinsam nutzen, können den Schlüssel für kryptografische Operationen verwenden.

Um den `key wrap aes-zero-pad` Befehl verwenden zu können, benötigen Sie zunächst einen AES-Schlüssel in Ihrem AWS CloudHSM Cluster. Mit dem [Generieren Sie einen symmetrischen AES-Schlüssel mit CloudHSM CLI](#) Befehl, dessen `wrap` Attribut auf `true` gesetzt ist, können Sie einen AES-Schlüssel für das Umschließen generieren.

Benutzertyp

Die folgenden Benutzertypen können diesen Befehl ausführen.

- Crypto-Benutzer (CUs)

Voraussetzungen

- Um diesen Befehl auszuführen, müssen Sie als CU angemeldet sein.

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

Beispiel

Dieses Beispiel zeigt, wie der key wrap aes-zero-pad Befehl mit einem AES-Schlüssel verwendet wird, dessen wrap Attributwert auf gesetzt ist true.

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": "0x000000000001c08f1",  
  "wrapping_key_reference": "0x000000000001c08ea",  
  "wrapped_key_data": "L1wV1L/YeBNVAw6Mpk3owFJZXBzDL0Nt"  
}  
}
```

Argumente

<CLUSTER_ID>

Die ID des Clusters, auf dem dieser Vorgang ausgeführt werden soll.

Erforderlich: Wenn mehrere Cluster [konfiguriert wurden](#).

<PAYLOAD_FILTER>

Schlüsselreferenz (z. B. key-reference=0xabc) oder durch Leerzeichen getrennte Liste von Schlüsselattributen in der Form „attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE“. Zur Auswahl eines Payload-Schlüssels“.

Erforderlich: Ja

<PATH>

Pfad zur Binärdatei, in der die verpackten Schlüsseldaten gespeichert werden.

Erforderlich: Nein

<WRAPPING_FILTER>

Schlüsselreferenz (z. B. key-reference=0xabc) oder durch Leerzeichen getrennte Liste von Schlüsselattributen in der Form „attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE“. Zur Auswahl eines Umbruchschlüssels“.

Erforderlich: Ja

<WRAPPING_APPROVALR>

Gibt den Dateipfad zu einer signierten Quorum-Token-Datei an, um den Vorgang zum Umschließen des Schlüssels zu genehmigen. Nur erforderlich, wenn der Quorumwert für den Schlüsselverwaltungsdienst beim Umschließen des Schlüssels größer als 1 ist.

<PAYLOAD_APPROVALR>

Gibt den Dateipfad zu einer signierten Quorum-Token-Datei an, um den Vorgang für den Payload-Schlüssel zu genehmigen. Nur erforderlich, wenn der Quorumwert für den Schlüsselverwaltungsdienst des Payload-Schlüssels größer als 1 ist.

Verwandte Themen

- [Der Key Wrap-Befehl in der CloudHSM-CLI](#)
- [Der Befehl zum Entpacken von Schlüsseln in der CloudHSM-CLI](#)

Einen Schlüssel CLOUDHSM-AES-GCM mithilfe der CloudHSM-CLI umschließen

Verwenden Sie den `key wrap cloudhsm-aes-gcm` Befehl in der CloudHSM-CLI, um einen Payload-Schlüssel mithilfe eines AES-Schlüssels auf dem Hardware-Sicherheitsmodul (HSM) und dem Wrapping-Mechanismus zu umschließen. CLOUDHSM-AES-GCM Das `extractable` Attribut des Payload-Schlüssels muss auf `gesetzt` sein. `true`

Nur der Besitzer eines Schlüssels, d. h. der Crypto-Benutzer (CU), der den Schlüssel erstellt hat, kann den Schlüssel umschließen. Benutzer, die den Schlüssel gemeinsam nutzen, können den Schlüssel für kryptografische Operationen verwenden.

Um den `key wrap cloudhsm-aes-gcm` Befehl verwenden zu können, benötigen Sie zunächst einen AES-Schlüssel in Ihrem AWS CloudHSM Cluster. Sie können einen AES-Schlüssel zum Umschließen generieren, indem Sie den [Generieren Sie einen symmetrischen AES-Schlüssel mit CloudHSM CLI](#) Befehl verwenden und das `wrap` Attribut auf `gesetzt` haben `true`.

Benutzertyp

Die folgenden Benutzertypen können diesen Befehl ausführen.

- Crypto-Benutzer (CUs)

Voraussetzungen

- Um diesen Befehl auszuführen, müssen Sie als CU angemeldet sein.

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

Beispiel

Dieses Beispiel zeigt, wie der key wrap cloudhsm-aes-gcm Befehl mit einem AES-Schlüssel verwendet wird.

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": "0x000000000001c08f1",
  "wrapping_key_reference": "0x000000000001c08ea",
  "wrapped_key_data": "6Rn8nkjEriDYlnP3P8nPkyQ8hpl0EJ899zsrF+aTB0i/fIlZ"
}
```

Argumente

<CLUSTER_ID>

Die ID des Clusters, auf dem dieser Vorgang ausgeführt werden soll.

Erforderlich: Wenn mehrere Cluster [konfiguriert wurden](#).

<PAYLOAD_FILTER>

Schlüsselreferenz (z. B. `key-reference=0xabc`) oder durch Leerzeichen getrennte Liste von Schlüsselattributen in der Form „`attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE`“ Zur Auswahl eines Payload-Schlüssels“.

Erforderlich: Ja

<PATH>

Pfad zur Binärdatei, in der die verpackten Schlüsseldaten gespeichert werden.

Erforderlich: Nein

<WRAPPING_FILTER>

Schlüsselreferenz (z. B. `key-reference=0xabc`) oder durch Leerzeichen getrennte Liste von Schlüsselattributen in der Form „`attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE`“ Zur Auswahl eines Umbruchschlüssels“.

Erforderlich: Ja

<AAD>

AES-GCM-Wert für zusätzliche authentifizierte Daten (AAD) in Hexadezimalzahl.

Erforderlich: Nein

<TAG_LENGTH_BITS>

Länge des AES-GCM-Tags in Bit.

Erforderlich: Ja

<WRAPPING_APPROVALR>

Gibt den Dateipfad zu einer signierten Quorum-Token-Datei an, um den Vorgang zum Umschließen des Schlüssels zu genehmigen. Nur erforderlich, wenn der Quorumwert für den Schlüsselverwaltungsdienst beim Umschließen des Schlüssels größer als 1 ist.

<PAYLOAD_APPROVALR>

Gibt den Dateipfad zu einer signierten Quorum-Token-Datei an, um den Vorgang für den Payload-Schlüssel zu genehmigen. Nur erforderlich, wenn der Quorumwert für den Schlüsselverwaltungsdienst des Payload-Schlüssels größer als 1 ist.

Verwandte Themen

- [Der Key Wrap-Befehl in der CloudHSM-CLI](#)
- [Der Befehl zum Entpacken von Schlüsseln in der CloudHSM-CLI](#)

Umschließen eines Schlüssels mit RSA-AES mithilfe der CloudHSM-CLI

Verwenden Sie den `key wrap rsa-aes` Befehl in der CloudHSM-CLI, um einen Payload-Schlüssel mithilfe eines öffentlichen RSA-Schlüssels auf dem Hardware-Sicherheitsmodul (HSM) und dem RSA-AES-Wrapping-Mechanismus zu verpacken. Das Attribut des Payload-Schlüssels muss auf `extractable true` gesetzt sein.

Nur der Besitzer eines Schlüssels, d. h. der Crypto-Benutzer (CU), der den Schlüssel erstellt hat, kann den Schlüssel umschließen. Benutzer, die den Schlüssel gemeinsam nutzen, können den Schlüssel für kryptografische Operationen verwenden.

Um den `key wrap rsa-aes` Befehl verwenden zu können, benötigen Sie zunächst einen RSA-Schlüssel in Ihrem AWS CloudHSM Cluster. Sie können ein RSA-Schlüsselpaar mit dem [Die generate-asymmetric-pair Kategorie in CloudHSM CLI](#) Befehl und dem auf `true` gesetzten `wrap` Attribut generieren.

Benutzertyp

Die folgenden Benutzertypen können diesen Befehl ausführen.

- Crypto-Benutzer () CUs

Voraussetzungen

- Um diesen Befehl auszuführen, müssen Sie als CU angemeldet sein.

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

Beispiel

Dieses Beispiel zeigt, wie der key wrap rsa-ae Befehl unter Verwendung eines öffentlichen RSA-Schlüssels mit dem wrap Attributwert auf true verwendet wird.

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": "0x000000000001c08f1",
    "wrapping-key-reference": "0x000000000007008da",
    "wrapped-key-data": "HrSE1DEyLjIeyGdPa9R+ebiqB5TIJGyamPker31ZebPwRA
+NcerbAJ08DJ11XPygZcI21vIFSZJuWMEiWpe1R9D/5WSYgxLVKex30xCFqebtEzxbKuv4D0mU4meSofqREYvtb3EoIKwjy
+RL5WGXKe4nAboAkC5G07veI5yHL1SaKlssSJtTL/CFpbSLsAFuYbv/NUCWwMY5mwyVTCSlw+HlgKK
+5TH1MzBaSi8fpfyepLT8sHy2Q/VRl6ifb49p6m0KQFbRVvz/0WUd6l4d97BdgtaEz6ueg=="
  }
}
```

Argumente

<CLUSTER_ID>

Die ID des Clusters, auf dem dieser Vorgang ausgeführt werden soll.

Erforderlich: Wenn mehrere Cluster [konfiguriert wurden](#).

<PAYLOAD_FILTER>

Schlüsselreferenz (z. B. `key-reference=0xabc`) oder durch Leerzeichen getrennte Liste von Schlüsselattributen in der Form „`attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE`“ Zur Auswahl eines Payload-Schlüssels“.

Erforderlich: Ja

<PATH>

Pfad zur Binärdatei, in der die verpackten Schlüsseldaten gespeichert werden.

Erforderlich: Nein

<WRAPPING_FILTER>

Schlüsselreferenz (z. B. `key-reference=0xabc`) oder durch Leerzeichen getrennte Liste von Schlüsselattributen in der Form „`attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE`“ Zur Auswahl eines Umbruchschlüssels“.

Erforderlich: Ja

<MGF>

Gibt die Funktion zur Maskengenerierung an.

Note

Die Hash-Funktion der Maskengenerierungsfunktion muss mit der Hash-Funktion des Signaturmechanismus übereinstimmen.

Zulässige Werte

- mgf1-sha1
- mgf1-sha224
- mgf1-sha256
- mgf1-sha384
- mgf1-sha512

Erforderlich: Ja

<WRAPPING_APPROVALR>

Gibt den Dateipfad zu einer signierten Quorum-Token-Datei an, um den Vorgang zum Umschließen des Schlüssels zu genehmigen. Nur erforderlich, wenn der Quorumwert für den Schlüsselverwaltungsdienst beim Umschließen des Schlüssels größer als 1 ist.

<PAYLOAD_APPROVALR>

Gibt den Dateipfad zu einer signierten Quorum-Token-Datei an, um den Vorgang für den Payload-Schlüssel zu genehmigen. Nur erforderlich, wenn der Quorumwert für den Schlüsselverwaltungsdienst des Payload-Schlüssels größer als 1 ist.

Verwandte Themen

- [Der Key Wrap-Befehl in der CloudHSM-CLI](#)
- [Der Befehl zum Entpacken von Schlüsseln in der CloudHSM-CLI](#)

Umschließen eines Schlüssels mit RSA-OAEP mithilfe der CloudHSM-CLI

Verwenden Sie den `key wrap rsa-oaep` Befehl in der CloudHSM-CLI, um einen Payload-Schlüssel mithilfe eines öffentlichen RSA-Schlüssels auf dem Hardware-Sicherheitsmodul (HSM) und dem Wrapping-Mechanismus zu verpacken. RSA-OAEP Das Attribut des Payload-Schlüssels `extractable` muss auf `true` gesetzt sein.

Nur der Besitzer eines Schlüssels, d. h. der Crypto-Benutzer (CU), der den Schlüssel erstellt hat, kann den Schlüssel umschließen. Benutzer, die den Schlüssel gemeinsam nutzen, können den Schlüssel für kryptografische Operationen verwenden.

Um den `key wrap rsa-oaep` Befehl verwenden zu können, benötigen Sie zunächst einen RSA-Schlüssel in Ihrem AWS CloudHSM Cluster. Sie können ein RSA-Schlüsselpaar generieren, indem Sie den [Die generate-asymmetric-pair Kategorie in CloudHSM CLI](#) Befehl verwenden und das `wrap` Attribut auf `true` setzen.

Benutzertyp

Die folgenden Benutzertypen können diesen Befehl ausführen.

- Crypto-Benutzer () CUs

Voraussetzungen

- Um diesen Befehl auszuführen, müssen Sie als CU angemeldet sein.

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

```

Beispiel

Dieses Beispiel zeigt, wie der key wrap rsa-oaep Befehl unter Verwendung eines öffentlichen RSA-Schlüssels mit dem wrap Attributwert auf true verwendet wird.

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": "0x000000000001c08f1",
    "wrapping-key-reference": "0x000000000007008da",
    "wrapped-key-data": "0jJe4msobPLz9TuSAdULEu17T5rMDWtS1LyBSkLbaZnYzzpdrhsbGLbwZJCtB/
jGkDNdB4qyTA0QwEpggGf6v+Yx6JcesNeKKNU8XZa1/YBoHC8noTGUSDI2qr+u2tDc84NPv6d
+F2K00NXsSxMhmxzzNG/gzTVIJh0uy/B1yHjGP4m0XoDZf5+7f5M1CjxBmz4Vva/
wrWHGCSG0y0aWblEv0iHAIIt3UBdyKmU+/
My4xjFJv7WGGu3DFUUIZ06TihRtKQhUYU1M9u6NPf9riJJfHsk6QCuSZ9yWThDT9as6i7e3htnyDhIhGwaoK8JU855cN/
YNKAUqkNpC4FPL3iw=="
  }
}

```

Argumente

<CLUSTER_ID>

Die ID des Clusters, auf dem dieser Vorgang ausgeführt werden soll.

Erforderlich: Wenn mehrere Cluster [konfiguriert wurden](#).

<PAYLOAD_FILTER>

Schlüsselreferenz (z. B. `key-reference=0xabc`) oder durch Leerzeichen getrennte Liste von Schlüsselattributen in der Form „`attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE`“ Zur Auswahl eines Payload-Schlüssels“.

Erforderlich: Ja

<PATH>

Pfad zur Binärdatei, in der die verpackten Schlüsseldaten gespeichert werden.

Erforderlich: Nein

<WRAPPING_FILTER>

Schlüsselreferenz (z. B. `key-reference=0xabc`) oder durch Leerzeichen getrennte Liste von Schlüsselattributen in der Form „`attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE`“ Zur Auswahl eines Umbruchschlüssels“.

Erforderlich: Ja

<MGF>

Gibt die Funktion zur Maskengenerierung an.

Note

Die Hash-Funktion der Maskengenerierungsfunktion muss mit der Hash-Funktion des Signaturmechanismus übereinstimmen.

Zulässige Werte

- `mgf1-sha1`

- mgf1-sha224
- mgf1-sha256
- mgf1-sha384
- mgf1-sha512

Erforderlich: Ja

<WRAPPING_APPROVALR>

Gibt den Dateipfad zu einer signierten Quorum-Token-Datei an, um den Vorgang zum Umschließen des Schlüssels zu genehmigen. Nur erforderlich, wenn der Quorumwert für den Schlüsselverwaltungsdienst beim Umschließen des Schlüssels größer als 1 ist.

<PAYLOAD_APPROVALR>

Gibt den Dateipfad zu einer signierten Quorum-Token-Datei an, um den Vorgang für den Payload-Schlüssel zu genehmigen. Nur erforderlich, wenn der Quorumwert für den Schlüsselverwaltungsdienst des Payload-Schlüssels größer als 1 ist.

Verwandte Themen

- [Der Key Wrap-Befehl in der CloudHSM-CLI](#)
- [Der Befehl zum Entpacken von Schlüsseln in der CloudHSM-CLI](#)

Umschließen eines Schlüssels mit RSA-PKCS mithilfe der CloudHSM-CLI

Verwenden Sie den `key wrap rsa-pkcs` Befehl in der CloudHSM-CLI, um einen Payload-Schlüssel mithilfe eines öffentlichen RSA-Schlüssels auf dem Hardware-Sicherheitsmodul (HSM) und dem Wrapping-Mechanismus zu verpacken. RSA-PKCS Das Attribut des Payload-Schlüssels `extractable` muss auf `true` gesetzt sein.

Nur der Besitzer eines Schlüssels, d. h. der Crypto-Benutzer (CU), der den Schlüssel erstellt hat, kann den Schlüssel umschließen. Benutzer, die den Schlüssel gemeinsam nutzen, können den Schlüssel für kryptografische Operationen verwenden.

Um den `key wrap rsa-pkcs` Befehl verwenden zu können, benötigen Sie zunächst einen RSA-Schlüssel in Ihrem AWS CloudHSM Cluster. Sie können ein RSA-Schlüsselpaar mit dem [Die generate-asymmetric-pair Kategorie in CloudHSM CLI](#) Befehl und dem auf `true` gesetzten `wrap` Attribut generieren.

Benutzertyp

Die folgenden Benutzertypen können diesen Befehl ausführen.

- Crypto-Benutzer () CUs

Voraussetzungen

- Um diesen Befehl auszuführen, müssen Sie als CU angemeldet sein.

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

Beispiel

Dieses Beispiel zeigt, wie der key wrap rsa-pkcs Befehl mit einem öffentlichen RSA-Schlüssel verwendet wird.

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": "0x000000000001c08f1",
    "wrapping_key_reference": "0x000000000007008da",
    "wrapped_key_data": "am0Nc7+YE8FWs+5HvU7sIBcXVb24QA0165nbNAD+1bK+e18BpSfnaI3P+r8Dp
+pLu1ofoUy/
vtzRjZoCiDofcz4EqCFnG14GdcJ1/3W/5WRvMatCa2d7cx02swaeZcjKsermPXYR011G1fq6NskwMeeTkV8R7Rx9artFrs1
c3XdFJ2+0Bo94c6og/
yfPcp00obJlITCoXhtMRepSd040ggYq/6nUDuHCtJ86pPGnNahyr7+sAaSI3a5ECQLUjwaIARUCyoRh7EFK3qPXcg=="
  }
}
```

Argumente

<CLUSTER_ID>

Die ID des Clusters, auf dem dieser Vorgang ausgeführt werden soll.

Erforderlich: Wenn mehrere Cluster [konfiguriert wurden](#).

<PAYLOAD_FILTER>

Schlüsselreferenz (z. B. `key-reference=0xabc`) oder durch Leerzeichen getrennte Liste von Schlüsselattributen in der Form „`attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE`“ Zur Auswahl eines Payload-Schlüssels“.

Erforderlich: Ja

<PATH>

Pfad zur Binärdatei, in der die verpackten Schlüsseldaten gespeichert werden.

Erforderlich: Nein

<WRAPPING_FILTER>

Schlüsselreferenz (z. B. `key-reference=0xabc`) oder durch Leerzeichen getrennte Liste von Schlüsselattributen in der Form „`attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE`“ Zur Auswahl eines Umbruchschlüssels“.

Erforderlich: Ja

<WRAPPING_APPROVALR>

Gibt den Dateipfad zu einer signierten Quorum-Token-Datei an, um den Vorgang zum Umschließen des Schlüssels zu genehmigen. Nur erforderlich, wenn der Quorumwert für den Schlüsselverwaltungsdienst beim Umschließen des Schlüssels größer als 1 ist.

<PAYLOAD_APPROVALR>

Gibt den Dateipfad zu einer signierten Quorum-Token-Datei an, um den Vorgang für den Payload-Schlüssel zu genehmigen. Nur erforderlich, wenn der Quorumwert für den Schlüsselverwaltungsdienst des Payload-Schlüssels größer als 1 ist.

Verwandte Themen

- [Der Key Wrap-Befehl in der CloudHSM-CLI](#)
- [Der Befehl zum Entpacken von Schlüsseln in der CloudHSM-CLI](#)

Melden Sie sich mit der CloudHSM-CLI bei einem HSM an

Sie können den login Befehl in der CloudHSM-CLI verwenden, um sich bei jedem Hardware Security (HSM) in einem Cluster an- und abzumelden. AWS CloudHSM Dieser Befehl hat den folgenden Unterbefehl:

- [mfa-token-sign](#)

Note

Wenn Sie mehr als fünf falsche Anmeldeversuche tätigen, wird Ihr Konto gesperrt. Um das Konto zu entsperren, muss ein Administrator Ihr Passwort mit dem Befehl [user change-password](#) in cloudhsm_cli zurücksetzen.

Um Probleme bei der An- und Abmeldung zu beheben

Wenn Sie mehr als ein HSM in Ihrem Cluster haben, sind Ihnen möglicherweise weitere falsche Anmeldeversuche gestattet, bevor Ihr Konto gesperrt wird. Dies liegt daran, dass der CloudHSM-Client die Last auf verschiedene verteilt. HSMs Aus diesem Grund beginnt der Anmeldeversuch möglicherweise nicht jedes Mal auf demselben HSM. Wenn Sie diese Funktion testen, empfehlen wir Ihnen, dies auf einem Cluster mit nur einem aktiven HSM zu tun.

Wenn Sie Ihren Cluster vor Februar 2018 erstellt haben, wird Ihr Konto nach 20 falschen Anmeldeversuchen gesperrt.

Benutzertyp

Die folgenden Benutzer können diese Befehle ausführen.

- Nicht aktivierter Admin
- Admin.
- Crypto-Benutzer (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')
```

Beispiel

Example

Mit diesem Befehl melden Sie sich bei all HSMs in einem Cluster mit den Anmeldeinformationen eines Admin-Benutzers mit dem Namen `admin1`

```
aws-cloudhsm > login --username admin1 --role admin
Enter password:
{
  "error_code": 0,
  "data": {
    "username": "admin1",
    "role": "admin"
  }
}
```

Argumente

<CLUSTER_ID>

Die ID des Clusters, auf dem dieser Vorgang ausgeführt werden soll.

Erforderlich: Wenn mehrere Cluster [konfiguriert wurden](#).

<USERNAME>

Gibt einen Anzeigenamen für den Benutzer an. Die maximale Länge beträgt 31 Zeichen. Das einzige zulässige Sonderzeichen ist ein Unterstrich (_). Beim Benutzernamen wird in diesem Befehl nicht zwischen Groß- und Kleinschreibung unterschieden, der Benutzername wird immer in Kleinbuchstaben angezeigt.

Erforderlich: Ja

<ROLE>

Gibt die diesem Benutzer zugewiesene Rolle an. Gültige Werte sind `admin`, `crypto-user`.

Verwenden Sie den `user list` Befehl, um die Rolle des Benutzers abzurufen. Ausführliche Informationen zu den Benutzertypen in einem HSM finden Sie unter [HSM-Benutzer verstehen](#).

Erforderlich: Ja

<PASSWORD>

Gibt das Passwort des Benutzers an, der sich bei der anmeldet HSMs.

Erforderlich: Kunden müssen das Passwort entweder über das `--password` Befehlszeilenargument angeben oder es weglassen, damit es interaktiv abgefragt wird.

Note

Wenn Sie das `--password` Argument in der Befehlszeile verwenden, werden möglicherweise Anmeldeinformationen in Ihrem Shell-Verlauf angezeigt. Erwägen Sie, das Passwort interaktiv einzugeben oder Ihren Shell-Verlauf nach der Verwendung zu löschen.

Verwandte Themen

- [Erste Schritte mit CloudHSM-CLI](#)
- [Aktivieren des Clusters](#)

Melden Sie sich mit MFA über die CloudHSM-CLI bei einem HSM an

Verwenden Sie den `login mfa-token-sign` Befehl in der AWS CloudHSM CloudHSM-CLI, um sich mithilfe der Multi-Faktor-Authentifizierung (MFA) bei einem Hardware-Sicherheitsmodul (HSM) anzumelden. Um diesen Befehl verwenden zu können, müssen Sie zuerst [MFA für CloudHSM-CLI](#) einrichten.

Benutzertyp

Die folgenden Benutzer können diese Befehle ausführen.

- Admin.
- Crypto-Benutzer (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
```

Beispiel

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"
  }
}
```

Argumente

<CLUSTER_ID>

Die ID des Clusters, auf dem dieser Vorgang ausgeführt werden soll.

Erforderlich: Wenn mehrere Cluster [konfiguriert wurden](#).

<TOKEN>

Dateipfad, in den die unsigned Tokendatei geschrieben wird.

Erforderlich: Ja

Verwandte Themen

- [Erste Schritte mit CloudHSM-CLI](#)
- [Aktivieren des Clusters](#)
- [Verwenden von CloudHSM-CLI zur Verwaltung von MFA](#)

Melden Sie sich mit der CloudHSM-CLI von einem HSM ab

Verwenden Sie den logout Befehl in der CloudHSM-CLI, um sich von jedem Hardware-Sicherheitsmodul (HSM) in einem Cluster abzumelden. AWS CloudHSM

Benutzertyp

Die folgenden Benutzer können diesen Befehl ausführen.

- Admin.
- Crypto-Benutzer (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
```

Beispiel

Example

Mit diesem Befehl werden Sie von allen Clustern abgemeldet HSMs .

```
aws-cloudhsm > logout
```



```
{
  "error_code": 0,
  "data": "Logout successful"
}
```

Verwandte Themen

- [Erste Schritte mit CloudHSM-CLI](#)
- [Aktivieren des Clusters](#)

Die Benutzerkategorie in CloudHSM CLI

In der CloudHSM-CLI user ist dies eine übergeordnete Kategorie für eine Gruppe von Befehlen, die in Kombination mit der übergeordneten Kategorie einen benutzerspezifischen Befehl erstellen. Derzeit besteht die Benutzerkategorie aus den folgenden Befehlen:

- [user change-mfa](#)
- [user change-password](#)
- [user create](#)
- [user delete](#)
- [user list](#)
- [Benutzer replizieren](#)

Die Kategorie „user change-mfa“ in der CloudHSM-CLI

In der CloudHSM-CLI user change-mfa ist dies eine übergeordnete Kategorie für eine Gruppe von Befehlen, die in Kombination mit der übergeordneten Kategorie einen Befehl erstellen, der speziell für die Änderung der Multi-Faktor-Authentifizierung (MFA) für Benutzer spezifisch ist.

Derzeit besteht diese Kategorie aus dem folgenden Unterbefehl:

- [token-sign](#)

Ändern Sie das MFA-Setup eines Benutzers mit der CloudHSM CLI

Verwenden Sie den user change-mfa token-sign Befehl in der CloudHSM-CLI, um die Einrichtung der Multi-Faktor-Authentifizierung (MFA) eines Benutzerkontos zu aktualisieren. Jedes Benutzerkonto

kann diesen Befehl ausführen. Konten mit der Administratorrolle können diesen Befehl für andere Benutzer ausführen.

Benutzertyp

Die folgenden Benutzer können diesen Befehl ausführen.

- Admin.
- Crypto-Benutzer

Syntax

Derzeit steht Benutzern nur eine einzige Multi-Faktor-Strategie zur Verfügung: Token Sign.

```
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)
```

Die Token-Sign-Strategie verlangt nach einer Token-Datei, in die unsignierte Token geschrieben werden sollen.

```
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'`)

Beispiel

Dieser Befehl schreibt ein unsigniertes Token pro HSM in Ihrem Cluster in die von token angegebene Datei. Signieren Sie die Token in der Datei, wenn Sie dazu aufgefordert werden.

Example: Schreiben Sie ein unsigniertes Token pro HSM in Ihr Cluster

```
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"  
}  
}
```

Argumente

<CLUSTER_ID>

Die ID des Clusters, auf dem dieser Vorgang ausgeführt werden soll.

Erforderlich: Wenn mehrere Cluster [konfiguriert wurden](#).

<ROLE>

Gibt die Rolle an, die dem Benutzerkonto zugewiesen wurde. Dieser Parameter muss angegeben werden. Ausführliche Informationen zu den Benutzertypen in einem HSM finden Sie unter [HSM-Benutzer verstehen](#).

Zulässige Werte

- Admin: Administratoren können Benutzer verwalten, aber sie können keine Schlüssel verwalten.
- Crypto-Benutzer: Crypto-Benutzer können Schlüssel erstellen und verwalten und Schlüssel in kryptografischen Vorgängen verwenden.

<USERNAME>

Gibt einen Anzeigenamen für den Benutzer an. Die maximale Länge beträgt 31 Zeichen. Das einzige zulässige Sonderzeichen ist ein Unterstrich (_).

Sie können den Namen eines Benutzers nach der Erstellung nicht mehr ändern. Bei CloudHSM-CLI-Befehlen wird bei der Rolle und dem Passwort zwischen Groß- und Kleinschreibung unterschieden, beim Benutzernamen jedoch nicht.

Erforderlich: Ja

<CHANGE_PASSWORD>

Gibt das neue Klartext-Passwort des Benutzers an, dessen MFA registriert/deregistriert wird.

Erforderlich: Ja

<TOKEN>

Dateipfad, in den die unsignierte Tokendatei geschrieben wird.

Erforderlich: Ja

<APPROVAL>

Gibt den Dateipfad zu einer signierten Quorum-Token-Datei an, um den Vorgang zu genehmigen. Nur erforderlich, wenn der Quorumwert des Quorum-Benutzerdienstes größer als 1 ist.

<DEREGISTER>

Deregistriert den öffentlichen MFA-Schlüssel, falls vorhanden.

<CHANGE-QUORUM>

Ändert den öffentlichen Quorum-Schlüssel zusammen mit dem MFA-Schlüssel.

Verwandte Themen

- [2FA für HSM-Benutzer verstehen](#)

Ändern Sie das Passwort eines Benutzers mit CloudHSM CLI

Verwenden Sie den `user change-password` Befehl in der CloudHSM-CLI, um das Passwort eines vorhandenen Benutzers in Ihrem AWS CloudHSM Cluster zu ändern. Verwenden Sie den `user change-mfa`-Befehl, um MFA für einen Benutzer zu aktivieren.

Jeder Benutzer kann das eigene Passwort ändern. Darüber hinaus können Benutzer mit der Administratorrolle das Passwort eines anderen Benutzers im Cluster ändern. Sie müssen das aktuelle Passwort nicht eingeben, um die Änderung vorzunehmen.

Note

Sie können jedoch nicht das Passwort eines Benutzers ändern, der beim Cluster oder angemeldet ist.

Benutzertyp

Die folgenden Benutzer können diesen Befehl ausführen.

- Admin.
- Crypto-Benutzer (Crypto User, CU)

Syntax

Note

Verwenden Sie den Befehl, um die Multi-Faktor-Authentifizierung (MFA) für einen Benutzer zu aktivieren. `user change-mfa`

```
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')
```

Beispiel

Die folgenden Beispiele zeigen, wie Sie mit `user change-password` das Passwort für den aktuellen Benutzer oder einen anderen Benutzer in Ihrem Cluster zurücksetzen können.

Example: Ändert Ihr Passwort

Jeder Benutzer im Cluster kann mit `user change-password` sein eigenes Passwort ändern.

Die folgende Ausgabe zeigt, dass Bob derzeit als CU angemeldet ist.

```
aws-cloudhsm > user change-password --username bob --role crypto-user
Enter password:
Confirm password:
{
  "error_code": 0,
  "data": {
    "username": "bob",
    "role": "crypto-user"
  }
}
```

Argumente

<CLUSTER_ID>

Die ID des Clusters, auf dem dieser Vorgang ausgeführt werden soll.

Erforderlich: Wenn mehrere Cluster [konfiguriert wurden](#).

<APPROVAL>

Gibt den Dateipfad zu einer signierten Quorum-Token-Datei an, um den Vorgang zu genehmigen.
Nur erforderlich, wenn der Quorumwert des Quorum-Benutzerdienstes größer als 1 ist.

<DEREGISTER-MFA>

Deregistriert den öffentlichen MFA-Schlüssel, falls vorhanden.

<DEREGISTER-QUORUM>

Zum Deregistrieren des öffentlichen Quorum-Schlüssels, falls vorhanden.

<PASSWORD>

Gibt das neue Klartext-Passwort des Benutzers an. Die folgenden Zeichen ':' sind nicht zulässig

Erforderlich: Ja

<ROLE>

Gibt die Rolle an, die dem Benutzerkonto zugewiesen wurde. Dieser Parameter muss angegeben werden. Ausführliche Informationen zu den Benutzertypen in einem HSM finden Sie unter [HSM-Benutzer verstehen](#).

Zulässige Werte

- Admin: Administratoren können Benutzer verwalten, aber sie können keine Schlüssel verwalten.
- Crypto-Benutzer: Crypto-Benutzer können Schlüssel erstellen und verwalten und Schlüssel in kryptografischen Vorgängen verwenden.

<USERNAME>

Gibt einen Anzeigenamen für den Benutzer an. Die maximale Länge beträgt 31 Zeichen. Das einzige zulässige Sonderzeichen ist ein Unterstrich (_).

Sie können den Namen eines Benutzers nach der Erstellung nicht mehr ändern. Bei CloudHSM-CLI-Befehlen wird bei der Rolle und dem Passwort zwischen Groß- und Kleinschreibung unterschieden, beim Benutzernamen jedoch nicht.

Erforderlich: Ja

Verwandte Themen

- [user list](#)
- [user create](#)
- [user delete](#)

Die Kategorie „Benutzeränderungsquorum“ in der CloudHSM-CLI

In der CloudHSM-CLI user change-quorum ist dies eine übergeordnete Kategorie für eine Gruppe von Befehlen, die in Kombination mit der übergeordneten Kategorie einen Befehl erstellen, der spezifisch für die Änderung des Quorums für Benutzer ist.

user change-quorum wird verwendet, um die Benutzerquorumauthentifizierung mithilfe einer bestimmten Quorumstrategie zu registrieren. Ab SDK 5.8.0 steht Benutzern nur eine einzige Quorumstrategie zur Verfügung, wie unten dargestellt.

Derzeit besteht diese Kategorie aus der folgenden Kategorie und dem folgenden Unterbefehl:

- [token-sign](#)
 - [register](#)

Die Kategorie Change-Quorum-Token-Sign-Zeichen für Benutzer in der CloudHSM-CLI

In der CloudHSM-CLI user change-quorum token-sign ist dies eine übergeordnete Kategorie für Befehle, die in Kombination mit dieser übergeordneten Kategorie einen Befehl erstellen, der spezifisch für Tokensign-Quorumoperationen ist.

Derzeit besteht diese Kategorie aus den folgenden Befehlen:

- [register](#)

Registrieren Sie die Token-Sign-Quorum-Strategie eines Benutzers mithilfe der CloudHSM-CLI

Verwenden Sie den user change-quorum token-sign register-Befehl in der CloudHSM-CLI, um die Token-Sign-Quorumstrategie für einen Admin-Benutzer zu registrieren.

Benutzertyp

Die folgenden Benutzer können diesen Befehl ausführen.

- Admin.

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')

```

Beispiel

Example

Um diesen Befehl auszuführen, müssen Sie als der Benutzer angemeldet sein, für den Sie register quorum token-sign wollen.

```

aws-cloudhsm > login --username admin1 --role admin
Enter password:
{
  "error_code": 0,
  "data": {
    "username": "admin1",
    "role": "admin"
  }
}

```

Der user change-quorum token-sign register-Befehl registriert Ihren öffentlichen Schlüssel beim HSM. Dadurch qualifizieren Sie sich als Quorum-Genehmiger für Quorum-Operationen, bei denen ein Benutzer Quorumsignaturen einholen muss, um den erforderlichen Quorumschwellenwert zu erreichen.

```

aws-cloudhsm > user change-quorum token-sign register \
  --public-key /home/mypemfile \
  --signed-token /home/mysignedtoken
{
  "error_code": 0,
  "data": {
    "username": "admin1",
    "role": "admin"
  }
}

```

Sie können jetzt den user list-Befehl ausführen und bestätigen, dass das Quorum-token-sign für diesen Benutzer registriert wurde.

```
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"
      }
    ]
  }
}
```

Argumente

<CLUSTER_ID>

Die ID des Clusters, auf dem dieser Vorgang ausgeführt werden soll.

Erforderlich: Wenn mehrere Cluster [konfiguriert wurden](#).

<PUBLIC-KEY>

Dateipfad zur PEM-Datei mit öffentlichem Schlüssel.

Erforderlich: Ja

<SIGNED-TOKEN>

Dateipfad mit Token, das mit dem privaten Schlüssel des Benutzers signiert wurde.

Erforderlich: Ja

Verwandte Themen

- [Verwenden der CloudHSM-CLI zur Verwaltung der Quorum-Authentifizierung](#)
- [Quorum-Authentifizierung für Admins verwenden: erstmalige Einrichtung](#)
- [Ändern Sie den Quorum-Mindestwert für Administratoren](#)
- [Namen und Typen von Diensten, die die Quorum-Authentifizierung unterstützen](#)

Einen AWS CloudHSM Benutzer mit CloudHSM CLI erstellen

Der `user create` Befehl in der CloudHSM-CLI erstellt einen Benutzer in Ihrem AWS CloudHSM Cluster. Nur Benutzerkonten mit der Administratorrolle können diesen Befehl ausführen.

Benutzertyp

Die folgenden Benutzertypen können diesen Befehl ausführen.

- Admin.

Voraussetzungen

Um diesen Befehl auszuführen, müssen Sie als Admin-Benutzer angemeldet sein

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')
```

Beispiel

Diese Beispiele zeigen, wie Sie user create neue Benutzer in Ihrem erstellen können. HSMs

Example: Erstellen eines Crypto-Benutzers

In diesem Beispiel wird in Ihrem AWS CloudHSM Cluster ein Konto mit der Crypto-Benutzerrolle erstellt.

```
aws-cloudhsm > user create --username alice --role crypto-user
Enter password:
Confirm password:
{
  "error_code": 0,
  "data": {
    "username": "alice",
    "role": "crypto-user"
  }
}
```

Argumente

<CLUSTER_ID>

Die ID des Clusters, auf dem dieser Vorgang ausgeführt werden soll.

Erforderlich: Wenn mehrere Cluster [konfiguriert wurden](#).

<USERNAME>

Gibt einen Anzeigenamen für den Benutzer an. Die maximale Länge beträgt 31 Zeichen. Das einzige zulässige Sonderzeichen ist ein Unterstrich (_). Beim Benutzernamen wird in diesem Befehl nicht zwischen Groß- und Kleinschreibung unterschieden, der Benutzername wird immer in Kleinbuchstaben angezeigt.

Erforderlich: Ja

<ROLE>

Gibt die diesem Benutzer zugewiesene Rolle an. Dieser Parameter muss angegeben werden. Gültige Werte sind admin, crypto-user.

Verwenden Sie den user list-Befehl, um die Rolle des Benutzers abzurufen. Ausführliche Informationen zu den Benutzertypen in einem HSM finden Sie unter [HSM-Benutzer verstehen](#).

<PASSWORD>

Gibt das Passwort des Benutzers an, der sich bei der anmeldet HSMs. Die folgenden Zeichen sind nicht zulässig ':'

Erforderlich: Ja

<APPROVAL>

Gibt den Dateipfad zu einer signierten Quorum-Token-Datei an, um den Vorgang zu genehmigen. Nur erforderlich, wenn der Quorumwert des Quorum-Benutzerdienstes größer als 1 ist.

Verwandte Themen

- [user list](#)
- [user delete](#)
- [user change-password](#)

Löschen Sie einen AWS CloudHSM Benutzer mit CloudHSM CLI

Der `user delete` Befehl in der CloudHSM-CLI löscht einen Benutzer aus Ihrem Cluster. AWS CloudHSM Nur Benutzerkonten mit der Administratorrolle dürfen diesen Befehl ausführen. Sie können keinen Benutzer löschen, der derzeit bei einem HSM angemeldet ist.

Benutzertyp

Die folgenden Benutzertypen können diesen Befehl ausführen.

- Admin.

Voraussetzungen

- Sie können keine Benutzerkonten löschen, die Schlüssel besitzen.
- Ihr Benutzerkonto muss über die Administratorrolle verfügen, um diesen Befehl ausführen zu können.

Syntax

Da dieser Befehl keine benannten Parameter hat, müssen Sie die Argumente in der im Syntaxdiagramm angegebenen Reihenfolge eingeben.

```
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

Beispiel

```
aws-cloudhsm > user delete --username alice --role crypto-user
{
  "error_code": 0,
  "data": {
    "username": "alice",
    "role": "crypto-user"
  }
}
```

Argumente

<CLUSTER_ID>

Die ID des Clusters, auf dem dieser Vorgang ausgeführt werden soll.

Erforderlich: Wenn mehrere Cluster [konfiguriert wurden](#).

<USERNAME>

Gibt einen Anzeigenamen für den Benutzer an. Die maximale Länge beträgt 31 Zeichen. Das einzige zulässige Sonderzeichen ist ein Unterstrich (_). Beim Benutzernamen wird in diesem Befehl nicht zwischen Groß- und Kleinschreibung unterschieden, der Benutzername wird immer in Kleinbuchstaben angezeigt.

Erforderlich: Ja

<ROLE>

Gibt die diesem Benutzer zugewiesene Rolle an. Dieser Parameter muss angegeben werden. Gültige Werte sind admin, crypto-user.

Verwenden Sie den user list-Befehl, um die Rolle des Benutzers abzurufen. Ausführliche Informationen zu den Benutzertypen in einem HSM finden Sie unter [HSM-Benutzer verstehen](#).

Erforderlich: Ja

<APPROVAL>

Gibt den Dateipfad zu einer signierten Quorum-Token-Datei an, um den Vorgang zu genehmigen. Nur erforderlich, wenn der Quorumwert des Quorum-Benutzerdienstes größer als 1 ist.

Erforderlich: Ja

Verwandte Themen

- [user list](#)
- [user create](#)
- [user change-password](#)

Alle AWS CloudHSM Benutzer mit CloudHSM CLI auflisten

Der `user list` Befehl in der CloudHSM-CLI listet die Benutzerkonten auf, die in Ihrem AWS CloudHSM Cluster vorhanden sind. Sie müssen nicht bei CloudHSM-CLI angemeldet sein, um diesen Befehl auszuführen.

Note

Wenn Sie hinzufügen oder löschen HSMs, aktualisieren Sie die Konfigurationsdateien, die der AWS CloudHSM Client und die Befehlszeilentools verwenden. Andernfalls sind die Änderungen, die Sie vornehmen, möglicherweise nicht für alle Mitglieder HSMs des Clusters wirksam.

Benutzertyp

Die folgenden Benutzertypen können diesen Befehl ausführen.

- Alle Benutzer. Sie müssen nicht angemeldet sein, um diesen Befehl auszuführen.

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
```

Beispiel

Dieser Befehl listet die Benutzer auf, die in Ihrem CloudHSM-Cluster vorhanden sind.

```
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"
      }
    ]
  }
}
```

```
]
}
}
```

Die Ausgabe umfasst die folgenden Benutzerattribute:

- **Benutzername:** Zeigt den benutzerdefinierten Anzeigenamen für den Benutzer an. Der Benutzername wird immer in Kleinbuchstaben angezeigt.
- **Benutzertyp:** Bestimmt die Operationen, die der Benutzer im HSM ausführen kann.
- **Gesperrt:** Zeigt an, ob dieses Benutzerkonto gesperrt wurde.
- **MFA:** Gibt die unterstützten Multi-Faktor-Authentifizierungsmechanismen für dieses Benutzerkonto an.
- **Clusterabdeckung:** Zeigt die clusterweite Verfügbarkeit dieses Benutzerkontos an.

Verwandte Themen

- [listUsers](#) in key_mgmt_util
- [user create](#)
- [user delete](#)
- [user change-password](#)

Replizieren Sie einen Benutzer mit CloudHSM CLI

Verwenden Sie den `user replicate` Befehl in der CloudHSM-CLI, um einen Benutzer von einem Quell-Cluster auf einen AWS CloudHSM Ziel-Cluster zu replizieren. AWS CloudHSM

Benutzertyp

Die folgenden Benutzertypen können diesen Befehl ausführen.

- Administratoren () COs

Voraussetzungen

- Bei den Quell- und Zielclustern muss es sich um Klone handeln. Das bedeutet, dass einer aus einem Backup des anderen erstellt wurde oder dass beide aus einem gemeinsamen Backup erstellt wurden. Weitere Informationen finden Sie unter [Cluster aus Sicherungen erstellen](#).

- Um diesen Befehl ausführen zu können, müssen Sie sowohl auf dem Quell- als auch auf dem Zielcluster als Administrator angemeldet sein.
- Im Einzelbefehlsmodus verwendet der Befehl die Umgebungsvariablen CLOUDHSM_PIN und CLOUDHSM_ROLE, um sich auf dem Quellcluster zu authentifizieren. Weitere Informationen finden Sie unter [Einzelbefehlsmodus](#). Um Anmeldeinformationen für den Zielcluster bereitzustellen, müssen Sie zwei zusätzliche Umgebungsvariablen festlegen: DESTINATION_CLOUDHSM_PIN und DESTINATION_CLOUDHSM_ROLE:

```
$ export DESTINATION_CLOUDHSM_ROLE=<role>
```

```
$ export DESTINATION_CLOUDHSM_PIN=<username:password>
```

- Im interaktiven Modus müssen sich Benutzer explizit sowohl beim Quell- als auch beim Zielcluster anmelden.

Syntax

```
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')
```

Beispiele

Example Beispiel: Benutzer replizieren

Dieser Befehl repliziert einen Benutzer von einem Quellcluster mit auf einen geklonten Zielcluster. Das folgende Beispiel zeigt die Ausgabe, wenn Sie auf beiden Clustern als Administrator angemeldet sind.

```
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"
  }
}
```

Argumente

<USERNAME>

Gibt den Benutzernamen des Benutzers an, der im Quellcluster repliziert werden soll.

Erforderlich: Ja

<ROLE>

Gibt die diesem Benutzer zugewiesene Rolle an. Dieser Parameter muss angegeben werden. Gültige Werte sind admin, crypto-user.

Verwenden Sie den `user list`-Befehl, um die Rolle des Benutzers abzurufen. Ausführliche Informationen zu den Benutzertypen in einem HSM finden Sie unter [HSM-Benutzer verstehen](#).

Erforderlich: Ja

<SOURCE_CLUSTER_ID>

Die Quellcluster-ID.

Erforderlich: Ja

<DESTINATION_CLUSTER_ID>

Die Zielcluster-ID.

Erforderlich: Ja

Verwandte Themen

- [Verbindung zu mehreren Clustern mit CloudHSM CLI herstellen](#)

Die Quorum-Kategorie in CloudHSM CLI

In der CloudHSM-CLI `quorum` ist dies eine übergeordnete Kategorie für eine Gruppe von Befehlen, die in Kombination mit `quorum` einem Befehl speziell für die Quorumauthentifizierung oder M-of-N-Operationen erstellt werden. Derzeit besteht diese Kategorie aus der `token-sign`-Unterkategorie, die aus eigenen Befehlen besteht. Weitere Informationen finden Sie unter dem nachfolgenden Link.

- [token-sign](#)

Admin-Dienste: Die Quorum-Authentifizierung wird für Dienste mit Administratorrechten verwendet, z. B. für das Erstellen von Benutzern, das Löschen von Benutzern, das Ändern von Benutzerkennwörtern, das Festlegen von Quorumwerten und das Deaktivieren von Quorum- und MFA-Funktionen.

Crypto User Services: Die Quorum-Authentifizierung wird für privilegierte Dienste verwendet, die Krypto-Benutzern vorbehalten sind und mit einem bestimmten Schlüssel verknüpft sind, z. B. das Signieren mit einem Schlüssel, `sharing/unsharing` einem Schlüssel, einem Schlüssel und das `wrapping/unwrapping` Festlegen eines Schlüsselattributs. Der Quorumwert eines zugehörigen Schlüssels wird konfiguriert, wenn der Schlüssel generiert, importiert oder entpackt wird. Der Quorumwert muss gleich oder kleiner als die Anzahl der Benutzer sein, denen der

Schlüssel zugeordnet ist. Dazu gehören Benutzer, mit denen der Schlüssel geteilt wird, und der Schlüsselbesitzer.

Jeder Dienstyp wird weiter in einen qualifizierenden Dienstnamen unterteilt, der eine bestimmte Gruppe von Quorum-unterstützten Dienstvorgängen enthält, die ausgeführt werden können.

Service-Name	Servicetyp	Serviceoperationen
user	Admin.	• user create • user delete • user change-password • user change-mfa
quorum	Admin.	• Quorum-Tokenzeichen set-quorum-value
Cluster 1	Admin.	• Cluster MTLS register-trust-anchor • Cluster-MTLS deregister-trust-anchor • Durchsetzung von Cluster-M TLS-Sätzen
Schlüsselverwaltung	Crypto-Benutzer	• Schlüsselhülle • Schlüssel auspacken • key share • key unshare • key set-attribute
Verwendung des Schlüssels	Crypto-Benutzer	• Schlüsselzeichen

[1] Der Cluster-Service ist ausschließlich auf hsm2m.medium verfügbar

Verwandte Themen

- [Richten Sie die Quorum-Authentifizierung für AWS CloudHSM Administratoren mithilfe der CloudHSM-CLI ein](#)

- [Verwaltung der Quorum-Authentifizierung \(M-of-N-Zugriffskontrolle\) mithilfe der CloudHSM-CLI](#)

Die Quorum-Tokensign-Kategorie in der CloudHSM-CLI

In der CloudHSM-CLI quorum token-sign ist dies eine Kategorie für eine Gruppe von Befehlen, die in Kombination mit quorum token-sign einem Befehl einen spezifischen Befehl für die Quorumauthentifizierung oder M-of-N-Operationen erstellen.

Derzeit besteht diese Kategorie aus den folgenden Befehlen:

- [delete](#)
- [generate](#)
- [list](#)
- [list-quorum-values](#)
- [set-quorum-value](#)

Quorum-Token mithilfe der CloudHSM-CLI löschen

Verwenden Sie den quorum token-sign delete-Befehl in der CloudHSM-CLI, um ein oder mehrere Token für einen vom Quorum autorisierten Dienst zu löschen.

Benutzertyp

Die folgenden Benutzer können diesen Befehl ausführen.

- Admin.

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')
```

Beispiel

Das folgende Beispiel zeigt, wie der `quorum token-sign delete`-Befehl in der CloudHSM-CLI verwendet werden kann, um ein oder mehrere Token für einen vom Quorum autorisierten Dienst zu löschen.

Example: Löschen Sie ein oder mehrere Token für einen vom Quorum autorisierten Dienst

```
aws-cloudhsm > quorum token-sign delete --scope all
{
  "error_code": 0,
  "data": "Deletion of quorum token(s) successful"
}
```

Argumente

<CLUSTER_ID>

Die ID des Clusters, auf dem dieser Vorgang ausgeführt werden soll.

Erforderlich: Wenn mehrere Cluster [konfiguriert wurden](#).

<SCOPE>

Der Bereich, in dem Token im AWS CloudHSM Cluster gelöscht werden.

Zulässige Werte

- Benutzer: Wird verwendet, um nur Token zu löschen, die dem angemeldeten Benutzer gehören.
- Alle: Wird verwendet, um alle Token im AWS CloudHSM Cluster zu löschen.

Verwandte Themen

- [user list](#)

- [user create](#)
- [user delete](#)

Generieren Sie ein Quorum-Token mit der CloudHSM-CLI

Verwenden Sie den `quorum token-sign generate`-Befehl in der CloudHSM-CLI, um ein Token für einen vom Quorum autorisierten Dienst zu generieren.

In einem HSM-Cluster für Dienstbenutzer und Quorum gibt es eine Obergrenze für den Erhalt eines aktiven Tokens pro Benutzer und Dienst. Dieses Limit gilt nicht für Token, die sich auf wichtige Dienste beziehen.

 Note

Nur Admins und Crypto-Benutzer dürfen bestimmte Service-Token generieren. Weitere Informationen zu Diensttypen und -namen finden Sie unter [Dienstnamen und Typen, die die Quorum-Authentifizierung unterstützen](#)

Admin-Dienste: Die Quorum-Authentifizierung wird für Dienste mit Administratorrechten verwendet, z. B. für das Erstellen von Benutzern, das Löschen von Benutzern, das Ändern von Benutzerkennwörtern, das Festlegen von Quorumwerten und das Deaktivieren von Quorum- und MFA-Funktionen.

Crypto User Services: Die Quorum-Authentifizierung wird für Dienste verwendet, für die Krypto-Benutzer privilegiert sind und mit einem bestimmten Schlüssel verknüpft sind, z. B. das Signieren mit `sharing/unsharing` einem Schlüssel, `wrapping/unwrapping` einem Schlüssel und das Festlegen eines Schlüsselattributs. Der Quorumwert eines zugehörigen Schlüssels wird konfiguriert, wenn der Schlüssel generiert, importiert oder entpackt wird. Der Quorumwert muss gleich oder kleiner als die Anzahl der Benutzer sein, denen der Schlüssel zugeordnet ist. Dazu gehören Benutzer, mit denen der Schlüssel geteilt wird, und der Schlüsselbesitzer.

Jeder Dienstyp wird weiter in einen qualifizierenden Dienstnamen unterteilt, der eine bestimmte Gruppe von Quorum-unterstützten Dienstvorgängen enthält, die ausgeführt werden können.

Service-Name	Servicetyp	Serviceoperationen
user	Admin.	• user create

Service-Name	Servicetyp	Serviceoperationen
		• user delete • user change-password • user change-mfa
quorum	Admin.	• Quorum-Tokenzeichen set-quorum-value
Cluster 1	Admin.	• Cluster MTLS register-trust-anchor • Cluster-MTLS deregister-trust-anchor • Durchsetzung von Cluster-M TLS-Sätzen
Schlüsselverwaltung	Crypto-Benutzer	• Schlüsselhülle • Schlüssel auspacken • key share • key unshare • key set-attribute
Verwendung des Schlüssels	Crypto-Benutzer	• Schlüsselzeichen

[1] Der Cluster-Service ist ausschließlich auf hsm2m.medium verfügbar

Benutzertyp

Die folgenden Benutzer können diesen Befehl ausführen.

- Admin.
- Crypto-Benutzer (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

Beispiel

Dieser Befehl schreibt ein unsigniertes Token pro HSM in Ihrem Cluster in die von token angegebene Datei.

Example: Schreiben Sie ein unsigniertes Token pro HSM in Ihr Cluster

```
aws-cloudhsm > quorum token-sign generate --service user --token /home/tfile
{
  "error_code": 0,
  "data": {
    "filepath": "/home/tfile"
  }
}
```

Argumente

<CLUSTER_ID>

Die ID des Clusters, auf dem dieser Vorgang ausgeführt werden soll.

Erforderlich: Wenn mehrere Cluster [konfiguriert wurden](#).

<SERVICE>

Gibt den vom Quorum autorisierten Dienst an, für den ein Token generiert werden soll. Dieser Parameter muss angegeben werden.

Zulässige Werte

- Benutzer: Der Benutzerverwaltungsdienst, der für die Ausführung von Vorgängen zur Verwaltung autorisierter Quorum-Benutzer verwendet wird.
- Quorum: Der Quorumverwaltungsdienst, der zum Festlegen von Quorum-autorisierten Quorumwerten für jeden vom Quorum autorisierten Dienst verwendet wird.
- Cluster: Der Clusterverwaltungsdienst, der für die Ausführung von Quorum für clusterweites Konfigurationsmanagement wie MTLS-Durchsetzung, MTLS-Registrierung und MTLS-Deregistrierung verwendet wird.
- Registrierung: Generiert ein unsigniertes Token zur Registrierung eines öffentlichen Schlüssels für die Quorumautorisierung.
- Schlüsselverwendung: Generiert ein unsigniertes Token, das für die Ausführung von Quorum-autorisierten Schlüsselverwendungsvorgängen verwendet wird.
- Schlüsselverwaltung: Generiert ein unsigniertes Token, das für die Ausführung von vom Quorum autorisierten Schlüsselverwaltungsoperationen verwendet wird.

Erforderlich: Ja

<TOKEN>

Dateipfad, in den die unsignierte Tokendatei geschrieben wird.

Erforderlich: Ja

Verwandte Themen

- [Namen und Typen von Diensten, die die Quorum-Authentifizierung unterstützen](#)

Quorum-Token mithilfe der CloudHSM-CLI auflisten

Verwenden Sie den `quorum token-sign list` Befehl in der CloudHSM-CLI, um alle Tokensign-Quorum-Token aufzulisten, die in Ihrem Cluster vorhanden sind. AWS CloudHSM Dazu gehören auch Token, die von anderen Benutzern generiert wurden. Ein Token ist an einen Benutzer gebunden. Sie können also möglicherweise Token von anderen Benutzern sehen, Sie können jedoch nur Token verwenden, die dem aktuell angemeldeten Benutzer zugeordnet sind.

Weitere Informationen zu Diensttypen und -namen finden Sie unter [Dienstnamen und -typen, die die Quorumauthentifizierung unterstützen](#). Weitere Informationen zu den Inhalten, die anhand der aufgelisteten Token angezeigt werden, finden Sie unter [the section called “Schlüsselverwaltung und -nutzung mit Quorum \(M von N\)”](#) Für Tokens, die mit key-management und key-usage Diensten verknüpft sind, bzw. unter [the section called “Benutzerverwaltung mit Quorum \(M von N\)”](#) Für Tokensuser, die mit `quorum`, oder dem `cluster` Dienst verknüpft sind.

Benutzertyp

Die folgenden Benutzer können diesen Befehl ausführen.

- Admin.
- Crypto-Benutzer (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
```

Beispiel

Dieser Befehl listet alle Tokensign-Token auf, die in Ihrem Cluster vorhanden sind. AWS CloudHSM Dies schließt Token ein, die von anderen Benutzern generiert wurden. Ein Token ist an einen Benutzer gebunden. Sie können also möglicherweise Token von anderen Benutzern sehen, Sie können jedoch nur Token verwenden, die dem aktuell angemeldeten Benutzer zugeordnet sind.

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": "0x0000000000002c33f7",
        "minimum-token-count": 1
      },
      {
```

```

    "username": "crypto_user1",
    "service": "key-usage",
    "key-reference": "0x000000000002c33f7",
    "minimum-token-count": 1
  }
]
}
}

```

Verwandte Themen

- [quorum token-sign generate](#)

Quorumwerte mit CloudHSM CLI anzeigen

Verwenden Sie den `quorum token-sign list-quorum-values` Befehl in der CloudHSM-CLI, um die in Ihrem Cluster festgelegten Quorumwerte aufzulisten. AWS CloudHSM

Benutzertyp

Die folgenden Benutzer können diesen Befehl ausführen.

- Alle Benutzer. Sie müssen nicht angemeldet sein, um diesen Befehl auszuführen.

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

```

Beispiel

Dieser Befehl listet die in Ihrem AWS CloudHSM Cluster für jeden Service festgelegten Quorumwerte auf.

Example

hsm1.medium:

```
aws-cloudhsm > quorum token-sign list-quorum-values
{
  "error_code": 0,
  "data": {
    "user": 1,
    "quorum": 1
  }
}
```

hsm2m. Mittel:

```
aws-cloudhsm > quorum token-sign list-quorum-values
{
  "error_code": 0,
  "data": {
    "user": 1,
    "quorum": 1,
    "cluster": 1
  }
}
```

Verwandte Themen

- [Namen und Typen von Diensten, die die Quorum-Authentifizierung unterstützen](#)
- [mTLS einrichten \(empfohlen\)](#)

Einen Quorumwert mithilfe der CloudHSM-CLI aktualisieren

Verwenden Sie den `quorum token-sign set-quorum-value`-Befehl in der CloudHSM-CLI, um einen neuen Quorumwert für einen autorisierten Quorum-Dienst festzulegen.

Benutzertyp

Die folgenden Benutzer können diesen Befehl ausführen.

- Admin.

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')

Beispiel

Example

Im folgenden Beispiel schreibt dieser Befehl ein unsigniertes Token pro HSM in Ihr Cluster in die durch das Token angegebene Datei. Signieren Sie die Token in der Datei, wenn Sie dazu aufgefordert werden.

```
aws-cloudhsm > quorum token-sign set-quorum-value --service quorum --value 2
{
  "error_code": 0,
  "data": "Set Quorum Value successful"
}
```

Anschließend können Sie den list-quorum-values-Befehl ausführen, um zu bestätigen, dass der Quorumwert für den Quorumverwaltungsdienst festgelegt wurde:

hsm1.medium:

```
aws-cloudhsm > quorum token-sign list-quorum-values
{
  "error_code": 0,
  "data": {
    "user": 1,
    "quorum": 2
  }
}
```

hsm2m. Mittel:

```
aws-cloudhsm > quorum token-sign list-quorum-values
{
  "error_code": 0,
  "data": {
    "user": 1,
    "quorum": 2,
    "cluster": 1
  }
}
```

Argumente

<CLUSTER_ID>

Die ID des Clusters, auf dem dieser Vorgang ausgeführt werden soll.

Erforderlich: Wenn mehrere Cluster [konfiguriert wurden](#).

<APPROVAL>

Der Dateipfad der signierten Tokendatei, die auf dem HSM genehmigt werden soll.

<SERVICE>

Gibt den vom Quorum autorisierten Dienst an, für den ein Token generiert werden soll. Dieser Parameter muss angegeben werden. Weitere Informationen zu Diensttypen und -namen finden Sie unter [Dienstnamen und -typen, die die Quorumauthentifizierung unterstützen](#).

Zulässige Werte

- Benutzer: Der Benutzerverwaltungsdienst. Dienst, der für die Ausführung autorisierter Quorum-Benutzerverwaltungsvorgänge verwendet wird.
- Quorum: Der Quorum-Verwaltungsdienst. Dienst, der zum Festlegen von Quorum-autorisierten Quorumwerten für jeden vom Quorum autorisierten Dienst verwendet wird.
- Cluster: Der Clusterverwaltungsdienst, der für die Ausführung von Quorum für clusterweites Konfigurationsmanagement wie MTLS-Durchsetzung, MTLS-Registrierung und MTLS-Deregistrierung verwendet wird.
- Registrierung: Generiert ein unsigniertes Token zur Registrierung eines öffentlichen Schlüssels für die Quorumautorisierung.

Erforderlich: Ja

<VALUE>

Gibt den Quorumwert an, der festgelegt werden soll. Der maximale Quorumwert ist acht (8).

Erforderlich: Ja

Verwandte Themen

- [Quorum-Tokenzeichen list-quorum-values](#)
- [Namen und Typen von Diensten, die die Quorum-Authentifizierung unterstützen](#)
- [mTLS einrichten \(empfohlen\)](#)

AWS CloudHSM Verwaltungsdienstprogramm (CMU)

Das Befehlszeilentool `cloudhsm_mgmt_util` hilft Kryptobeamten bei der Verwaltung von Benutzern in den Hardware-Sicherheitsmodulen (HSMs) in Clustern. Das AWS CloudHSM Management Utility (CMU) umfasst Tools zum Erstellen, Löschen und Auflisten von Benutzern sowie zum Ändern von Benutzerkennwörtern.

Die CMU und das Key Management Utility (KMU) sind Teil [der Client SDK 3-Suite](#). Client SDK 3 und die zugehörigen Befehlszeilentools (Key Management Utility und CloudHSM Management Utility) sind nur im HSM-Typ `hsm1.medium` verfügbar.

`cloudhsm_mgmt_util` enthält auch Befehle, die es Krypto-Benutzern ermöglichen, Schlüssel gemeinsam zu nutzen und Schlüsselattribute abzurufen und festzulegen. Diese Befehle ergänzen die Schlüsselverwaltungsbefehle im Primärschlüssel-Verwaltungstool, [key_mgmt_util](#).

Informationen zum schnellen Einstieg finden Sie unter [Erste Schritte mit AWS CloudHSM Management Utility \(CMU\)](#). Ausführliche Informationen über die `cloudhsm_mgmt_util`-Befehle und Beispiele für die Verwendung der Befehle finden Sie unter [Referenz für Befehle der AWS CloudHSM Management Utility](#).

Themen

- [Unterstützte Plattformen für AWS CloudHSM Management Utility](#)
- [Erste Schritte mit AWS CloudHSM Management Utility \(CMU\)](#)
- [Installieren und konfigurieren Sie den AWS CloudHSM Client für CMU \(Linux\)](#)
- [Installieren und konfigurieren Sie den AWS CloudHSM Client für CMU \(Windows\)](#)
- [Referenz für Befehle der AWS CloudHSM Management Utility](#)

Unterstützte Plattformen für AWS CloudHSM Management Utility

In diesem Thema werden die Linux- und Windows-Plattformen beschrieben, die von der AWS CloudHSM Management Utility (CMU) unterstützt werden.

Linux-Support

- 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-Unterstützung

- Microsoft Windows Server 2012
- Microsoft Windows Server 2012 R2
- Microsoft Windows Server 2016
- Microsoft Windows Server 2019

Erste Schritte mit AWS CloudHSM Management Utility (CMU)

AWS CloudHSM Mit dem Management Utility (CMU) können Sie Benutzer des Hardware Security Module (HSM) verwalten. Erfahren Sie in diesem Thema, um mit grundlegenden HSM-Benutzerverwaltungsaufgaben wie dem Erstellen von Benutzern, dem Auflisten von Benutzern und dem Verbinden der CMU mit dem Cluster zu beginnen.

1. Um CMU verwenden zu können, müssen Sie zunächst das Konfigurationstool verwenden, um die lokale CMU-Konfiguration mit dem `--cmu` Parameter und einer IP-Adresse von einer der HSMs Adressen in Ihrem Cluster zu aktualisieren. Tun Sie dies jedes Mal, wenn Sie CMU verwenden, um sicherzustellen, dass Sie HSM-Benutzer auf allen HSM im Cluster verwalten.

Linux

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

Windows

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

2. Verwenden Sie den folgenden Befehl, um die CLI im interaktiven Modus zu starten.

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
```

Je nachdem, wie viele HSMs Sie haben, sollte die Ausgabe wie folgt aussehen.

```
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.
```

Wenn `cloudhsm_mgmt_util` ausgeführt wird, ändert sich die Eingabeaufforderung in `aws-cloudhsm>`.

3. Verwenden Sie den `loginHSM`-Befehl, um sich beim Cluster anzumelden. Jeder Benutzertyp kann sich mit diesem Befehl beim Cluster anmelden.

Mit dem Befehl im folgenden Beispiel wird `admin` angemeldet, der Standard-[Verschlüsselungsverantwortliche \(CO, Crypto Officer\)](#). Sie legen das Passwort für diesen Benutzer nach der Aktivierung des Clusters fest. Sie können den `-hpswd`-Parameter verwenden, um Ihr Passwort zu verbergen.

```
aws-cloudhsm>loginHSM CO admin -hpswd
```

Sie werden vom System aufgefordert, Ihr Passwort einzugeben. Sie geben das Passwort ein, das System versteckt das Passwort und die Ausgabe zeigt, dass der Befehl erfolgreich war und dass Sie eine Verbindung zu allen HSMs im Cluster hergestellt haben.

```
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. Verwenden Sie `listUsers`, um alle Benutzer im Cluster aufzulisten.

```
aws-cloudhsm>listUsers
```

CMU listet alle Benutzer des Clusters auf.

```
Users on server 0(10.0.2.9):
```

```
Number of users found:2
```

User Id	User Type	User Name	
MofnPubKey	LoginFailureCnt	2FA	
1	C0	admin	N0
	0		N0
2	AU	app_user	N0
	0		N0

```
Users on server 1(10.0.3.11):
```

```
Number of users found:2
```

User Id	User Type	User Name	
MofnPubKey	LoginFailureCnt	2FA	
1	C0	admin	N0
	0		N0
2	AU	app_user	N0
	0		N0

```
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			
2		AU	app_user	NO
	0			

5. Verwenden Sie `createUser`, um einen CU-Benutzer namens **example_user** mit dem Passwort **password1** zu erstellen.

Sie verwenden CU-Benutzer in Ihren Anwendungen, um kryptografische und Schlüsselmanagementvorgänge durchzuführen. Sie können CU-Benutzer erstellen, da Sie sich in Schritt 3 als CO-Benutzer angemeldet haben. Nur CO-Benutzer können mit dem CMU Benutzerverwaltungsaufgaben durchführen, wie das Anlegen und Löschen von Benutzern und das Ändern der Passwörter anderer Benutzer.

```
aws-cloudhsm>createUser CU example_user password1
```

Die CMU fordert Sie auf, den Vorgang zum Erstellen eines Benutzers auszuführen.

```
*****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. Geben Sie **y** ein, um den CU-Benutzer **example_user** zu erstellen.
7. Verwenden Sie `listUsers`, um alle Benutzer im Cluster aufzulisten.

```
aws-cloudhsm>listUsers
```

CMU listet alle Benutzer im Cluster auf, einschließlich des neuen CU-Benutzers, den Sie gerade erstellt haben.

```
Users on server 0(10.0.2.9):
Number of users found:3
```

User Id	User Type	User Name	
MofnPubKey	LoginFailureCnt	2FA	
1	C0	admin	N0
	0		N0
2	AU	app_user	N0
	0		N0
3	CU	example_user	N0
	0		N0

Users on server 1(10.0.3.11):

Number of users found:3

User Id	User Type	User Name	
MofnPubKey	LoginFailureCnt	2FA	
1	C0	admin	N0
	0		N0
2	AU	app_user	N0
	0		N0
3	CU	example_user	N0
	0		N0

Users on server 2(10.0.1.12):

Number of users found:3

User Id	User Type	User Name	
MofnPubKey	LoginFailureCnt	2FA	
1	C0	admin	N0
	0		N0
2	AU	app_user	N0
	0		N0
3	CU	example_user	N0
	0		N0

- Verwenden Sie den `logoutHSM` Befehl, um sich von der HSMs abzumelden.

```
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)
```

- Verwenden Sie den `quit`-Befehl, um `cloudhsm_mgmt_util` zu beenden.

```
aws-cloudhsm>quit
```

```
disconnecting from servers, please wait...
```

Installieren und konfigurieren Sie den AWS CloudHSM Client für CMU (Linux)

Um mit dem Hardware-Sicherheitsmodul (HSM) in Ihrem AWS CloudHSM Cluster mithilfe von cloudhsm_mgmt_util (CMU) zu interagieren, benötigen Sie die Client-Software für Linux. AWS CloudHSM Sie sollten es auf der Amazon EC2 EC2-Linux-Client-Instance installieren, die Sie zuvor erstellt haben. Sie können auch einen Client für Windows installieren. Weitere Informationen finden Sie unter [Installieren und konfigurieren Sie den AWS CloudHSM Client für CMU \(Windows\)](#).

Aufgaben

- [Schritt 1. Installieren Sie den AWS CloudHSM Client und die Befehlszeilentools](#)
- [Schritt 2. Bearbeiten der Clientkonfiguration](#)

Schritt 1. Installieren Sie den AWS CloudHSM Client und die Befehlszeilentools

Connect zu Ihrer Client-Instance her und führen Sie die folgenden Befehle aus, um den AWS CloudHSM Client und die Befehlszeilentools herunterzuladen und zu installieren.

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
```

Schritt 2. Bearbeiten der Clientkonfiguration

Bevor Sie den AWS CloudHSM Client verwenden können, um eine Verbindung zu Ihrem Cluster herzustellen, müssen Sie die Client-Konfiguration bearbeiten.

Bearbeiten der Clientkonfiguration

1. Wenn Sie Client-SDK 3 auf cloudhsm_mgmt_util installieren, führen Sie die folgenden Schritte aus, um sicherzustellen, dass alle Knoten im Cluster synchronisiert sind.
 - a. Führen Sie `configure -a` **<IP of one of the HSMs>**.
 - b. Starten Sie den Client-Service neu.
 - c. Führen Sie `configure -m`.
2. Kopieren Sie Ihr ausstellendes Zertifikat – [das Zertifikat, mit dem Sie das Zertifikat des Clusters signiert haben](#) – an den folgenden Speicherort in der Client-Instance: `/opt/cloudhsm/etc/customerCA.crt`. Sie müssen Root-Berechtigungen für die Client-Instance haben, um Ihr Zertifikat an diesen Speicherort zu kopieren.
3. Verwenden Sie den folgenden [Konfigurationsbefehl](#), um die Konfigurationsdateien für den AWS CloudHSM Client und die Befehlszeilentools zu aktualisieren, und geben Sie dabei die IP-Adresse des HSM in Ihrem Cluster an. Um die IP-Adresse des HSM abzurufen, zeigen Sie Ihren Cluster in der [AWS CloudHSM Konsole](#) an oder führen Sie den [describe-clusters](#) AWS CLI Befehl aus. Die IP-Adresse des HSM ist in der Ausgabe des Befehls der Wert des Feldes `EniIp`.

Wenn Sie mehr als ein HSM haben, wählen Sie die IP-Adresse für eines der beiden HSMs; es spielt keine Rolle, welche.

```
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. Wechseln Sie zu [Aktiviere den Cluster in AWS CloudHSM](#).

Installieren und konfigurieren Sie den AWS CloudHSM Client für CMU (Windows)

Um mit einem Hardware-Sicherheitsmodul (HSM) in Ihrem AWS CloudHSM Cluster unter Windows unter Verwendung von cloudhsm_mgmt_util (CMU) arbeiten zu können, benötigen Sie die Client-Software für Windows. AWS CloudHSM Sie sollten die Software auf der Windows Server-Instance installieren, die Sie zuvor erstellt haben.

Note

- Bei der Aktualisierung des Clients werden vorhandene Konfigurationsdateien aus früheren Installationen nicht überschrieben.
- Das AWS CloudHSM Client-Installationsprogramm für Windows registriert automatisch die Cryptography API: Next Generation (CNG) und Key Storage Provider (KSP). Um den Client zu deinstallieren, führen Sie das Installationsprogramm erneut aus und folgen Sie den Anweisungen zur Deinstallation.
- Wenn Sie Linux verwenden, können Sie den Linux-Client installieren. Weitere Informationen finden Sie unter [Installieren und konfigurieren Sie den AWS CloudHSM Client für CMU \(Linux\)](#).

So installieren (oder aktualisieren) Sie die den neuesten Windows-Client und die Befehlszeilen-Tools

1. Stellen Sie eine Verbindung zur Windows Server-Instance her.
2. [Laden Sie das MSI-Installationsprogramm AWSCloudHSMClient-latest herunter](#).

3. Wenn Sie Client-SDK 3 auf cloudhsm_mgmt_util installieren, führen Sie die folgenden Schritte aus, um sicherzustellen, dass alle Knoten im Cluster synchronisiert sind.
 - a. Führen Sie `configure.exe -a <IP of one of the HSMs>`.
 - b. Starten Sie den Client-Service neu.
 - c. Führen Sie `configure.exe -m`.
4. Gehen Sie zu Ihrem Download-Speicherort und führen Sie das Installationsprogramm (AWSCloudHSMClient-latest.msi) mit Administratorrechten aus.
5. Folgen Sie den Anweisungen des Installationsprogramms und wählen Sie dann Schließen, wenn das Installationsprogramm abgeschlossen ist.
6. Kopieren Sie das selbstsignierte Ausstellerzertifikat – [das Zertifikat, mit dem Sie das Clusterzertifikat signiert haben](#) – in den Ordner `C:\ProgramData\Amazon\CloudHSM`.
7. Führen Sie den folgenden Befehl aus, um die Konfigurationsdateien zu aktualisieren. Beim Aktualisieren muss der Client während der Neukonfiguration angehalten und dann wieder gestartet werden.

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

8. Wechseln Sie zu [Aktiviere den Cluster in AWS CloudHSM](#).

Referenz für Befehle der AWS CloudHSM Management Utility

Das Befehlszeilentool AWS CloudHSM cloudhsm_mgmt_util hilft Kryptobeamten bei der Verwaltung von Benutzern in den Hardware-Sicherheitsmodulen () im Cluster. HSMs AWS CloudHSM Es enthält auch Befehle, die es Krypto-Benutzern (CUs) ermöglichen, Schlüssel gemeinsam zu nutzen und Schlüsselattribute abzurufen und festzulegen. Diese Befehle ergänzen die Primärschlüsselverwaltungsbefehle im [key_mgmt_util](#)-Befehlszeilen-Tool.

Informationen zum schnellen Einstieg finden Sie unter [Erste Schritte mit AWS CloudHSM Management Utility \(CMU\)](#).

Bevor Sie einen cloudhsm_mgmt_util-Befehl ausführen, müssen Sie cloudhsm_mgmt_util starten und sich beim HSM anmelden. Stellen Sie sicher, dass Sie sich mit einem Benutzerkontotyp anmelden, der die Befehle ausführen kann, die Sie verwenden möchten.

Um alle cloudhsm_mgmt_util-Befehle aufzuführen, führen Sie den folgenden Befehl aus:

```
aws-cloudhsm> help
```

Um die Syntax für einen cloudhsm_mgmt_util-Befehl anzufordern, führen Sie den folgenden Befehl aus:

```
aws-cloudhsm> help <command-name>
```

 Note

Verwenden Sie die Syntax gemäß der Dokumentation. Die integrierte Softwarehilfe bietet zwar zusätzliche Optionen, diese sollten jedoch nicht als unterstützt betrachtet und nicht im Produktionscode verwendet werden.

Um einen Befehl auszuführen, geben Sie den Namen des Befehls ein, oder einen Teil des Namens, der ausreicht, um ihn von den Namen anderer cloudhsm_mgmt_util-Befehle zu unterscheiden.

Um beispielsweise eine Liste der Benutzer auf dem zu erhalten HSMs, geben Sie listUsers oder einlistU.

```
aws-cloudhsm> listUsers
```

Um Ihre cloudhsm_mgmt_util-Sitzung zu beenden, führen Sie den folgenden Befehl aus:

```
aws-cloudhsm> quit
```

Hilfe zur Interpretation der Schlüsselattribute finden Sie unter [AWS CloudHSM Referenz zu Schlüsselattributen für KMU](#).

Die folgenden Themen beschreiben Befehle in cloudhsm_mgmt_util.

 Note

Einige Befehle in key_mgmt_util und cloudhsm_mgmt_util haben dieselben Namen. Die Befehle haben jedoch in der Regel eine andere Syntax, eine andere Ausgabe und eine leicht unterschiedliche Funktionalität.

Befehl	Description	Benutzertyp
<u>changePswd</u>	Ändert die Passwörter von Benutzern auf dem HSMs. Jeder Benutzer kann sein eigenes Passwort ändern. COs kann das Passwort von jedem ändern.	CO
<u>createUser</u>	Erzeugt Benutzer aller Typen auf dem HSMs.	CO
<u>deleteUser</u>	Löscht Benutzer aller Typen aus dem HSMs.	CO
<u>findAllKeys</u>	Ruft die Schlüssel ab, die ein Benutzer besitzt oder freigibt. Außerdem wird ein Hash der Schlüsseleigenschaft und Freigabedaten für alle Schlüssel in jedem HSM abgerufen.	CO, AU
<u>getAttribute</u>	Ruft einen Attributwert für einen AWS CloudHSM Schlüssel ab und schreibt ihn in eine Datei oder Standardausgabe (Standardausgabe).	CU
<u>getHSMInfo</u>	Ruft Informationen über die Hardware ab, auf der ein HSM ausgeführt wird.	Alle. Eine Anmeldung ist nicht erforderlich.
<u>getKeyInfo</u>	Ruft Eigentümer, gemeinsame Benutzer und den Quorum-Authentifizierungsstatus eines Schlüssels ab.	Alle. Eine Anmeldung ist nicht erforderlich.

Befehl	Description	Benutzertyp
info	Ruft Informationen zu einem HSM ab, einschließlich IP-Adresse, Hostname, Port und aktuellem Benutzer.	Alle. Eine Anmeldung ist nicht erforderlich.
listUsers	Ruft die Benutzer in den einzelnen Attributen HSMs, ihren Benutzertyp und ihre ID sowie andere Attribute ab.	Alle. Eine Anmeldung ist nicht erforderlich.
loginHSM und logoutHSM	An- und Abmeldung bei einem HSM.	Alle.
quit	Beendet cloudhsm_mgmt_util.	Alle. Eine Anmeldung ist nicht erforderlich.
server	Dient zum Aufrufen und Beenden des Servermodus auf einem HSM.	Alle.
registerQuorumPubKey (Schlüssel)	Ordnet einen HSM-Benutzer einem asymmetrischen RSA-2048-Schlüsselpaar zu.	CO
setAttribute	Ändert die Werte der Attribute zum Bezeichnen, Verschlüsseln, Entschlüsseln, Verpacken und Entpacken eines vorhandenen Schlüssels.	CU
shareKey	Gibt einen vorhandenen Schlüssel für andere Benutzer frei.	CU

Befehl	Description	Benutzertyp
syncKey	Synchronisiert einen Schlüssel zwischen geklonten Clustern AWS CloudHSM .	CU, CO
syncUser	Synchronisiert einen Benutzer über geklonte Cluster hinweg. AWS CloudHSM	CO

Ändern Sie das Passwort eines Benutzers mit CMU

Verwenden Sie den `changePswd` Befehl in der Datei AWS CloudHSM `cloudhsm_mgmt_util` (CMU), um das Passwort eines vorhandenen Benutzers auf den Hardware-Sicherheitsmodulen (HSM) im Cluster zu ändern. AWS CloudHSM

Jeder Benutzer kann das eigene Passwort ändern. Darüber hinaus können Crypto-Beauftragte (COs und PCOs) das Passwort eines anderen COs oder Crypto-Benutzers (CU) ändern. Sie müssen das aktuelle Passwort nicht eingeben, um die Änderung vorzunehmen.

Note

Sie können das Passwort eines Benutzers, der derzeit am AWS CloudHSM Client oder `key_mgmt_util` angemeldet ist, nicht ändern.

Um `changePswd` zu beheben

Vor dem Ausführen eines CMU-Befehls müssen Sie die CMU starten und sich beim HSM anmelden. Stellen Sie sicher, dass Sie sich mit einem Benutzertyp anmelden, der die Befehle ausführen kann, die Sie verwenden möchten.

Aktualisieren Sie die Konfigurationsdateien für CMU HSMs, wenn Sie sie hinzufügen oder löschen. Andernfalls sind die Änderungen, die Sie vornehmen, möglicherweise nicht für alle HSMs Mitglieder des Clusters wirksam.

Benutzertyp

Die folgenden Benutzer können diesen Befehl ausführen.

- Verschlüsselungsverantwortliche (Crypto Officers, CO)
- Crypto-Benutzer (Crypto User, CU)

Syntax

Geben Sie die Argumente in der Reihenfolge ein, die im Syntaxdiagramm angegeben ist. Verwenden Sie den `-hpswd`-Parameter, um Ihr Passwort zu maskieren. Um die Zwei-Faktor-Authentifizierung (2FA) für einen CO-Benutzer zu aktivieren, verwenden Sie den `-2fa`-Parameter und geben Sie einen Dateipfad an. Weitere Informationen finden Sie unter [the section called “Argumente”](#).

```
changePswd <user-type> <user-name> <password> [-hpswd] [-2fa </path/to/authdata>]
```

Beispiele

Die folgenden Beispiele zeigen, wie Sie `changePassword` das Passwort für den aktuellen Benutzer oder einen anderen Benutzer in Ihrem Konto zurücksetzen können HSMs.

Example: Ändert Ihr Passwort

Jeder Benutzer auf der HSMs kann `changePswd` damit sein eigenes Passwort ändern. Bevor Sie das Passwort ändern, verwenden Sie [info](#), um Informationen über jedes Mitglied des HSMs Clusters abzurufen, einschließlich des Benutzernamens und des Benutzertyps des angemeldeten Benutzers.

Die folgende Ausgabe zeigt, dass Bob derzeit als CU angemeldet ist.

```
aws-cloudhsm> info server 0
```

Id	Name	Hostname	Port	State	Partition
0	10.1.9.193	10.1.9.193	2225	Connected	hsm-jqici4covtv

```

LoginState
Logged in as 'bob(CU)'

aws-cloudhsm> info server 1
```

Id	Name	Hostname	Port	State	Partition
1	10.1.10.7	10.1.10.7	2225	Connected	hsm-ogi3sywxbqx

```

LoginState
Logged in as 'bob(CU)'
```

Um das Passwort zu ändern, führt Bob `changePswd` aus, gefolgt von dem Benutzertyp, dem Benutzernamen und einem neuen Passwort.

```
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: Ändert das Passwort eines anderen Benutzers

Sie müssen ein CO oder PCO sein, um das Passwort eines anderen CO oder CU auf dem HSMs ändern zu können. Bevor Sie das Passwort für einen anderen Benutzer ändern, prüfen Sie mit dem Befehl [info](#), dass Ihr Benutzertyp entweder CO oder PCO ist.

Die folgende Ausgabe bestätigt, dass Alice eine CO und derzeit angemeldet ist.

```
aws-cloudhsm>info server 0
```

Id	Name	Hostname	Port	State	Partition
0	10.1.9.193	10.1.9.193	2225	Connected	hsm-jqici4covtv

LoggedIn as 'alice(C0)'

```
aws-cloudhsm>info server 1
```

Id	Name	Hostname	Port	State	Partition
0	10.1.10.7	10.1.10.7	2225	Connected	hsm-ogi3sywxbqx

LoggedIn as 'alice(C0)'

Alice will das Passwort eines anderen Benutzers (John) zurücksetzen. Bevor sie das Passwort ändert, überprüft sie mit dem Befehl [listUsers](#) den Benutzertyp von John.

Die folgende Ausgabe listet John als CO-Benutzer auf.

```
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
NO			
2	AU	jane	NO
NO			
3	CU	bob	NO
NO			
4	CU	alice	NO
NO			
5	CO	john	NO
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
NO			
2	AU	jane	NO
NO			
3	CU	bob	NO
NO			
4	CO	alice	NO
NO			
5	CO	john	NO
NO			

Um das Passwort zu ändern, führt Alice `changePswd` aus, gefolgt von Johns Benutzertyp, Benutzernamen und einem neuen Passwort.

```
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(C0) on 2 nodes

Argumente

Geben Sie die Argumente in der Reihenfolge ein, die im Syntaxdiagramm angegeben ist. Verwenden Sie den `-hpswd`-Parameter, um Ihr Passwort zu maskieren. Um 2FA für einen CO-Benutzer zu aktivieren, verwenden Sie den `-2fa`-Parameter und geben Sie einen Dateipfad an. Weitere Informationen zur Arbeit mit 2FA finden Sie unter [2FA für Benutzer verwalten](#).

```
changePswd <user-type> <user-name> <password | -hpswd> [-2fa </path/to/authdata>]
```

<user-type>

Gibt den aktuellen Typ des Benutzers an, dessen Passwort Sie ändern. Sie können `changePswd` nicht verwenden, um den Benutzertyp zu ändern.

Gültige Werte sind C0, CU, PC0 und PREC0.

Zum Abrufen des Benutzertyps verwenden Sie den Befehl [listUsers](#). Ausführliche Informationen zu den Benutzertypen in einem HSM finden Sie unter [HSM-Benutzertypen für Management Utility AWS CloudHSM](#).

Erforderlich: Ja

<user-name>

Gibt den Anzeigenamen des Benutzers an. Bei diesem Parameter wird nicht zwischen Groß- und Kleinschreibung unterschieden. Sie können `changePswd` nicht verwenden, um den Benutzernamen zu ändern.

Erforderlich: Ja

<password | -hpswd >

Gibt ein neues Passwort für den Benutzer an. Geben Sie eine Zeichenfolge von 7 bis 32 Zeichen ein. Bei diesem Wert ist die Groß- und Kleinschreibung zu beachten. Das Passwort wird in

Klartext angezeigt, wenn Sie es eingeben. Um Ihr Passwort zu verbergen, verwenden Sie den `-hpswd`-Parameter anstelle des Passworts und folgen Sie den Aufforderungen.

Erforderlich: Ja

`[-2fa </ >path/to/authdata]`

Gibt an, dass 2FA für diesen CO-Benutzer aktiviert wird. Um die für die Einrichtung von 2FA erforderlichen Daten abzurufen, fügen Sie einen Pfad zu einem Speicherort im Dateisystem mit einem Dateinamen nach dem `-2fa`-Parameter ein. Weitere Informationen zur Arbeit mit 2FA finden Sie unter [2FA für Benutzer verwalten](#).

Erforderlich: Nein

Verwandte Themen

- [info](#)
- [listUsers](#)
- [createUser](#)
- [deleteUser](#)

Einen AWS CloudHSM Benutzer mit CMU erstellen

Verwenden Sie den `createUser` Befehl in `cloudhsm_mgmt_util` (CMU), um einen Benutzer für die Hardware-Sicherheitsmodule (HSM) im Cluster zu erstellen. AWS CloudHSM Nur Kryptobeauftragte (und) können diesen Befehl ausführen. COs PRECOs Wenn der Befehl erfolgreich ist, wird der gesamte Benutzer HSMs im Cluster erstellt.

Um Fehler bei `createUser` zu beheben

Wenn Ihre HSM-Konfiguration falsch ist, wurde der Benutzer möglicherweise nicht auf allen erstellt. HSMs Um den Benutzer zu einem Benutzer hinzuzufügen, HSMs in dem er fehlt, verwenden Sie den Befehl [SyncUser](#) oder [createUser](#) nur für diejenigen HSMs, denen dieser Benutzer fehlt. Um Konfigurationsfehler zu meiden, führen Sie das [configure](#)-Tool mit der `-m`-Option aus.

Vor dem Ausführen eines CMU-Befehls müssen Sie die CMU starten und sich beim HSM anmelden. Stellen Sie sicher, dass Sie sich mit einem Benutzertyp anmelden, der die Befehle ausführen kann, die Sie verwenden möchten.

Wenn Sie etwas hinzufügen oder löschen HSMs, aktualisieren Sie die Konfigurationsdateien für CMU. Andernfalls sind die Änderungen, die Sie vornehmen, möglicherweise nicht für alle HSMs Mitglieder des Clusters wirksam.

Benutzertyp

Die folgenden Benutzertypen können diesen Befehl ausführen.

- Crypto Officers (CO, PRECO)

Syntax

Geben Sie die Argumente in der Reihenfolge ein, die im Syntaxdiagramm angegeben ist. Verwenden Sie den `-hpswd`-Parameter, um Ihr Passwort zu maskieren. Um einen CO-Benutzer mit Zwei-Faktor-Authentifizierung (2FA) zu erstellen, verwenden Sie den `-2fa`-Parameter und geben einen Dateipfad an. Weitere Informationen finden Sie unter [the section called “Argumente”](#).

```
createUser <user-type> <user-name> <password> [-hpswd] [-2fa </path/to/authdata>]
```

Beispiele

Diese Beispiele zeigen, wie `createUser` Sie neue Benutzer in Ihrem erstellen können HSMs.

Example: Erstellen eines Crypto Officers

In diesem Beispiel wird ein Crypto Officer (CO) auf dem HSMs in einem Cluster erstellt. Der erste Befehl verwendet [loginHSM](#), um sich beim HSM als Verschlüsselungsverantwortlicher anzumelden.

```
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)
```

Der zweite Befehl verwendet den `createUser`-Befehl zum Erstellen von `alice`, einem neuen Verschlüsselungsverantwortlichen im HSM.

In der Warnmeldung wird erklärt, dass der HSMs Befehl Benutzer auf allen Clustern erstellt. Schlägt der Befehl jedoch auf einem der Server fehl HSMs, ist der Benutzer auf diesen nicht vorhanden HSMs. Geben Sie `y` ein, um fortzufahren.

Die Ausgabe zeigt, dass der neue Benutzer auf allen drei HSMs im Cluster erstellt wurde.

```
aws-cloudhsm> createUser C0 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(C0) on 3 nodes
```

Wenn der Befehl abgeschlossen ist, verfügt er auf dem HSM über dieselben Berechtigungen wie der admin CO-Benutzer, einschließlich der Änderung des Kennworts eines beliebigen Benutzers auf dem HSMs.

Der letzte Befehl verwendet den Befehl [ListUsers](#), um zu überprüfen, ob auf allen drei HSMs im Cluster alice vorhanden ist. Die Ausgabe zeigt auch, dass alice die Benutzer-ID 3 zugewiesen wird.. Sie verwenden die Benutzer-ID zur Identifizierung alice in anderen Befehlen, z. [findAllKeys](#)

```
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
1	PRECO	admin	YES
0	NO		
2	AU	app_user	NO
0	NO		
3	CO	alice	NO
0	NO		

Example: Erstellen eines Crypto-Benutzers

In diesem Beispiel wird ein Crypto-Benutzer (CU, Crypto User), bob, im HSM erstellt. Crypto-Benutzer können Schlüssel erstellen und verwalten, aber sie können keine Benutzer verwalten.

Nachdem Sie eingegeben haben, um auf die Warnmeldung zu antworten, zeigt die Ausgabe, dass die Datei auf allen drei HSMs Clustern erstellt wurde. Der neue CU kann sich bei dem HSM anmelden, um Schlüssel zu erstellen und zu verwalten.

Der Befehl verwendete den Passwortwert `defaultPassword`. Später kann bob oder ein beliebiger CO mithilfe des [changePswd](#)-Befehls sein Passwort ändern.

```
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

Argumente

Geben Sie die Argumente in der Reihenfolge ein, die im Syntaxdiagramm angegeben ist. Verwenden Sie den `-hpswd`-Parameter, um Ihr Passwort zu maskieren. Um einen CO-Benutzer mit aktivierter 2FA zu erstellen, verwenden Sie den `-2fa`-Parameter und geben Sie einen Dateipfad an. Weitere Informationen über 2FA finden Sie in [2FA für Benutzer verwalten](#).

```
createUser <user-type> <user-name> <password> | -hpswd> [-2fa </path/to/authdata>]
```

<user-type>

Gibt den Benutzertyp an. Dieser Parameter muss angegeben werden.

Ausführliche Informationen zu den Benutzertypen in einem HSM finden Sie unter [HSM-Benutzertypen für Management Utility AWS CloudHSM](#).

Zulässige Werte:

- CO: Verschlüsselungsverantwortliche können Benutzer, aber keine Schlüssel verwalten.
- CU: Crypto-Benutzer können Schlüssel erstellen und verwalten und Schlüssel in kryptografischen Vorgängen verwenden.

Der PRECO wird in einen CO konvertiert, wenn Sie während der [HSM-Aktivierung](#) ein Passwort zuweisen.

Erforderlich: Ja

<user-name>

Gibt einen Anzeigenamen für den Benutzer an. Die maximale Länge beträgt 31 Zeichen. Das einzige zulässige Sonderzeichen ist ein Unterstrich (`_`).

Sie können den Namen eines Benutzers nach der Erstellung nicht mehr ändern. In `cloudhsm_mgmt_util`-Befehlen muss bei Benutzertyp und Passwort die Groß- und Kleinschreibung beachtet werden, beim Benutzernamen nicht.

Erforderlich: Ja

<password | -hpswd >

Gibt ein Passwort für den Benutzer an. Geben Sie eine Zeichenfolge von 7 bis 32 Zeichen ein. Bei diesem Wert ist die Groß- und Kleinschreibung zu beachten. Das Passwort wird in Klartext

angezeigt, wenn Sie es eingeben. Um Ihr Passwort zu verbergen, verwenden Sie den `-hpswd`-Parameter anstelle des Passworts und folgen Sie den Aufforderungen.

Um ein Benutzerpasswort zu ändern, verwenden Sie [changePswd](#). Jeder HSM-Benutzer kann sein eigenes Passwort ändern, aber CO-Benutzer können das Passwort jedes Benutzers (beliebigen Typs) auf dem HSMs ändern.

Erforderlich: Ja

`[-2fa </ >] path/to/authdata`

Gibt die Erstellung eines CO-Benutzers mit aktivierter 2FA an. Um die für die Einrichtung der 2FA-Authentifizierung erforderlichen Daten abzurufen, fügen Sie einen Pfad zu einem Speicherort im Dateisystem mit einem Dateinamen nach dem `-2fa`-Parameter ein. Weitere Informationen zum Einrichten und Arbeiten mit 2FA finden Sie unter [2FA für Benutzer verwalten](#).

Erforderlich: Nein

Verwandte Themen

- [listUsers](#)
- [deleteUser](#)
- [syncUser](#)
- [changePswd](#)

Löschen Sie einen AWS CloudHSM Benutzer mit CMU

Verwenden Sie den `deleteUser` Befehl in der Datei AWS CloudHSM `cloudhsm_mgmt_util` (CMU), um einen Benutzer aus den Hardware-Sicherheitsmodulen (HSM) im Cluster zu löschen. AWS CloudHSM Nur Crypto Officers (CO) können diesen Befehl ausführen. Sie können keinen Benutzer löschen, der derzeit bei einem HSM angemeldet ist. Weitere Informationen zum Löschen von Benutzern finden Sie unter [Löschen von HSM-Benutzern](#).

Tip

Sie können keine Crypto-Benutzer (CU) löschen, die Schlüssel besitzen.

Benutzertyp

Die folgenden Benutzertypen können diesen Befehl ausführen.

- CO

Syntax

Da dieser Befehl keine benannten Parameter hat, müssen Sie die Argumente in der im Syntaxdiagramm angegebenen Reihenfolge eingeben.

```
deleteUser <user-type> <user-name>
```

Beispiel

In diesem Beispiel wird ein Crypto Officer (CO) aus dem HSMs in einem Cluster gelöscht. Der erste Befehl verwendet [ListUsers](#), um alle Benutzer auf dem HSMs aufzulisten.

Die Ausgabe zeigt, dass der Benutzer3,alice, ein CO auf dem HSMs ist.

```
aws-cloudhsm> listUsers
```

```
Users on server 0(10.0.0.1):
```

```
Number of users found:3
```

User Id LoginFailureCnt	User Type 2FA	User Name	MofnPubKey
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 LoginFailureCnt	User Type 2FA	User Name	MofnPubKey
1	PCO	admin	YES
0	NO		
2	AU	app_user	NO
0	NO		

3	C0	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	PC0	admin	YES
0	NO		
2	AU	app_user	NO
0	NO		
3	C0	alice	NO
0	NO		

Der zweite Befehl verwendet den `deleteUser` Befehl zum Löschen `alice` aus dem HSMs.

Die Ausgabe zeigt, dass der Befehl bei allen drei HSMs im Cluster erfolgreich war.

```
aws-cloudhsm> deleteUser C0 alice
Deleting user alice(C0) 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)
```

Der letzte Befehl verwendet den `listUsers` Befehl, um zu überprüfen, `alice` ob er von allen drei HSMs auf dem Cluster gelöscht wurde.

```
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	PC0	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	PC0	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	PC0	admin	YES
0	NO		
2	AU	app_user	NO
0	NO		

Argumente

Da dieser Befehl keine benannten Parameter hat, müssen Sie die Argumente in der im Syntaxdiagramm angegebenen Reihenfolge eingeben.

```
deleteUser <user-type> <user-name>
```

<user-type>

Gibt den Benutzertyp an. Dieser Parameter muss angegeben werden.

Tip

Sie können keine Crypto-Benutzer (CU) löschen, die Schlüssel besitzen.

Gültige Werte sind CO und CU.

Zum Abrufen des Benutzertyps verwenden Sie den Befehl [listUsers](#). Ausführliche Informationen zu den Benutzertypen in einem HSM finden Sie unter [HSM-Benutzertypen für Management Utility AWS CloudHSM](#).

Erforderlich: Ja

<user-name>

Gibt einen Anzeigenamen für den Benutzer an. Die maximale Länge beträgt 31 Zeichen. Das einzige zulässige Sonderzeichen ist ein Unterstrich (_).

Sie können den Namen eines Benutzers nach der Erstellung nicht mehr ändern. In `cloudhsm_mgmt_util`-Befehlen muss bei Benutzertyp und Passwort die Groß- und Kleinschreibung beachtet werden, beim Benutzernamen nicht.

Erforderlich: Ja

Verwandte Themen

- [listUsers](#)
- [createUser](#)
- [syncUser](#)
- [changePswd](#)

Listet die Schlüssel auf, die ein AWS CloudHSM Krypto-Benutzer besitzt, mithilfe der CMU

Verwenden Sie den `findAllKeys` Befehl in der Datei AWS CloudHSM `cloudhsm_mgmt_util` (CMU), um die Schlüssel abzurufen, die ein bestimmter Krypto-Benutzer (CU) besitzt oder gemeinsam nutzt. AWS CloudHSM Der Befehl gibt auch einen Hash der Benutzerdaten auf jedem der HSMs. Sie können den Hash verwenden, um auf einen Blick festzustellen, ob die Benutzer-, Schlüsseleigentums- und Schlüsselfreigabedaten für alle HSMs im Cluster identisch sind. In der Ausgabe werden die Schlüssel, die dem Benutzer gehören, von (o) mit Anmerkungen versehen, und gemeinsam genutzte Schlüssel werden von (s) kommentiert.

`findAllKeys` gibt öffentliche Schlüssel nur zurück, wenn die angegebene CU Eigentümer des Schlüssels ist, obwohl alle CUs Benutzer des HSM jeden öffentlichen Schlüssel verwenden können. Dieses Verhalten unterscheidet sich vom Befehl [findKey](#) in `key_mgmt_util`, der öffentliche Schlüssel für alle Crypto-Benutzer zurückgibt.

Nur Kryptobeauftragte (COs und PCOs) und Appliance-Benutzer (AUs) können diesen Befehl ausführen. Crypto-Benutzer (CUs) können die folgenden Befehle ausführen:

- [listUsers](#), um alle Benutzer zu suchen
- [findKey](#) in `key_mgmt_util`, um die Schlüssel, die sie verwenden können, zu suchen
- [getKeyInfo](#) in `key_mgmt_util`, um den Besitzer und die gemeinsamen Benutzer eines bestimmten Schlüssels zu finden, den sie besitzen oder teilen

Vor dem Ausführen eines CMU-Befehls müssen Sie die CMU starten und sich beim HSM anmelden. Stellen Sie sicher, dass Sie sich mit einem Benutzertyp anmelden, der die Befehle ausführen kann, die Sie verwenden möchten.

Wenn Sie etwas hinzufügen oder löschen HSMs, aktualisieren Sie die Konfigurationsdateien für CMU. Andernfalls sind die Änderungen, die Sie vornehmen, möglicherweise nicht für alle HSMs Mitglieder des Clusters wirksam.

Benutzertyp

Die folgenden Benutzer können diesen Befehl ausführen.

- Verschlüsselungsverantwortliche (CO, PCO)
- Appliance-Benutzer (Appliance User, AU)

Syntax

Da dieser Befehl keine benannten Parameter hat, müssen Sie die Argumente in der im Syntaxdiagramm angegebenen Reihenfolge eingeben.

```
findAllKeys <user id> <key hash (0/1)> [<output file>]
```

Beispiele

Diese Beispiele zeigen, wie Sie `findAllKeys` alle Schlüssel für einen Benutzer finden und einen Hash mit den wichtigsten Benutzerinformationen für jeden Benutzer abrufen können HSMs.

Example: Suchen der Schlüssel für einen CU

In diesem Beispiel werden `findAllKeys` die Schlüssel in dem Verzeichnis gesucht HSMs , das Benutzer 4 besitzt und gemeinsam nutzt. Der Befehl verwendet den Wert 0 für das zweite Argument, um den Hashwert zu unterdrücken. Da der optionale Dateiname weggelassen wird, führt der Befehl Schreibvorgänge in der Standardausgabe (stdout) aus.

Die Ausgabe zeigt, dass Benutzer 4 6 Schlüssel verwenden kann: 8, 9, 17, 262162, 19 und 31. Die Ausgabe verwendet ein (s), um Schlüssel anzugeben, die explizit vom Benutzer freigegeben werden. Die Schlüssel, die der Benutzer besitzt, werden durch ein (o) gekennzeichnet und umfassen symmetrische und private Schlüssel, die der Benutzer nicht freigibt, sowie öffentliche Schlüssel, die für alle Crypto-Benutzer verfügbar sind.

```
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: Sicherstellen, dass Benutzerdaten synchronisiert sind

In diesem Beispiel wird `findAllKeys` überprüft, ob alle HSMs im Cluster dieselben Werte für Benutzer, Schlüsselbesitz und gemeinsame Nutzung von Schlüsseln enthalten. Zu diesem Zweck wird ein Hashwert der Schlüsselbenutzerdaten in jedem HSM abgerufen und die Hashwerte werden verglichen.

Zum Abrufen des Schlüsselhashwerts verwendet der Befehl den Wert 1 für das zweite Argument. Der optionale Dateiname wird weggelassen, sodass der Befehl den Schlüsselhashwert in die Standardausgabe (stdout) schreibt.

In dem Beispiel wird Benutzer angegeben 6, aber der Hashwert ist derselbe für jeden Benutzer, der einen der Schlüssel auf dem besitzt oder teilt HSMs. Wenn der angegebene Benutzer keine Schlüssel besitzt oder gemeinsam verwendet, z. B. ein Verschlüsselungsverantwortlicher (CO), gibt der Befehl keinen Hashwert zurück.

Die Ausgabe zeigt, dass der Schlüssel-Hash mit beiden Hash HSMs im Cluster identisch ist. Wenn eines der HSMs verschiedene Benutzer, unterschiedliche Schlüsselbesitzer oder verschiedene gemeinsame Benutzer hätte, wären die Schlüsselhashwerte nicht identisch.

```

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)

```

Mit diesem Befehl wird gezeigt, dass der Hashwert die Benutzerdaten für alle Schlüssel im HSM darstellt. Der Befehl verwendet `findAllKeys` für Benutzer 3. Im Gegensatz zu Benutzer 6, der nur 3 Schlüssel besitzt oder gemeinsam verwendet, besitzt oder verwendet Benutzer 3 17 Schlüssel. Dennoch ist der Schlüsselhashwert identisch.

```

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)

```

Argumente

Da dieser Befehl keine benannten Parameter hat, müssen Sie die Argumente in der im Syntaxdiagramm angegebenen Reihenfolge eingeben.

```
findAllKeys <user id> <key hash (0/1)> [<output file>]
```

<user id>

Ruft alle Schlüssel ab, die der Benutzer besitzt oder die für ihn freigegeben sind. Geben Sie die Benutzer-ID eines Benutzers auf dem ein HSMs. Um den Benutzer IDs aller Benutzer zu finden, verwenden Sie [ListUsers](#).

Alle Benutzer IDs sind gültig, geben aber nur Schlüssel für Krypto-Benutzer findAllKeys zurück (CUs).

Erforderlich: Ja

<key hash>

Umfasst (1) oder schließt (0) einen Hashwert des Benutzereigentums oder der Freigabedaten für alle Schlüssel in jedem HSM aus.

Wenn das Argument `user id` einen Benutzer darstellt, der Schlüssel besitzt oder gemeinsam verwendet, ist der Schlüsselhashwert ausgefüllt. Der Schlüsselhashwert ist für alle Benutzer, die Schlüssel besitzen oder im HSM gemeinsam verwenden, identisch, auch wenn sie verschiedene Schlüssel besitzen und gemeinsam verwenden. Wenn `user id` jedoch einen Benutzer darstellt, der keine Schlüssel besitzt oder gemeinsam verwendet, z. B. einen Verschlüsselungsverantwortlichen (CO), wird der Hashwert nicht angegeben.

Erforderlich: Ja

<output file>

Schreibt die Ausgabe in die angegebene Datei.

Erforderlich: Nein

Standard: Stdout

Verwandte Themen

- [changePswd](#)
- [deleteUser](#)
- [listUsers](#)

- [syncUser](#)
- [findKey](#) in key_mgmt_util
- [getKeyInfo](#) in key_mgmt_util

Rufen Sie mithilfe von AWS CloudHSM CMU einen identifizierenden Attributwert ab

Verwenden Sie den `getAttribute` Befehl in der Datei AWS CloudHSM `cloudhsm_mgmt_util` (CMU), um einen Attributwert für einen Schlüssel von allen Hardware-Sicherheitsmodulen (HSM) im AWS CloudHSM Cluster abzurufen und ihn in die Standardausgabe (Standardausgabe) oder in eine Datei zu schreiben. CUs Nur Krypto-Benutzer () können diesen Befehl ausführen.

Schlüsselattribute sind Eigenschaften eines Schlüssels. Dazu zählen Merkmale wie der Schlüsseltyp, Klasse, Beschriftung und ID sowie Werte, die Aktionen darstellen, die Sie für den Schlüssel ausführen können, wie verschlüsseln, entschlüsseln, packen, signieren und überprüfen.

Sie können `getAttribute` nur für Schlüssel ausführen, die Ihnen gehören oder die für Sie freigegeben wurden. Sie können diesen Befehl oder den Befehl [getAttribute](#) in `key_mgmt_util` ausführen, der einen oder alle Attributwerte eines Schlüssels in eine Datei schreibt.

Zum Abrufen einer Liste der Attribute und Konstanten, die sie darstellen, verwenden Sie den Befehl [listAttributes](#). Um die Attributwerte von vorhandenen Schlüsseln zu ändern, verwenden Sie [setAttribute](#) in `key_mgmt_util` und [setAttribute](#) in `cloudhsm_mgmt_util`. Hilfe zur Interpretation der Schlüsselattribute finden Sie unter [AWS CloudHSM Referenz zu Schlüsselattributen für KMU](#).

Vor dem Ausführen eines CMU-Befehls müssen Sie die CMU starten und sich beim HSM anmelden. Stellen Sie sicher, dass Sie sich mit einem Benutzertyp anmelden, der die Befehle ausführen kann, die Sie verwenden möchten.

Aktualisieren Sie die Konfigurationsdateien für CMU HSMs, wenn Sie sie hinzufügen oder löschen. Andernfalls sind die Änderungen, die Sie vornehmen, möglicherweise nicht für alle HSMs Mitglieder des Clusters wirksam.

Benutzertyp

Die folgenden Benutzer können diesen Befehl ausführen.

- Crypto-Benutzer (Crypto User, CU)

Syntax

Da dieser Befehl keine benannten Parameter hat, müssen Sie die Argumente in der im Syntaxdiagramm angegebenen Reihenfolge eingeben.

```
getAttribute <key handle> <attribute id> [<filename>]
```

Beispiel

In diesem Beispiel wird der Wert des extrahierbaren Attributs für einen Schlüssel in der HSMs abgerufen. Sie können einen Befehl wie diesen verwenden, um festzustellen, ob Sie einen Schlüssel aus dem HSMs exportieren können.

Der erste Befehl verwendet [listAttributes](#), um die Konstante zu suchen, die das extrahierbare Attribut darstellt. Die Ausgabe zeigt, dass die Konstante für OBJ_ATTR_EXTRACTABLE 354 ist. Sie können diese Informationen mit Beschreibungen der Attribute und ihrer Werte auch unter [AWS CloudHSM Referenz zu Schlüsselattributen für KMU](#) finden.

```
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

Der zweite Befehl verwendet `getAttribute`, um den Wert des extrahierbaren Attributs für den Schlüssel mit dem Schlüsselnamen `262170` im HSMs abzurufen. Zum Angeben des extrahierbaren Attributs verwendet der Befehl `354`, die Konstante, die das extrahierbare Attribut darstellt. Da der Befehl keinen Dateinamen angibt, schreibt `getAttribute` die Ausgabe in die Standardausgabe (stdout).

Die Ausgabe zeigt, dass der Wert des extrahierbaren Attributs im gesamten HSM 1 ist. Dieser Wert gibt an, dass der Eigentümer des Schlüssels ihn exportieren kann. Wenn der Wert 0 (0x0) ist, kann er nicht exportiert werden. HSMs Sie legen den Wert des extrahierbaren Attributs beim Erstellen eines Schlüssels fest und können ihn nicht mehr ändern.

```
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
```

Argumente

Da dieser Befehl keine benannten Parameter hat, müssen Sie die Argumente in der im Syntaxdiagramm angegebenen Reihenfolge eingeben.

```
getAttribute <key handle> <attribute id> [<filename>]
```


<key-handle>

Gibt das Schlüssel-Handle des Zielschlüssels an. Sie können nur jeweils einen Schlüssel in den einzelnen Befehlen angeben. Um das Schlüsselhandle eines Schlüssels zu erhalten, verwenden Sie [findKey](#) in key_mgmt_util.

Sie müssen Eigentümer des angegebenen Schlüssels sein oder er muss für Sie freigegeben sein. Um die Benutzer eines Schlüssels zu finden, verwenden Sie [getKeyInfo](#) in key_mgmt_util.

Erforderlich: Ja

<attribute id>

Bezeichnet das Attribut. Geben Sie eine Konstante, die ein Attribut darstellt, oder den Wert 512 ein, der alle Attribute repräsentiert. Zum Abrufen des Schlüsseltyps geben Sie z. B. 256 ein. Dies ist die Konstante für das Attribut OBJ_ATTR_KEY_TYPE.

Verwenden Sie [listAttributes](#), um die Attribute und deren Konstanten aufzulisten. Hilfe zur Interpretation der Schlüsselattribute finden Sie unter [AWS CloudHSM Referenz zu Schlüsselattributen für KMU](#).

Erforderlich: Ja

<filename>

Schreibt die Ausgabe in die angegebene Datei. Geben Sie einen Dateipfad ein.

Wenn die angegebene Datei vorhanden ist, überschreibt getAttribute die Datei ohne Warnung.

Erforderlich: Nein

Standard: Stdout

Verwandte Themen

- [getAttribute](#) in key_mgmt_util
- [listAttributes](#)
- [setAttribute](#) in cloudhsm_mgmt_util
- [setAttribute](#) in key_mgmt_util
- [Schlüsselattributreferenz](#)

Rufen Sie Hardwareinformationen für jedes HSM in einem AWS CloudHSM Cluster mit CMU ab

Verwenden Sie den `getHSMInfo` Befehl in der Datei AWS CloudHSM `cloudhsm_mgmt_util` (CMU), um Informationen über die Hardware abzurufen, auf der jedes Hardware-Sicherheitsmodul (HSM) ausgeführt wird, einschließlich Modell, Seriennummer, FIPS-Status, Speicher, Temperatur und Versionsnummern der Hardware und Firmware. Die Informationen umfassen auch die Server-ID, die `cloudhsm_mgmt_util` verwendet, um auf das HSM zu verweisen.

Vor dem Ausführen eines CMU-Befehls müssen Sie die CMU starten und sich beim HSM anmelden. Stellen Sie sicher, dass Sie sich mit einem Benutzertyp anmelden, der die Befehle ausführen kann, die Sie verwenden möchten.

Aktualisieren Sie die Konfigurationsdateien für CMU, wenn Sie sie hinzufügen oder löschen. HSMs Andernfalls sind die Änderungen, die Sie vornehmen, möglicherweise nicht für alle HSMs Mitglieder des Clusters wirksam.

Benutzertyp

Die folgenden Benutzertypen können diesen Befehl ausführen.

- Alle Benutzer. Sie müssen nicht angemeldet sein, um diesen Befehl auszuführen.

Syntax

Dieser Befehl hat keine Parameter.

```
getHSMInfo
```

Beispiel

In diesem Beispiel `getHSMInfo` werden Informationen über die HSMs im Cluster abgerufen.

```
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         :xxxxxxxxxxxxxxxx

```

...

Verwandte Themen

- [info](#)

AWS CloudHSM Benutzerinformationen zu einem Schlüssel mithilfe von CMU abrufen

Verwenden Sie den `getKeyInfo` Befehl in AWS CloudHSM `key_mgmt_util` (KMU), um den HSM-Benutzer (Hardware Security Module) der Benutzer zurückzugeben, die den Schlüssel verwenden können, einschließlich IDs des Besitzers und der Crypto-Benutzer (CU), mit denen der Schlüssel gemeinsam genutzt wird. Wenn die Quorum-Authentifizierung für einen Schlüssel aktiviert ist, gibt `getKeyInfo` auch die Anzahl der Benutzer zurück, die kryptografische Operationen genehmigen

müssen, von denen der Schlüssel verwendet wird. Sie können `getKeyInfo` nur auf Schlüsseln ausführen, die Ihnen gehören oder die für Sie freigegeben wurden.

Wenn Sie `getKeyInfo` auf öffentliche Schlüsseln ausführen, gibt `getKeyInfo` nur den Schlüsseleigentümer zurück, auch wenn alle HSM-Benutzer den öffentlichen Schlüssel verwenden können. Um den HSM-Benutzer IDs von Benutzern in Ihrem zu finden HSMs, verwenden Sie [ListUsers](#). Um nach den Schlüsseln für einen bestimmten Benutzer zu suchen, verwenden Sie [findKey](#) -u in `key_mgmt_util`. Crypto Officers können [findAllKeys](#) in `cloudhsm_mgmt_util` verwenden.

Ihren gehören die Schlüssel, die Sie erstellen. Sie können einen Schlüssel für andere Benutzer freigeben, wenn Sie ihn erstellen. Nutzen Sie dann zum Freigeben oder zum Aufheben der Freigabe [shareKey](#) in `cloudhsm_mgmt_util`.

Vor dem Ausführen eines CMU-Befehls müssen Sie die CMU starten und sich beim HSM anmelden. Stellen Sie sicher, dass Sie sich mit einem Benutzertyp anmelden, der die Befehle ausführen kann, die Sie verwenden möchten.

Aktualisieren Sie die Konfigurationsdateien für CMU, wenn Sie sie hinzufügen oder löschen HSMs. Andernfalls sind die Änderungen, die Sie vornehmen, möglicherweise nicht für alle HSMs Mitglieder des Clusters wirksam.

Benutzertyp

Die folgenden Benutzertypen können diesen Befehl ausführen.

- Crypto-Benutzer (Crypto User, CU)

Syntax

```
getKeyInfo -k <key-handle> [<output file>]
```

Beispiele

Diese Beispiele veranschaulichen die Verwendung von `getKeyInfo` zum Abrufen von Informationen über die Benutzer eines Schlüssels.

Example: Abfragen der Benutzer eines asymmetrischen Schlüssels

Über diesen Befehl erhalten Sie die Benutzer, die den (asymmetrischen) AES-Schlüssel mit dem Schlüssel-Handle 262162 verwenden können. Die Ausgabe zeigt, dass Benutzer 3 den Schlüssel besitzt und mit Benutzer 4 und 6 teilt.

Nur die Benutzer 3, 4 und 6 können getKeyInfo auf dem Schlüssel 262162 ausführen.

```
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: Abfragen der Benutzer eines symmetrischen Schlüsselpaares

Diese Befehle verwenden getKeyInfo, um die Benutzer abzurufen, die die Schlüssel in einem [\(symmetrischen\) ECC-Schlüsselpaar](#) verwenden können. Das Schlüssel-Handle des öffentlichen Schlüssels ist 262179. Das Schlüssel-Handle des privaten Schlüssels ist 262177.

Wenn Sie getKeyInfo den privaten Schlüssel (262177) verwenden, werden der Schlüsselbesitzer (3) und die Krypto-Benutzer (4CUs) zurückgegeben, mit denen der Schlüssel geteilt wird.

```
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
```

Wenn Sie `getKeyInfo` auf dem öffentlichen Schlüssel (262179) ausführen, wird nur der Schlüsseleigentümer, Benutzer 3, zurückgegeben.

```
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
```

Um zu bestätigen, dass Benutzer 4 den öffentlichen Schlüssel (und alle öffentlichen Schlüssel im HSM) verwenden kann, verwenden Sie den `-u`-Parameter von [findKey](#) in `key_mgmt_util`.

Die Ausgabe zeigt, dass Benutzer 4 sowohl den öffentlichen (262179) als auch den privaten Schlüssel (262177) im Schlüsselpaar verwenden kann. Benutzer 4 kann zudem alle anderen öffentlichen und privaten Schlüssel nutzen, die sie erstellt haben oder die für sie freigegeben wurden.

```
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: Abrufen des Quorum-Authentifizierungswerts (m_value) für einen Schlüssel

Dieses Beispiel zeigt, wie der m_value für einen Schlüssel abgerufen wird. m_value gibt die Anzahl der Benutzer im Quorum an, die alle kryptografischen Operationen genehmigen müssen, bei denen der Schlüssel verwendet wird, sowie alle Operationen zum Freigeben oder zum Aufheben der Freigabe eines Schlüssels.

Wenn die Quorum-Authentifizierung für einen Schlüssel aktiviert ist, muss das Quorum der Benutzer alle kryptografischen Operationen genehmigen, bei denen der Schlüssel verwendet wird. Zum Aktivieren der Quorum-Authentifizierung und Festlegen der Quorum-Größe nutzen Sie beim Erstellen des Schlüssels den Parameter -m_value.

Dieser Befehl wird verwendet [genSymKey](#), um einen 256-Bit-AES-Schlüssel zu erstellen, der mit Benutzer 4 geteilt wird. Er verwendet den Parameter m_value zur Aktivierung der Quorum-Authentifizierung und zur Festlegung der Quorum-Größe auf zwei Benutzer. Die Anzahl der Benutzer muss groß genug sein, um die erforderlichen Genehmigungen bereitstellen zu können.

Die Ausgabe zeigt, dass die Befehl Schlüssel 10 erstellt hat.

```
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
```

Dieser Befehl nutzt getKeyInfo in cloudhsm_mgmt_util zum Abrufen von Informationen über die Benutzer des Schlüssels 10. Die Ausgabe zeigt, dass der Schlüssel im Besitz des Benutzers 3 ist und gemeinsam mit Benutzer 4 verwendet wird. Sie zeigt auch, dass ein Quorum von zwei Benutzern jede kryptografische Operation genehmigen muss, bei der der Schlüssel verwendet wird.

```
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
```

Argumente

Da dieser Befehl keine benannten Parameter hat, müssen Sie die Argumente in der im Syntaxdiagramm angegebenen Reihenfolge eingeben.

```
getKeyInfo -k <key-handle> <output file>
```

<key-handle>

Gibt das Schlüssel-Handle eines Schlüssels im HSM an. Geben Sie das Schlüssel-Handle eines Schlüssels ein, den Sie besitzen oder teilen. Dieser Parameter muss angegeben werden.

Erforderlich: Ja

<output file>

Schreibt die Ausgabe nicht in stdout, sondern in die angegebene Datei. Wenn die Datei vorhanden ist, wird diese ohne Warnung überschrieben.

Erforderlich: Nein

Standard: stdout

Verwandte Themen

- [getKeyInfo](#) in `key_mgmt_util`
- [findKey](#) in `key_mgmt_util`
- [findAllKeys](#) in `cloudhsm_mgmt_util`
- [listUsers](#)
- [shareKey](#)

Rufen Sie mithilfe von CMU Informationen für jedes HSM in einem AWS CloudHSM Cluster ab

Verwenden Sie den `info` Befehl im AWS CloudHSM `cloudhsm_mgmt_util` (CMU), um Informationen über jedes Hardware-Sicherheitsmodul (HSM) im AWS CloudHSM Cluster abzurufen, einschließlich Hostname, Port, IP-Adresse sowie Name und Typ des Benutzers, der auf dem HSM bei `cloudhsm_mgmt_util` angemeldet ist.

Vor dem Ausführen eines CMU-Befehls müssen Sie die CMU starten und sich beim HSM anmelden. Stellen Sie sicher, dass Sie sich mit einem Benutzertyp anmelden, der die Befehle ausführen kann, die Sie verwenden möchten.

Wenn Sie die Konfigurationsdateien für CMU hinzufügen oder löschen, aktualisieren Sie sie. HSMs Andernfalls sind die Änderungen, die Sie vornehmen, möglicherweise nicht für alle HSMs Mitglieder des Clusters wirksam.

Benutzertyp

Die folgenden Benutzertypen können diesen Befehl ausführen.

- Alle Benutzer. Sie müssen nicht angemeldet sein, um diesen Befehl auszuführen.

Syntax

Da dieser Befehl keine benannten Parameter hat, müssen Sie die Argumente in der im Syntaxdiagramm angegebenen Reihenfolge eingeben.

```
info server <server ID>
```

Beispiel

In diesem Beispiel wird `info` verwendet, um Informationen über ein HSM im Cluster abzurufen. Der Befehl verwendet 0, um auf das erste HSM im Cluster zu verweisen. Die Ausgabe zeigt die IP-Adresse, den Port, den Typ und den Namen des aktuellen Benutzers an.

```
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)'				

Argumente

Da dieser Befehl keine benannten Parameter hat, müssen Sie die Argumente in der im Syntaxdiagramm angegebenen Reihenfolge eingeben.

```
info server <server ID>
```

<server id>

Gibt die Server-ID des HSMs an. Ihnen HSMs werden Ordnungszahlen zugewiesen, die die Reihenfolge angeben, in der sie dem Cluster hinzugefügt werden, beginnend mit 0. Verwenden Sie `get`, um die Server-ID eines HSM zu ermitteln. `HSMInfo`

Erforderlich: Ja

Verwandte Themen

- [getHSMInfo](#)
- [loginHSM und logoutHSM](#)

Die Attribute eines AWS CloudHSM Schlüssels mithilfe von CMU auflisten

Verwenden Sie den `listAttributes` Befehl in der Datei AWS CloudHSM `cloudhsm_mgmt_util` (CMU), um die Attribute eines Schlüssels und die Konstanten, die sie repräsentieren, aufzulisten. AWS CloudHSM Sie nutzen die Konstanten zum Identifizieren der Attribute in den Befehlen [getAttribute](#) und [setAttribute](#).

Hilfe zur Interpretation der Schlüsselattribute finden Sie unter [AWS CloudHSM Referenz zu Schlüsselattributen für KMU](#).

Bevor Sie einen key_mgmt_util-Befehl ausführen, müssen Sie [key_mgmt_util starten](#) und sich am HSM als Crypto-Benutzer (CU) [anmelden](#).

Benutzertyp

Die folgenden Benutzer können diesen Befehl ausführen.

- Alle Benutzer. Sie müssen nicht angemeldet sein, um diesen Befehl auszuführen.

Syntax

```
listAttributes [-h]
```

Beispiel

Mit diesem Befehl werden die Schlüsselattribute aufgelistet, die Sie mit key_mgmt_util abrufen und ändern können, sowie die Konstanten, die sie darstellen. Hilfe zur Interpretation der Schlüsselattribute finden Sie unter [AWS CloudHSM Referenz zu Schlüsselattributen für KMU](#). Um alle Attribute darzustellen, verwenden Sie 512.

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

Parameters

-h

Zeigt Hilfe für den Befehl an.

Erforderlich: Ja

Verwandte Themen

- [getAttribute](#)
- [setAttribute](#)
- [Schlüsselattributreferenz](#)

Alle AWS CloudHSM Benutzer auflisten, die CMU verwenden

Verwenden Sie den listUsers Befehl in der Datei AWS CloudHSM cloudhsm_mgmt_util, um die Benutzer in den einzelnen Hardware-Sicherheitsmodulen (HSM) zusammen mit ihrem Benutzertyp und anderen Attributen abzurufen. Alle Arten von Benutzern können diesen Befehl ausführen. Sie müssen nicht bei cloudhsm_mgmt_util angemeldet sein, um diesen Befehl auszuführen.

Vor dem Ausführen eines CMU-Befehls müssen Sie die CMU starten und sich beim HSM anmelden. Stellen Sie sicher, dass Sie sich mit einem Benutzertyp anmelden, der die Befehle ausführen kann, die Sie verwenden möchten.

Wenn Sie sie hinzufügen oder löschen, aktualisieren Sie die Konfigurationsdateien für die CMU. HSMs Andernfalls sind die Änderungen, die Sie vornehmen, möglicherweise nicht für alle HSMs Mitglieder des Clusters wirksam.

Benutzertyp

Die folgenden Benutzertypen können diesen Befehl ausführen.

- Alle Benutzer. Sie müssen nicht angemeldet sein, um diesen Befehl auszuführen.

Syntax

Dieser Befehl hat keine Parameter.

```
listUsers
```

Beispiel

Dieser Befehl listet die Benutzer der einzelnen Benutzer HSMs im Cluster auf und zeigt ihre Attribute an. Sie können das Attribut `User ID` zum Identifizieren von Benutzern in anderen Befehlen, z. B. `deleteUser`, `changePswd` und `findAllKeys`, verwenden.

```
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	PC0	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          C0          officer1          YES          0
      NO
      6          C0          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          PC0          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          C0          officer1        YES          0
      NO

```

Die Ausgabe umfasst die folgenden Benutzerattribute:

- Benutzer-ID: Identifiziert den Benutzer in den Befehlen `key_mgmt_util` und [cloudhsm_mgmt_util](#).
- [User type](#): Bestimmt die Operationen, die der Benutzer im HSM ausführen kann.
- User Name: Zeigt den benutzerdefinierten Anzeigenamen für den Benutzer an.
- MofnPubKey: Gibt an, ob der Benutzer ein key pair zum Signieren von [Quorum-Authentifizierungstoken](#) registriert hat.
- LoginFailureCnt: Gibt an, wie oft sich der Benutzer erfolglos angemeldet hat.
- 2FA: Gibt an, dass der Benutzer Multifaktor-Authentifizierung aktiviert hat.

Verwandte Themen

- [listUsers](#) in `key_mgmt_util`
- [createUser](#)
- [deleteUser](#)
- [changePswd](#)

Mit dem AWS CloudHSM Management Utility bei einem HSM an- und abmelden

Verwenden Sie die `logoutHSM` Befehle `loginHSM` und in der Datei `AWS CloudHSM cloudhsm_mgmt_util`, um sich bei jedem HSM in einem Cluster an- und abzumelden. Jeder beliebige Benutzer kann unabhängig vom Typ diese Befehle verwenden.

Note

Wenn Sie mehr als fünf falsche Anmeldeversuche tätigen, wird Ihr Konto gesperrt. Ein Crypto Officer (CO) muss Ihr Passwort mit dem Befehl [changePswd](#) in `cloudhsm_mgmt_util` zurücksetzen, um das Konto zu entsperren.

So beheben Sie Probleme mit `loginHSM` und `logoutHSM`

Bevor Sie diese `cloudhsm_mgmt_util`-Befehle ausführen, müssen Sie `cloudhsm_mgmt_util` starten.

Wenn Sie etwas hinzufügen oder löschen HSMs, aktualisieren Sie die Konfigurationsdateien, die der Client und die Befehlszeilentools verwenden. AWS CloudHSM Andernfalls sind die Änderungen, die Sie vornehmen, möglicherweise nicht für alle Mitglieder HSMs des Clusters wirksam.

Wenn Sie mehr als ein HSM in Ihrem Cluster haben, sind Ihnen möglicherweise weitere falsche Anmeldeversuche gestattet, bevor Ihr Konto gesperrt wird. Dies liegt daran, dass der CloudHSM-Client die Last auf verschiedene verteilt. HSMs Aus diesem Grund beginnt der Anmeldeversuch möglicherweise nicht jedes Mal auf demselben HSM. Wenn Sie diese Funktion testen, empfehlen wir Ihnen, dies auf einem Cluster mit nur einem aktiven HSM zu tun.

Wenn Sie Ihren Cluster vor Februar 2018 erstellt haben, wird Ihr Konto nach 20 falschen Anmeldeversuchen gesperrt.

Benutzertyp

Die folgenden Benutzer können diese Befehle ausführen.

- Beauftragter für die Zeit vor der Kryptographie (PRECO)
- Crypto Officer (CO)
- Crypto-Benutzer (Crypto User, CU)

Syntax

Geben Sie die Argumente in der Reihenfolge ein, die im Syntaxdiagramm angegeben ist. Verwenden Sie den `-hpswd`-Parameter, um Ihr Passwort zu maskieren. Um sich mit der Zwei-Faktor-Authentifizierung (2FA) anzumelden, verwenden Sie den `-2fa`-Parameter und geben Sie einen Dateipfad an. Weitere Informationen finden Sie unter [the section called “Argumente”](#).

```
loginHSM <user-type> <user-name> <password> [-hpswd] [-2fa </path/to/authdata>]
```

```
logoutHSM
```

Beispiele

Diese Beispiele zeigen, wie Sie alle HSMs in einem Cluster verwenden `loginHSM` und sich `logoutHSM` an- und abmelden können.

Example: Melden Sie sich beim HSMs In a Cluster an

Mit diesem Befehl melden Sie sich bei all HSMs in einem Cluster mit den Anmeldeinformationen eines CO-Benutzers mit dem Namen `admin` und dem Passwort von `anco12345`. Die Ausgabe zeigt, dass der Befehl erfolgreich war und dass Sie eine Verbindung mit dem hergestellt haben HSMs (was in diesem Fall `server 0` und `istserver 1`).

```
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: Melden Sie sich mit einem versteckten Passwort an

Dieser Befehl entspricht dem obigen Beispiel, außer dass Sie dieses Mal angeben, dass das System das Passwort verbergen soll.

```
aws-cloudhsm>loginHSM CO admin -hpswd
```

Sie werden vom System aufgefordert, Ihr Passwort einzugeben. Sie geben das Passwort ein, das System versteckt das Passwort und die Ausgabe zeigt, dass der Befehl erfolgreich war und dass Sie sich mit dem HSMs verbunden haben.


```
Enter password:
```

```
loginHSM success on server 0(10.0.2.9)
loginHSM success on server 1(10.0.3.11)

aws-cloudhsm>
```

Example: Abmelden von einem HSM

Mit diesem Befehl melden Sie sich von dem ab HSMs , bei dem Sie gerade angemeldet sind (was in diesem Fall `server 0` und `istserver 1`). Die Ausgabe zeigt, dass der Befehl erfolgreich war und dass Sie die Verbindung zum HSMs getrennt haben.

```
aws-cloudhsm>logoutHSM

logoutHSM success on server 0(10.0.2.9)
logoutHSM success on server 1(10.0.3.11)
```

Argumente

Geben Sie die Argumente in der Reihenfolge ein, die im Syntaxdiagramm angegeben ist. Verwenden Sie den `-hpswd`-Parameter, um Ihr Passwort zu maskieren. Um sich mit der Zwei-Faktor-Authentifizierung (2FA) anzumelden, verwenden Sie den `-2fa`-Parameter und geben Sie einen Dateipfad an. Weitere Informationen zur Arbeit mit 2FA finden Sie unter [2FA für Benutzer verwalten](#).

```
loginHSM <user-type> <user-name> <password> [-hpswd] [-2fa </path/to/authdata>]
```

<user type>

Gibt den Benutzertyp an, der sich bei der HSMs anmeldet. Weitere Informationen finden Sie unter [Benutzertyp](#) oben.

Erforderlich: Ja

<user name>

Gibt den Benutzernamen des Benutzers an, der sich bei der anmeldet HSMs.

Erforderlich: Ja

<password | -hpswd >

Gibt das Passwort des Benutzers an, der sich bei der anmeldet HSMs. Um Ihr Passwort zu verbergen, verwenden Sie den -hpswd-Parameter anstelle des Passworts und folgen Sie der Aufforderung.

Erforderlich: Ja

[-2fa </ >path/to/authdata]

Gibt an, dass das System einen zweiten Faktor verwenden soll, um diesen 2FA-fähigen CO-Benutzer zu authentifizieren. Um die erforderlichen Daten für die Anmeldung mit 2FA zu erhalten, fügen Sie einen Pfad zu einem Speicherort im Dateisystem mit einem Dateinamen nach dem -2fa-Parameter ein. Weitere Informationen zur Arbeit mit 2FA finden Sie unter [2FA für Benutzer verwalten](#).

Erforderlich: Nein

Verwandte Themen

- [Erste Schritte mit cloudhsm_mgmt_util](#)
- [Aktivieren des Clusters](#)

AWS CloudHSM Benutzer mithilfe von CMU mit Schlüsseln verknüpfen

Verwenden Sie den registerQuorumPubKey Befehl in der Datei AWS CloudHSM cloudhsm_mgmt_util, um Benutzern des Hardware Security Module (HSM) asymmetrische RSA-2048-Schlüsselpaare zuzuordnen. Sobald Sie HSM-Benutzer mit Schlüsseln verknüpft haben, können diese Benutzer den privaten Schlüssel verwenden, um Quorumanfragen zu genehmigen, und der Cluster kann den registrierten öffentlichen Schlüssel verwenden, um zu überprüfen, ob die Signatur vom Benutzer stammt. Weitere Informationen zur Quorum-Authentifizierung finden Sie unter [Quorum-Authentifizierung verwalten \(M-von-N-Zugriffskontrolle\)](#).

Tip

In der AWS CloudHSM Dokumentation wird die Quorumauthentifizierung manchmal als M of N (MoFN) bezeichnet, was bedeutet, dass mindestens M Genehmiger von insgesamt N Genehmigern vorhanden sind.

Benutzertyp

Die folgenden Benutzertypen können diesen Befehl ausführen.

- Verschlüsselungsverantwortliche (Crypto Officers, CO)

Syntax

Da dieser Befehl keine benannten Parameter hat, müssen Sie die Argumente in der im Syntaxdiagramm angegebenen Reihenfolge eingeben.

```
registerQuorumPubKey <user-type> <user-name> <registration-token> <signed-registration-token> <public-key>
```

Beispiele

Dieses Beispiel zeigt, wie Crypto Officers (CO) als Genehmiger für Quorum-Authentifizierungsanfragen mit registerQuorumPubKey registriert werden können. Um diesen Befehl auszuführen, benötigen Sie ein asymmetrisches RSA-2048-Schlüsselpaar, ein signiertes Token und ein unsigniertes Token. Weitere Informationen zu diesen Anforderungen finden Sie unter [the section called “Argumente”](#).

Example: Registrieren Sie einen HSM-Benutzer für die Quorum-Authentifizierung

In diesem Beispiel wird ein CO namens quorum_officer als Genehmiger für die Quorumauthentifizierung registriert.

```
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)
```

Der letzte Befehl verwendet den Befehl [ListUsers](#), um zu überprüfen, ob ein MoFN-Benutzer registriert `quorum_officer` ist.

```
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		

Argumente

Da dieser Befehl keine benannten Parameter hat, müssen Sie die Argumente in der im Syntaxdiagramm angegebenen Reihenfolge eingeben.

```
registerQuorumPubKey <user-type> <user-name> <registration-token> <signed-registration-token> <public-key>
```

<user-type>

Gibt den Benutzertyp an. Dieser Parameter muss angegeben werden.

Ausführliche Informationen zu den Benutzertypen in einem HSM finden Sie unter [HSM-Benutzertypen für Management Utility AWS CloudHSM](#).

Zulässige Werte:

- CO: Verschlüsselungsverantwortliche können Benutzer, aber keine Schlüssel verwalten.

Erforderlich: Ja

<user-name>

Gibt einen Anzeigenamen für den Benutzer an. Die maximale Länge beträgt 31 Zeichen. Das einzige zulässige Sonderzeichen ist ein Unterstrich (_).

Sie können den Namen eines Benutzers nach der Erstellung nicht mehr ändern. In `cloudhsm_mgmt_util`-Befehlen muss bei Benutzertyp und Passwort die Groß- und Kleinschreibung beachtet werden, beim Benutzernamen nicht.

Erforderlich: Ja

<registration-token>

Gibt den Pfad zu einer Datei an, die ein unsigniertes Registrierungstoken enthält. Kann beliebige Daten mit einer maximalen Dateigröße von 245 Byte enthalten. Weitere Informationen zum Erstellen eines unsignierten Registrierungstokens finden Sie unter [Registrierungstoken erstellen und signieren](#).

Erforderlich: Ja

<signed-registration-token>

Gibt den Pfad zu einer Datei an, die den vom SHA256 _PKCS-Mechanismus signierten Hash des Registrierungstokens enthält. Weitere Informationen finden Sie unter [Registrierungstoken erstellen und signieren](#).

Erforderlich: Ja

<public-key>

Gibt den Pfad zu einer Datei an, die den öffentlichen Schlüssel eines asymmetrischen RSA-2048-Schlüsselpaars enthält. Verwenden Sie den privaten Schlüssel, um das Registrierungstoken zu signieren. Weitere Informationen finden Sie unter [Erstellen eines RSA-Schlüsselpaars](#).

Erforderlich: Ja

 Note

Der Cluster verwendet denselben Schlüssel für die Quorum-Authentifizierung und für die Zwei-Faktor-Authentifizierung (2FA). Das bedeutet, dass Sie einen Quorumschlüssel für einen Benutzer, für den 2FA mit `registerQuorumPubKey` aktiviert ist, nicht rotieren können. Um den Schlüssel zu rotieren, müssen Sie `changePswd` verwenden. [Weitere Informationen zur Verwendung von Quorum-Authentifizierung und 2FA finden Sie unter Quorum-Authentifizierung und 2FA.](#)

Verwandte Themen

- [Erstellen eines RSA-Schlüsselpaares](#)
- [Erstellen und signieren Sie ein Registrierungstoken](#)
- [Registrieren Sie den öffentlichen Schlüssel beim HSM](#)
- [Verwaltung der Quorum-Authentifizierung \(M-von-N-Zugriffskontrolle\)](#)
- [Quorum-Authentifizierung und 2FA](#)
- [listUsers](#)

Interagieren Sie mit einem HSM in einem AWS CloudHSM Cluster mithilfe von CMU

Verwenden Sie den `server` Befehl in der Datei AWS CloudHSM `cloudhsm_mgmt_util`, um in den Servermodus zu wechseln und direkt mit einer bestimmten HSM-Instanz (Hardware Security Module) zu interagieren.

Wenn Sie einen Befehl in `cloudhsm_mgmt_util` ausführen, wirkt sich der Befehl normalerweise auf alle im angegebenen Cluster aus (globaler Modus). HSMs Unter bestimmten Umständen müssen Sie auf einem einzelnen HSM Befehle ausgeben. Sollte beispielsweise die automatische Synchronisierung fehlschlagen, müssen Sie möglicherweise Schlüssel und Benutzer auf einem HSM synchronisieren, um die Konsistenz im Cluster zu wahren.

Nach erfolgreicher Initiierung wird die `aws-cloudhsm >`-Eingabeaufforderung durch die `server >`-Eingabeaufforderung ersetzt.

Um den Servermodus zu beenden, verwenden Sie den Befehl `exit`. Nach erfolgreicher Beendigung kehren Sie zur `cloudhsm_mgmt_util`-Eingabeaufforderung zurück.

Bevor Sie einen `cloudhsm_mgmt_util`-Befehl ausführen, müssen Sie `cloudhsm_mgmt_util` starten.

Benutzertyp

Die folgenden Benutzer können diesen Befehl ausführen.

- Alle Benutzer.

Voraussetzungen

Um den Servermodus aufrufen zu können, muss Ihnen die Servernummer des Ziel-HSM bekannt sein. Die Servernummern werden in der Trace-Ausgabe aufgelistet, die von der `cloudhsm_mgmt_util`

bei der Initiierung generiert wird. Servernummern werden in derselben Reihenfolge zugewiesen, in der sie in der Konfigurationsdatei erscheinen. HSMs In diesem Beispiel gehen wir davon aus, dass `server 0` der Server ist, der dem gewünschten HSM entspricht.

Syntax

So starten Sie den Servermodus:

```
server <server-number>
```

So beenden Sie den Servermodus:

```
server> exit
```

Beispiel

Dieser Befehl ruft den Servermodus auf einem HSM mit der Servernummer 0 auf.

```
aws-cloudhsm> server 0  
  
Server is in 'E2' mode...
```

Um den Servermodus zu beenden, verwenden Sie den Befehl `exit`.

```
server0> exit
```

Argumente

```
server <server-number>
```

<server-number>

Gibt die Servernummer des Ziel-HSM an.

Erforderlich: Ja

Für den Befehl `exit` gibt es keine Argumente.

Verwandte Themen

- [syncKey](#)
- [createUser](#)
- [deleteUser](#)

Legen Sie die Attribute von AWS CloudHSM Schlüsseln mit CMU fest

Verwenden Sie den `setAttribute` Befehl in AWS CloudHSM `cloudhsm_mgmt_util`, um den Wert der Attribute `label`, `encrypt`, `decrypt`, `wrap` und `unwrap` eines Schlüssels in der zu ändern. HSMs Sie können auch den Befehl [setAttribute](#) in `key_mgmt_util` verwenden, um einen Sitzungsschlüssel in einen dauerhaften Schlüssel umzuwandeln. Sie können nur die Attribute der Schlüssel ändern, deren Eigentümer Sie sind.

Vor dem Ausführen eines CMU-Befehls müssen Sie die CMU starten und sich beim HSM anmelden. Stellen Sie sicher, dass Sie sich mit einem Benutzertyp anmelden, der die Befehle ausführen kann, die Sie verwenden möchten.

Wenn Sie sie hinzufügen oder löschen, aktualisieren Sie die Konfigurationsdateien für CMU. HSMs Andernfalls sind die Änderungen, die Sie vornehmen, möglicherweise nicht für alle HSMs Mitglieder des Clusters wirksam.

Benutzertyp

Die folgenden Benutzer können diesen Befehl ausführen.

- Crypto-Benutzer (Crypto User, CU)

Syntax

Da dieser Befehl keine benannten Parameter hat, müssen Sie die Argumente in der im Syntaxdiagramm angegebenen Reihenfolge eingeben.

```
setAttribute <key handle> <attribute id>
```

Beispiel

Das folgende Beispiel zeigt, wie die Entschlüsselungsfunktionen eines symmetrischen Schlüssels deaktiviert werden. Sie können einen Befehl wie diesen verwenden, um einen Verpackungsschlüssel

zu konfigurieren, der andere Schlüssel verpacken und entpacken, aber nicht Daten verschlüsseln oder entschlüsseln kann.

Der erste Schritt ist das Erstellen des Verpackungsschlüssels. Dieser Befehl verwendet [genSymKey](#) in `key_mgmt_util`, um einen symmetrischen 256-Bit-AES-Schlüssel zu generieren. Die Ausgabe zeigt, dass der neue Schlüssel das Schlüssel-Handle 14 aufweist.

```
$ 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
```

Als nächstes werden wir den aktuellen Wert des Entschlüsselungsattributs bestätigen. Um die Attribut-ID des Entschlüsselungsattributs abzurufen, verwenden Sie [listAttributes](#). Die Ausgabe zeigt, dass die Konstante, die das OBJ_ATTR_DECRYPT-Attribut darstellt, 261 lautet. Hilfe zur Interpretation der Schlüsselattribute finden Sie unter [AWS CloudHSM Referenz zu Schlüsselattributen für KMU](#).

```
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

```

Um den aktuellen Wert des Entschlüsselungsattributs für Schlüssel 14 aufzurufen, verwendet der nächste Befehl [getAttribute](#) in `cloudhsm_mgmt_util`.

Die Ausgabe zeigt, dass der Wert des Decrypt-Attributs für beide im Cluster wahr (1) ist. HSMs

```

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

```

Dieser Befehl verwendet `setAttribute`, um den Wert des Entschlüsselungsattributs (Attribut 261) des Schlüssels 14 auf 0 zu ändern. Dies deaktiviert die Entschlüsselungsfunktion des Schlüssels.

Die Ausgabe zeigt, dass der Befehl auf beiden HSMs Clustern erfolgreich war.

```

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)
```

Der letzte Befehl wiederholt den Befehl `getAttribute`. Es wird wieder das Entschlüsselungsattribut (Attribut 261) des Schlüssels 14 abgerufen.

Diesmal zeigt die Ausgabe, dass der Wert des Decrypt-Attributs für beide HSMs im Cluster falsch (0) ist.

```
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
```

Argumente

```
setAttribute <key handle> <attribute id>
```

<key-handle>

Gibt das Schlüssel-Handle eines Schlüssels an, dessen Eigentümer Sie sind. Sie können nur jeweils einen Schlüssel in den einzelnen Befehlen angeben. Um das Schlüsselhandle eines Schlüssels zu erhalten, verwenden Sie [findKey](#) in `key_mgmt_util`. Um die Benutzer eines Schlüssels zu finden, verwenden Sie [getKeyInfo](#).

Erforderlich: Ja

<attribute id>

Gibt die Konstante an, die das Attribut darstellt, das Sie ändern möchten. Sie können nur jeweils ein Attribut in den einzelnen Befehlen angeben. Verwenden Sie [listAttributes](#), um die Attribute und deren Ganzzahlwerte abzurufen. Hilfe zur Interpretation der Schlüsselattribute finden Sie unter [AWS CloudHSM Referenz zu Schlüsselattributen für KMU](#).

Zulässige Werte:

- 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.

Erforderlich: Ja

Verwandte Themen

- [setAttribute](#) in key_mgmt_util
- [getAttribute](#)
- [listAttributes](#)
- [Schlüsselattributreferenz](#)

Verlassen Sie die CMU

Verwenden Sie den quit Befehl in der Datei cloudhsm_mgmt_util, um die AWS CloudHSM Datei cloudhsm_mgmt_util zu beenden. Jeder beliebige Benutzer kann unabhängig vom Typ diesen Befehl verwenden.

Bevor Sie einen cloudhsm_mgmt_util-Befehl ausführen, müssen Sie cloudhsm_mgmt_util starten.

Benutzertyp

Die folgenden Benutzer können diesen Befehl ausführen.

- Alle Benutzer. Sie müssen nicht angemeldet sein, um diesen Befehl auszuführen.

Syntax

```
quit
```

Beispiel

Dieser Befehl beendet cloudhsm_mgmt_util. Nach dem erfolgreichen Abschluss wechseln Sie zurück zu Ihrer regulären Befehlszeile. Dieser Befehl hat keine Ausgabeparameter.

```
aws-cloudhsm> quit  
  
disconnecting from servers, please wait...
```

Verwandte Themen

- [Erste Schritte mit cloudhsm_mgmt_util](#)

AWS CloudHSM Schlüssel mit CMU teilen

Verwenden Sie den shareKey Befehl in der Datei AWS CloudHSM cloudhsm_mgmt_util, um Schlüssel, die Sie besitzen, mit anderen Krypto-Benutzern zu teilen und deren Freigabe aufzuheben. Nur der Schlüsselbesitzer kann einen Schlüssel freigeben und die Freigabe aufheben. Sie können einen Schlüssel auch freigeben, wenn Sie ihn erstellen.

Benutzer, für die der Schlüssel freigegeben ist, können ihn in kryptografischen Vorgänge verwenden, ihn jedoch nicht löschen, exportieren, freigeben bzw. seine Freigabe aufheben oder seine Attribute ändern. Wenn die Quorum-Authentifizierung für einen Schlüssel aktiviert ist, muss das Quorum alle Operationen genehmigen, die den Schlüssel freigeben bzw. seine Freigabe aufheben.

Vor dem Ausführen eines CMU-Befehls müssen Sie die CMU starten und sich beim HSM anmelden. Stellen Sie sicher, dass Sie sich mit einem Benutzertyp anmelden, der die Befehle ausführen kann, die Sie verwenden möchten.

Aktualisieren Sie die Konfigurationsdateien für CMU, wenn Sie sie hinzufügen oder löschen. HSMs. Andernfalls sind die Änderungen, die Sie vornehmen, möglicherweise nicht für alle HSMs Mitglieder des Clusters wirksam.

Benutzertyp

Die folgenden Benutzertypen können diesen Befehl ausführen.

- Crypto-Benutzer (Crypto User, CU)

Syntax

Da dieser Befehl keine benannten Parameter hat, müssen Sie die Argumente in der im Syntaxdiagramm angegebenen Reihenfolge eingeben.

Benutzertyp: Crypto-Benutzer (CU)

```
shareKey <key handle> <user id> <(share/unshare key?) 1/0>
```

Beispiel

Die folgenden Beispiele zeigen, wie Sie mit shareKey-Schlüssel, die Sie besitzen, für andere Crypto-Benutzer freigeben bzw. ihre Freigabe aufheben.

Example: Freigeben eines Schlüssels

In diesem Beispiel wird ein [privater ECC-Schlüssel](#), den der aktuelle Benutzer besitzt, mit einem anderen Krypto-Benutzer auf dem HSMs gemeinsam genutztshareKey. Öffentliche Schlüssel stehen allen Benutzern des HSM zur Verfügung, sodass Sie diese weder freigeben noch deren Freigabe aufheben können.

Der erste Befehl verwendet [getKeyInfo](#), um die Benutzerinformationen für den Schlüssel abzurufen262177, einen privaten ECC-Schlüssel auf dem. HSMs

Die Ausgabe zeigt, dass der Schlüssel 262177 im Besitz des Benutzers 3, aber nicht freigegeben ist.

```
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
```

Dieser Befehl dient `shareKey` zur gemeinsamen Nutzung des Schlüssels 262177 mit dem Benutzer4, einem anderen Krypto-Benutzer auf dem HSMs. Das letzte Argument verwendet den Wert 1, um eine Freigabeoperation anzugeben.

Die Ausgabe zeigt, dass der Vorgang auf beiden HSMs Clustern erfolgreich war.

```
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)
```

Um zu überprüfen, ob die Operation erfolgreich war, wird im Beispiel der erste Befehl `getKeyInfo` wiederholt.

Die Ausgabe zeigt, dass der Schlüssel 262177 jetzt für den Benutzer 4 freigegeben ist.

```
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):
```

Example: Aufheben der Freigabe eines Schlüssels

In diesem Beispiel wird gezeigt, wie die Freigabe für einen symmetrischen Schlüssel aufgehoben wird, d. h., ein Crypto-Benutzer wird aus der Liste der für den Schlüssel freigegebenen Benutzer entfernt.

Dieser Befehl verwendet `shareKey`, um den Benutzer 4 aus der Liste der für den Schlüssel 6 freigegebenen Benutzer zu entfernen. Das letzte Argument verwendet den Wert `0`, um eine Operation zum Aufheben der Freigabe anzugeben.

Die Ausgabe zeigt, dass der Befehl in beiden Fällen erfolgreich war HSMS. Dies hat zur Folge, dass der Benutzer 4 den Schlüssel 6 nicht mehr in kryptografischen Operationen verwenden kann.

```
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)
```

Argumente

Da dieser Befehl keine benannten Parameter hat, müssen Sie die Argumente in der im Syntaxdiagramm angegebenen Reihenfolge eingeben.

```
shareKey <key handle> <user id> <(share/unshare key?) 1/0>
```

<key-handle>

Gibt das Schlüssel-Handle eines Schlüssels an, dessen Eigentümer Sie sind. Sie können nur jeweils einen Schlüssel in den einzelnen Befehlen angeben. Um das Schlüsselhandle eines Schlüssels zu erhalten, verwenden Sie [findKey](#) in `key_mgmt_util`. Um zu überprüfen, ob Sie einen Schlüssel besitzen, verwenden Sie [getKeyInfo](#).

Erforderlich: Ja

<user id>

Gibt die Benutzer-ID des Crypto-Benutzers (CU) an, für den Sie den Schlüssel freigeben bzw. dessen Freigabe aufheben. Verwenden Sie zur Suche nach der Benutzer-ID eines Benutzers den Befehl [listUsers](#).

Erforderlich: Ja

<share 1 or unshare 0>

Geben Sie zum Freigeben des Schlüssels für den angegebenen Benutzer 1 ein. Um die Freigabe des Schlüssels aufzuheben, d. h. zum Entfernen des angegebenen Benutzers aus der Liste der gemeinsamen Benutzer für den Schlüssel, geben Sie 0 ein.

Erforderlich: Ja

Verwandte Themen

- [getKeyInfo](#)

Synchronisieren Sie Schlüssel im gesamten AWS CloudHSM Cluster mithilfe von CMU

Verwenden Sie den syncKey Befehl in AWS CloudHSM cloudhsm_mgmt_util, um Schlüssel zwischen HSM-Instanzen innerhalb eines Clusters oder zwischen geklonten Clustern manuell zu synchronisieren. Im Allgemeinen müssen Sie diesen Befehl nicht verwenden, da HSM-Instances in einem Cluster Schlüssel automatisch synchronisieren. Allerdings muss die Synchronisierung von Schlüsseln für geklonte Cluster manuell durchgeführt werden. Geklonte Cluster werden normalerweise in verschiedenen AWS Regionen erstellt, um die globale Skalierung und Disaster Recovery-Prozesse zu vereinfachen.

Sie können syncKey nicht verwenden, um Schlüssel für beliebige Cluster zu synchronisieren: einer der Cluster muss aus einer Sicherung des anderen erstellt worden sein. Außerdem müssen beide Cluster über konsistente CO- und CU-Anmeldeinformationen verfügen, damit die Operation erfolgreich durchgeführt werden kann. Weitere Informationen finden Sie unter [HSM-Benutzer](#).

Zur Verwendung syncKey müssen Sie zunächst [eine AWS CloudHSM Konfigurationsdatei erstellen](#), die ein HSM aus dem Quellcluster und eines aus dem Zielcluster angibt. Auf diese Weise kann cloudhsm_mgmt_util die Verbindung mit beiden HSM-Instances herstellen. Verwenden Sie diese Konfigurationsdatei, um cloudhsm_mgmt_util zu starten. Dann melden Sie sich mit den

Anmeldeinformationen eines CO oder CU an, der die Schlüssel besitzt, die Sie synchronisieren möchten.

Benutzertyp

Die folgenden Benutzertypen können diesen Befehl ausführen.

- Verschlüsselungsverantwortliche (Crypto Officers, CO)
- Crypto-Benutzer (Crypto User, CU)

Note

COs kann für alle Schlüssel verwendet `syncKey` werden, während dieser Befehl nur für Schlüssel verwendet werden CUs kann, deren Eigentümer sie sind. Weitere Informationen finden Sie unter [the section called “Benutzertypen”](#).

Voraussetzungen

Bevor Sie beginnen, müssen Sie den `key handle` des Schlüssels im Quell-HSM kennen, der mit dem Ziel-HSM synchronisiert werden soll. Um den `key handle` zu suchen, verwenden Sie den [listUsers](#)-Befehl für das Auflisten aller Kennungen für benannte Benutzer. Verwenden Sie dann den [findAllKeys](#)-Befehl, um alle Schlüssel zu finden, die einem bestimmten Benutzer gehören.

Sie müssen auch die der Quelle und das Ziel `server IDs` zugewiesenen Daten kennen, die in der Trace-Ausgabe angezeigt werden HSMs, die `cloudhsm_mgmt_util` bei der Initiierung zurückgibt. Diese werden in derselben Reihenfolge zugewiesen, in der sie in der Konfigurationsdatei erscheinen. HSMs

Folgen Sie den Anweisungen in [CMU über geklonte Cluster hinweg verwenden](#) und initialisieren Sie `cloudhsm_mgmt_util` mit der neuen Konfigurationsdatei. Anschließend wechseln Sie in den Servermodus im Quell-HSM, indem Sie den [server](#)-Befehl ausführen.

Syntax

Note

Um `syncKey` auszuführen, wechseln Sie zuerst in den Servermodus im HSM, der den zu synchronisierenden Schlüssel enthält.

Da dieser Befehl keine benannten Parameter hat, müssen Sie die Argumente in der im Syntaxdiagramm angegebenen Reihenfolge eingeben.

Benutzertyp: Crypto-Benutzer (CU)

```
syncKey <key handle> <destination hsm>
```

Beispiel

Führen Sie den Server-Befehl aus, um sich beim Quell-HSM anzumelden und in den Servermodus zu wechseln. In diesem Beispiel gehen wir davon aus, dass `server 0` der Quell-HSM ist.

```
aws-cloudhsm> server 0
```

Führen Sie jetzt den `syncKey`-Befehl aus. In diesem Beispiel gehen wir davon aus, dass der Schlüssel 261251 mit `server 1` synchronisiert werden soll.

```
aws-cloudhsm> syncKey 261251 1  
syncKey success
```

Argumente

Da dieser Befehl keine benannten Parameter hat, müssen Sie die Argumente in der im Syntaxdiagramm angegebenen Reihenfolge eingeben.

```
syncKey <key handle> <destination hsm>
```

<key handle>

Gibt den Key-Handle des zu synchronisierenden Schlüssels an. Sie können nur jeweils einen Schlüssel in den einzelnen Befehlen angeben. Um das Schlüssel-Handle eines Schlüssels abzurufen, verwenden Sie, [findAllKeys](#) während Sie an einem HSM-Server angemeldet sind.

Erforderlich: Ja

<destination hsm>

Gibt die Anzahl der Server an, zu denen Sie einen Schlüssel synchronisieren.

Erforderlich: Ja

Verwandte Themen

- [listUsers](#)
- [findAllKeys](#)
- [Beschreiben](#) Sie Cluster in AWS CLI
- [server](#)

Synchronisieren Sie Benutzer im gesamten AWS CloudHSM Cluster mithilfe von CMU

Verwenden Sie den `syncUser` Befehl in AWS CloudHSM `cloudhsm_mgmt_util`, um Crypto-Benutzer () oder Crypto Officers () manuell zwischen HSM-Instanzen innerhalb eines Clusters oder zwischen geklonten Clustern zu synchronisieren. CUs COs AWS CloudHSM synchronisiert Benutzer nicht automatisch. Im Allgemeinen verwalten Sie Benutzer im globalen Modus, sodass alle Benutzer HSMs in einem Cluster zusammen aktualisiert werden. Möglicherweise müssen Sie `syncUser` verwenden, wenn ein HSM versehentlich desynchronisiert wurde (z. B. aufgrund von Passwortänderungen) oder wenn Sie Benutzeranmeldeinformationen über geklonte Cluster hinweg rotieren möchten. Geklonte Cluster werden normalerweise in verschiedenen AWS Regionen erstellt, um die globale Skalierung und die Notfallwiederherstellung zu vereinfachen.

Vor dem Ausführen eines CMU-Befehls müssen Sie die CMU starten und sich beim HSM anmelden. Stellen Sie sicher, dass Sie sich mit einem Benutzertyp anmelden, der die Befehle ausführen kann, die Sie verwenden möchten.

Aktualisieren Sie die Konfigurationsdateien für HSMs CMU, wenn Sie sie hinzufügen oder löschen. Andernfalls sind die Änderungen, die Sie vornehmen, möglicherweise nicht für alle HSMs Mitglieder des Clusters wirksam.

Benutzertyp

Die folgenden Benutzertypen können diesen Befehl ausführen.

- Verschlüsselungsverantwortliche (Crypto Officers, CO)

Voraussetzungen

Bevor Sie beginnen, müssen Sie das `user ID` des Benutzers auf dem Quell-HSM kennen, das mit dem Ziel-HSM synchronisiert werden soll. Um das zu finden `user ID`, verwenden Sie den Befehl [ListUsers](#), um alle Benutzer auf dem HSMs in einem Cluster aufzulisten.

Sie müssen auch die der Quelle und das Ziel `server ID` zugewiesenen Daten kennen, die in der Trace-Ausgabe angezeigt werden HSMs, die `cloudhsm_mgmt_util` bei der Initiierung zurückgibt. Diese werden in derselben Reihenfolge zugewiesen, in der sie in der Konfigurationsdatei erscheinen. HSMs

Wenn Sie HSMs zwischen geklonten Clustern synchronisieren, folgen Sie den Anweisungen unter [Using CMU Across Cloned Clusters](#) und initialisieren Sie `cloudhsm_mgmt_util` mit der neuen Konfigurationsdatei.

Wenn Sie zur Ausführung von `syncUser` bereit sind, wechseln Sie auf dem Quell-HSM mit dem Befehl [server](#) in den Servermodus.

Syntax

Da dieser Befehl keine benannten Parameter hat, müssen Sie die Argumente in der im Syntaxdiagramm angegebenen Reihenfolge eingeben.

```
syncUser <user ID> <server ID>
```

Beispiel

Führen Sie den Befehl `server` aus, um sich am Quell-HSM anzumelden und in den Servermodus zu wechseln. In diesem Beispiel gehen wir davon aus, dass `server 0` der Quell-HSM ist.

```
aws-cloudhsm> server 0
```

Führen Sie nun den Befehl `syncUser` aus. In diesem Beispiel gehen wir davon aus, dass der Benutzer 6 der zu synchronisierende Benutzer und `server 1` der Ziel-HSM ist.

```
server 0> syncUser 6 1
ExtractMaskedObject: 0x0 !
InsertMaskedObject: 0x0 !
syncUser success
```

Argumente

Da dieser Befehl keine benannten Parameter hat, müssen Sie die Argumente in der im Syntaxdiagramm angegebenen Reihenfolge eingeben.

```
syncUser <user ID> <server ID>
```

<user ID>

Gibt die ID des zu synchronisierenden Benutzers an. Sie können in jedem Befehl nur einen Benutzer angeben. Um die ID eines Benutzers zu erhalten, verwenden Sie [listUsers](#).

Erforderlich: Ja

<server ID>

Gibt die Servernummer des HSM an, mit dem Sie einen Benutzer synchronisieren.

Erforderlich: Ja

Verwandte Themen

- [listUsers](#)
- [Beschreiben Sie Cluster in AWS CLI](#)
- [server](#)

AWS CloudHSM Schlüsselverwaltungsprogramm (KMU)

Das Key Management Utility (KMU) ist ein Befehlszeilentool AWS CloudHSM, das Krypto-Benutzern (CU) bei der Verwaltung von Schlüsseln auf den Hardware-Sicherheitsmodulen (HSM) hilft. Ein KMU umfasst mehrere Befehle, die Schlüssel erzeugen, löschen, importieren und exportieren, Attribute abrufen und setzen, Schlüssel finden und kryptografische Operationen durchführen.

KMU und CMU sind Teil [der Client-SDK-3-Suite](#).

Informationen zum schnellen Einstieg finden Sie unter [Erste Schritte mit AWS CloudHSM key_mgmt_util](#). Ausführliche Informationen zu den Befehlen finden Sie unter [Referenz für Befehle AWS CloudHSM der Key Management Utility](#). Hilfe zur Interpretation der Schlüsselattribute finden Sie unter [AWS CloudHSM Referenz zu Schlüsselattributen für KMU](#).

Um key_mgmt_util unter Linux zu verwenden, stellen Sie eine Verbindung zur Client-Instance her. Beachten Sie dann [Installieren und konfigurieren Sie den AWS CloudHSM Client für KMU \(Linux\)](#). Wenn Sie Windows verwenden, siehe [Installieren und konfigurieren Sie den AWS CloudHSM Client für KMU \(Windows\)](#).

Themen

- [Erste Schritte mit AWS CloudHSM key_mgmt_util](#)
- [Installieren und konfigurieren Sie den AWS CloudHSM Client für KMU \(Linux\)](#)
- [Installieren und konfigurieren Sie den AWS CloudHSM Client für KMU \(Windows\)](#)
- [Referenz für Befehle AWS CloudHSM der Key Management Utility](#)

Erste Schritte mit AWS CloudHSM key_mgmt_util

AWS CloudHSM [beinhaltet zwei Befehlszeilentools in der Client-Software.](#) Das [cloudhsm_mgmt_util](#)-Tool enthält Befehle zum Verwalten von HSM-Benutzern. Das [key_mgmt_util](#)-Tool enthält Befehle zum Verwalten von Schlüsseln. Weitere Informationen über die ersten Schritte mit dem key_mgmt_util-Befehlszeilen-Tool finden Sie in den folgenden Themen.

Themen

- [Richten Sie AWS CloudHSM key_mgmt_util ein](#)
- [Melden Sie sich mit KMU bei einem Cluster an HSMs AWS CloudHSM](#)
- [Melden Sie sich mithilfe von KMU vom HSMs in einem AWS CloudHSM Cluster ab](#)
- [AWS CloudHSM Stoppen Sie key_mgmt_util](#)

Wenn Sie eine Fehlermeldung oder ein unerwartetes Ergebnis für einen Befehl erhalten, finden Sie in den [Problembehebung AWS CloudHSM](#)-Themen Hilfe. Einzelheiten zu den Befehlen key_mgmt_util finden Sie unter [Referenz für Befehle AWS CloudHSM der Key Management Utility](#).

Richten Sie AWS CloudHSM key_mgmt_util ein

Schließen Sie das folgende Setup ab, bevor Sie key_mgmt_util (KMU) verwenden. AWS CloudHSM

Themen

- [Schritt 1. AWS CloudHSM Starten Sie den Client](#)
- [Schritt 2. key_mgmt_util starten](#)

Schritt 1. AWS CloudHSM Starten Sie den Client

Bevor Sie key_mgmt_util verwenden, müssen Sie den Client starten. AWS CloudHSM Der Client ist ein Daemon, der eine end-to-end verschlüsselte Kommunikation mit dem in Ihrem Cluster herstellt. HSMs Das Tool key_mgmt_util verwendet die Client-Verbindung, um mit dem in Ihrem Cluster zu kommunizieren. HSMs Ohne diese Verbindung funktioniert key_mgmt_util nicht.

Um AWS CloudHSM den Client zu starten

Verwenden Sie den folgenden Befehl, um den AWS CloudHSM Client zu starten.

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

- Für Windows-Client 1.1.2 und höher:


```
C:\Program Files\Amazon\CloudHSM>net.exe start AWSCloudHSMClient
```

- Für Windows-Clients 1.1.1 und früher:

```
C:\Program Files\Amazon\CloudHSM>start "cloudhsm_client" cloudhsm_client.exe C:\ProgramData\Amazon\CloudHSM\data\cloudhsm_client.cfg
```

Schritt 2. key_mgmt_util starten

Nachdem Sie den AWS CloudHSM Client gestartet haben, verwenden Sie den folgenden Befehl, um key_mgmt_util zu starten.

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"
```

Wenn `key_mgmt_util` ausgeführt wird, ändert sich die Eingabeaufforderung in `Command:`.

Wenn der Befehl fehlschlägt, also z. B. die Meldung `Daemon socket connection error` zurückgegeben wird, versuchen Sie eine [Aktualisierung Ihrer Konfigurationsdatei](#).

Melden Sie sich mit KMU bei einem Cluster an HSMs AWS CloudHSM

Verwenden Sie den `loginHSM` Befehl in `key_mgmt_util` (KMU), um sich bei den Hardware-Sicherheitsmodulen (HSM) in einem Cluster anzumelden. AWS CloudHSM Mit dem folgenden Befehl melden Sie sich als [Crypto-Benutzer \(CU, Crypto User\)](#) mit dem Namen `example_user` an. Die Ausgabe zeigt eine erfolgreiche Anmeldung für alle drei Mitglieder des Clusters an. HSMs

```
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
```

Nachfolgend finden Sie die allgemeine Syntax für den Befehl `loginHSM`.

```
Command: loginHSM -u <USER TYPE> -s <USERNAME> -p <PASSWORD>
```

Melden Sie sich mithilfe von KMU vom HSMs in einem AWS CloudHSM Cluster ab

Verwenden Sie den `logoutHSM` Befehl in `key_mgmt_util` (KMU), um sich von den Hardware-Sicherheitsmodulen (HSM) in einem Cluster abzumelden. AWS CloudHSM

```
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 Stoppen Sie `key_mgmt_util`

Verwenden Sie den Befehl, um `key_mgmt_util` zu beenden `exit`. AWS CloudHSM

```
Command:  exit
```

Installieren und konfigurieren Sie den AWS CloudHSM Client für KMU (Linux)

Um mit dem Hardware-Sicherheitsmodul (HSM) in Ihrem AWS CloudHSM Cluster mithilfe von `key_mgmt_util` (KMU) zu interagieren, benötigen Sie die Client-Software für Linux. AWS CloudHSM Sie sollten die Software auf der Linux-EC2-Client-Instance installieren, die Sie zuvor erstellt haben. Sie können auch einen Client für Windows installieren. Weitere Informationen finden Sie unter [Installieren und konfigurieren Sie den AWS CloudHSM Client für KMU \(Windows\)](#).

Aufgaben

- [Schritt 1. AWS CloudHSM Installieren Sie den Client und die Befehlszeilentools](#)
- [Schritt 2. Bearbeiten der Clientkonfiguration](#)

Schritt 1. AWS CloudHSM Installieren Sie den Client und die Befehlszeilentools

Connect zu Ihrer Client-Instance her und führen Sie die folgenden Befehle aus, um den AWS CloudHSM Client und die Befehlszeilentools herunterzuladen und zu installieren.

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
```

Schritt 2. Bearbeiten der Clientkonfiguration

Bevor Sie den AWS CloudHSM Client verwenden können, um eine Verbindung zu Ihrem Cluster herzustellen, müssen Sie die Client-Konfiguration bearbeiten.

Bearbeiten der Clientkonfiguration

1. Kopieren Sie Ihr ausstellendes Zertifikat – [das Zertifikat, mit dem Sie das Zertifikat des Clusters signiert haben](#) – an den folgenden Speicherort in der Client-Instance: `/opt/cloudhsm/etc/`

customerCA.crt. Sie müssen Root-Berechtigungen für die Client-Instance haben, um Ihr Zertifikat an diesen Speicherort zu kopieren.

2. Verwenden Sie den folgenden [Konfigurationsbefehl](#), um die Konfigurationsdateien für den AWS CloudHSM Client und die Befehlszeilentools zu aktualisieren. Geben Sie dabei die IP-Adresse des HSM in Ihrem Cluster an. Um die IP-Adresse des HSM abzurufen, zeigen Sie Ihren Cluster in der [AWS CloudHSM Konsole](#) an oder führen Sie den [describe-clusters](#) AWS CLI Befehl aus. Die IP-Adresse des HSM ist in der Ausgabe des Befehls der Wert des Feldes EniIp. Wenn Sie mehr als ein HSM haben, wählen Sie die IP-Adresse für eines der beiden HSMs; es spielt keine Rolle, welche.

```
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. Wechseln Sie zu [Aktiviere den Cluster in AWS CloudHSM](#).

Installieren und konfigurieren Sie den AWS CloudHSM Client für KMU (Windows)

Um mit einem Hardware-Sicherheitsmodul (HSM) in Ihrem AWS CloudHSM Cluster unter Windows unter Verwendung von key_mgmt_util (KMU) arbeiten zu können, benötigen Sie die Client-Software für Windows. AWS CloudHSM Sie sollten die Software auf der Windows Server-Instance installieren, die Sie zuvor erstellt haben.

So installieren (oder aktualisieren) Sie die den neuesten Windows-Client und die Befehlszeilen-Tools

1. Stellen Sie eine Verbindung zur Windows Server-Instance her.
2. [Laden Sie die neueste Version \(.msi\) von der Downloadseite herunter. AWSCloudHSMClient-latest](#)
3. Gehen Sie zu Ihrem Download-Speicherort und führen Sie das Installationsprogramm (AWSCloudHSMClient-latest.msi) mit Administratorrechten aus.
4. Folgen Sie den Anweisungen des Installationsprogramms und wählen Sie dann Schließen, wenn das Installationsprogramm abgeschlossen ist.
5. Kopieren Sie das selbstsignierte Ausstellerzertifikat – [das Zertifikat, mit dem Sie das Clusterzertifikat signiert haben](#) – in den Ordner C:\ProgramData\Amazon\CloudHSM.

6. Führen Sie den folgenden Befehl aus, um die Konfigurationsdateien zu aktualisieren. Beim Aktualisieren muss der Client während der Neukonfiguration angehalten und dann wieder gestartet werden.

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

7. Wechseln Sie zu [Aktiviere den Cluster in AWS CloudHSM](#).

Hinweise:

- Bei der Aktualisierung des Clients werden vorhandene Konfigurationsdateien aus früheren Installationen nicht überschrieben.
- Das AWS CloudHSM Client-Installationsprogramm für Windows registriert automatisch die Cryptography API: Next Generation (CNG) und Key Storage Provider (KSP). Um den Client zu deinstallieren, führen Sie das Installationsprogramm erneut aus und folgen Sie den Anweisungen zur Deinstallation.
- Wenn Sie Linux verwenden, können Sie den Linux-Client installieren. Weitere Informationen finden Sie unter [Installieren und konfigurieren Sie den AWS CloudHSM Client für KMU \(Linux\)](#).

Referenz für Befehle AWS CloudHSM der Key Management Utility

Das Befehlszeilentool `key_mgmt_util` unterstützt Sie bei der Verwaltung von Schlüsseln in den Hardware-Sicherheitsmodulen (HSM) in Ihrem AWS CloudHSM Cluster, einschließlich der Erstellung, Löschung und Suche nach Schlüsseln und deren Attributen. Das Tool umfasst mehrere Befehle, die in diesem Thema ausführlich beschrieben werden.

Informationen zum schnellen Einstieg finden Sie unter [Erste Schritte mit AWS CloudHSM key_mgmt_util](#). Hilfe zur Interpretation der Schlüsselattribute finden Sie unter [AWS CloudHSM Referenz zu Schlüsselattributen für KMU](#). Weitere Informationen über das `cloudhsm_mgmt_util`-Befehlszeilentool, das Befehle für die Verwaltung von HSM und Benutzern in Ihrem Cluster umfasst, finden Sie unter [AWS CloudHSM Verwaltungsdienstprogramm \(CMU\)](#).

Bevor Sie einen `key_mgmt_util`-Befehl ausführen, müssen Sie [key_mgmt_util starten](#) und sich am HSM als Crypto-Benutzer (CU) [anmelden](#).

Um alle `key_mgmt_util`-Befehle aufzulisten, geben Sie Folgendes ein:

```
Command: help
```

Um Hilfe für einen bestimmten `key_mgmt_util`-Befehl zu erhalten, geben Sie Folgendes ein:

Command: **<command-name> -h**

Um Ihre `key_mgmt_util`-Sitzung zu beenden, geben Sie Folgendes ein:

Command: **exit**

Die folgenden Themen beschreiben Befehle in `key_mgmt_util`.

Note

Einige Befehle in `key_mgmt_util` und `cloudhsm_mgmt_util` haben dieselben Namen. Die Befehle haben jedoch in der Regel eine andere Syntax, eine andere Ausgabe und eine leicht unterschiedliche Funktionalität.

Befehl	Description
<code>aesWrapUnwrap</code>	Verschlüsselt und entschlüsselt den Inhalt eines Schlüssels in einer Datei.
<code>deleteKey</code>	HSMsLöscht einen Schlüssel aus dem.
<code>Error2String</code>	Gibt den Fehler zurück, der dem hexadezimalen <code>key_mgmt_util</code> -Fehlercode entspricht.
<code>exit</code>	Beendet das <code>key_mgmt_util</code> .
<code>exportPrivateKey</code>	Exportiert die Kopie eines privaten Schlüssel s aus einem HSM in eine Datei auf einem Datenträger.
<code>exportPubKey</code>	Exportiert eine Kopie eines öffentlichen Schlüssels aus einem HSM in eine Datei.
<code>exSymKey</code>	Exportiert eine Klartextkopie eines symmetrischen Schlüssels von in eine Datei HSMs .

Befehl	Description
<u>extractMaskedObject</u>	Extrahiert einen Schlüssel aus einem HSM als Datei in Form eines maskierten Objekts.
<u>findKey</u>	Sucht nach Schlüsseln anhand des Schlüsselattributwerts.
<u>findSingleKey</u>	Überprüft, ob ein Schlüssel für alle HSMs im Cluster vorhanden ist.
<u>Gen-Paar DSAKey</u>	Generiert ein DSA-Schlüsselpaar (<u>Digital Signing Algorithm</u>) in Ihrem HSMs.
<u>Gen-Paar ECCKey</u>	Generiert ein ECC-Schlüsselpaar (<u>Elliptic Curve Cryptography</u>) in Ihrem. HSMs
<u>Gen-Paar RSAKey</u>	Generiert ein <u>asymmetrisches RSA-Schlüsselpaar</u> in Ihrem. HSMs
<u>genSymKey</u>	Generiert einen symmetrischen Schlüssel in Ihrem HSMs
<u>getAttribute</u>	Liest die Attributwerte eines AWS CloudHSM-Schlüssels und schreibt sie in eine Datei.
<u>getCaviumPrivKey</u> (Schlüssel)	Erstellt eine Fake-Version eines privaten Schlüssels im PEM-Format und exportiert ihn in eine Datei.
<u>getCert</u>	Ruft die Partitionszertifikate eines HSMs ab und speichert sie in einer Datei.
<u>getKeyInfo</u>	Ruft den HSM-Benutzer IDs der Benutzer ab, die den Schlüssel verwenden können. Wird der Schlüssel durch ein Quorum gesteuert , wird die Anzahl der Benutzer in dem Quorum gelesen.

Befehl	Description
help	Zeigt Hilfeinformationen zu den in key_mgmt_util verfügbaren Befehlen an.
importPrivateKey	Importiert einen privaten Schlüssel in ein HSM.
importPubKey	Importiert einen öffentlichen Schlüssel in ein HSM.
imSymKey	Importiert eine Klartextkopie eines symmetrischen Schlüssels von einer Datei in das HSM.
insertMaskedObject	Fügt ein maskiertes Objekt aus einer Datei auf einem Datenträger in ein HSM ein, das sich in einem dem ursprünglichen Objekt-Cluster zugehörigen Cluster befindet. Zusammgehörige Cluster sind alle aus einer Sicherung des ursprünglichen Clusters erstellten Cluster.
???	Ermittelt, ob eine bestimmte Datei einen echten privaten Schlüssel oder einen Beispiel-PEM-Schlüssel enthält.
listAttributes	Listet die Attribute eines AWS CloudHSM Schlüssels und die Konstanten auf, die sie repräsentieren.
listUsers	Ruft die Benutzer in HSMs, ihren Benutzertyp und ihre ID sowie andere Attribute ab.
loginHSM und logoutHSM	Melden Sie sich in einem Cluster HSMs an und ab.
setAttribute	Konvertiert einen Sitzungsschlüssel in einen persistenten Schlüssel.
sign	Erstellen Sie mit einem ausgewählten privaten Schlüssel eine Signatur für eine Datei.

Befehl	Description
unWrapKey	Importiert einen verpackten (verschlüsselten) Schlüssel aus einer Datei in die HSMs.
verify	Prüft, ob ein bestimmter Schlüssel zum Signieren einer bestimmten Datei verwendet wurde.
wrapKey	Exportiert eine verschlüsselte Kopie eines Schlüssels aus dem HSM in eine Datei.

Eine AWS CloudHSM Datei mit KMU verschlüsseln und entschlüsseln

Verwenden Sie den `aesWrapUnwrap` Befehl in AWS CloudHSM `key_mgmt_util`, um den Inhalt einer Datei auf der Festplatte zu verschlüsseln oder zu entschlüsseln. Mit diesem Befehl können Sie Verschlüsselungsschlüssel ein- und auspacken. Sie können ihn für jede Datei verwenden, die weniger als 4 KB (4096 Byte) Daten enthält.

`aesWrapUnwrap` verwendet AES Key Wrap mit PKCS #5 -Padding. Es verwendet einen AES-Schlüssel im HSM als Ver- und Entpackschlüssel. Dann schreibt er das Ergebnis in eine andere Datei auf dem Datenträger.

Bevor Sie einen `key_mgmt_util`-Befehl ausführen, müssen Sie [key_mgmt_util starten](#) und sich am HSM als Crypto-Benutzer (CU) [anmelden](#).

Syntax

```
aesWrapUnwrap -h

aesWrapUnwrap -m <wrap-unwrap mode>
               -f <file-to-wrap-unwrap>
               -w <wrapping-key-handle>
               [-i <wrapping-IV>]
               [-out <output-file>]
```

Beispiele

Diese Beispiele zeigen, wie man mit `aesWrapUnwrap` einen Verschlüsselungsschlüssel in einer Datei verschlüsselt und entschlüsselt.

Example: Verschlüsselungsschlüssel verpacken

Dieser Befehl verwendet `aesWrapUnwrap`, um einen symmetrischen Dreifach-DES-Schlüssel, der [vom HSM im Klartext exportiert wurde](#), in die `3DES.key`-Datei zu packen. Mit einem ähnlichen Befehl können Sie jeden in einer Datei gespeicherten Schlüssel verpacken.

Der Befehl verwendet den `-m`-Parameter mit dem Wert `1`, um den Wrap-Modus festzulegen. Er verwendet den `-w`-Parameter, um einen AES-Schlüssel im HSM (Key-Handle 6) als Wrapping-Schlüssel anzugeben. Der resultierende verpackte Schlüssel wird in die `3DES.key.wrapped`-Datei geschrieben.

Die Ausgabe zeigt, dass der Befehl erfolgreich war und dass die Operation die Standard-IV verwendet hat, die bevorzugt wird.

```
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: Verschlüsselungsschlüssel entpacken

Dieses Beispiel zeigt, wie man `aesWrapUnwrap` verwendet, um einen verpackten (verschlüsselten) Schlüssel in einer Datei zu entpacken (entschlüsseln). Sie können eine solche Operation durchführen, bevor Sie einen Schlüssel in das HSM importieren. Wenn Sie beispielsweise versuchen, den [imSymKey](#)-Befehl zum Importieren eines verschlüsselten Schlüssels zu verwenden, wird ein Fehler zurückgegeben, da der verschlüsselte Schlüssel nicht das Format hat, das für einen Klartext-Schlüssel dieses Typs erforderlich ist.

Der Befehl entpackt den Schlüssel in der Datei `3DES.key.wrapped` und schreibt den Klartext in die Datei `3DES.key.unwrapped`. Der Befehl verwendet den `-m`-Parameter mit dem Wert `0`, um den Unwrap-Modus festzulegen. Er verwendet den `-w`-Parameter, um einen AES-Schlüssel im HSM

(Key-Handle 6) als Wrapping-Schlüssel anzugeben. Der resultierende verpackte Schlüssel wird in die `3DES.key.unwrapped`-Datei geschrieben.

```
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
```

Parameters

-h

Zeigt Hilfe für den Befehl an.

Erforderlich: Ja

-m

Gibt den Modus an. Um den Dateiinhalt zu verpacken (verschlüsseln), geben Sie 1 ein. Um den Dateiinhalt zu entpacken (entschlüsseln), geben Sie 0 ein.

Erforderlich: Ja

-f

Gibt die zu verpackende Datei an. Geben Sie eine Datei ein, die weniger als 4 KB (4096 Byte) Daten enthält. Diese Operation wurde entwickelt, um Verschlüsselungsschlüssel ein- und auszupacken.

Erforderlich: Ja

-w

Gibt den Wrapping-Schlüssel an. Geben Sie das Schlüssel-Handle eines AES-Schlüssels auf dem HSM ein. Dieser Parameter muss angegeben werden. Verwenden Sie den [findKey](#)-Befehl, um Schlüssel-Handles zu finden.

Um einen Wrapping-Schlüssel zu erstellen, verwenden Sie, [genSymKey](#)um einen AES-Schlüssel zu generieren (Typ 31).

Erforderlich: Ja

-i

Gibt einen alternativen Initialwert (IV) für den Algorithmus an. Verwenden Sie den Standardwert, es sei denn, es gibt eine spezielle Bedingung, die eine Alternative erfordert.

Standard: 0xA6A6A6A6A6A6A6A6. Der Standardwert ist in der Spezifikation des [AES Key Wrap](#)-Algorithmus definiert.

Erforderlich: Nein

-out

Gibt einen alternativen Namen für die Ausgabedatei an, die den verpackten oder entpackten Schlüssel enthält. Die Voreinstellung ist `wrapped_key` (für Wrap-Operationen) und `unwrapped_key` (für Unwrap-Operationen) im lokalen Verzeichnis.

Wenn die Datei vorhanden ist, wird diese ohne Warnung von `aesWrapUnwrap` überschrieben. Wenn der Befehl fehlschlägt, erstellt `aesWrapUnwrap` eine Ausgabedatei ohne Inhalt.

Voreinstellung: Für Verpacken: `wrapped_key`. Zum Entpacken: `unwrapped_key`.

Erforderlich: Nein

Verwandte Themen

- [exSymKey](#)
- [imSymKey](#)
- [unWrapKey](#)
- [wrapKey](#)

Löschen Sie einen AWS CloudHSM Schlüssel mit KMU

Verwenden Sie den `deleteKey` Befehl in AWS CloudHSM `key_mgmt_util`, um einen Schlüssel aus dem Hardware-Sicherheitsmodul (HSM) in einem Cluster zu löschen. AWS CloudHSM Sie können nur jeweils einen Schlüssel löschen. Das Löschen eines Schlüssels in einem Schlüsselpaar hat keine Auswirkungen auf den anderen Schlüssel in diesem Paar.

Nur der Schlüsselbesitzer kann einen Schlüssel löschen. Benutzer, für die der Schlüssel freigegeben ist, können ihn in kryptografischen Vorgängen verwenden, aber nicht löschen.

Bevor Sie einen `key_mgmt_util`-Befehl ausführen, müssen Sie [key_mgmt_util starten](#) und sich am HSM als Crypto-Benutzer (CU) [anmelden](#).

Syntax

```
deleteKey -h
```

```
deleteKey -k
```

Beispiele

Diese Beispiele zeigen, wie Sie Schlüssel aus Ihrem löschen können. `deleteKey` HSMs

Example: Löschen eines Schlüssels

Dieser Befehl löscht den Schlüssel mit dem Schlüssel-Handle 6. Wenn der Befehl erfolgreich ist, gibt `deleteKey` eine Erfolgsmeldung von jedem HSM im Cluster zurück.

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: Löschen eines Schlüssels (Fehler)

Wenn der Befehl fehlschlägt, da kein Schlüssel über das angegebenen Schlüssel-Handle verfügt, gibt `deleteKey` eine Fehlermeldung aufgrund eines ungültigen Objekt-Handles zurück.

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
```

Wenn der Befehl fehlschlägt, da der aktuelle Benutzer nicht der Eigentümer des Schlüssels ist, gibt der Befehl eine „Zugriff verweigert“-Fehlermeldung zurück.

```
Command: deleteKey -k 262152
```

```
Cfm3DeleteKey returned: 0xc6 : HSM Error: Key Access is denied.
```

Parameters

-h

Zeigt die Befehlszeilenhilfe für den Befehl an.

Erforderlich: Ja

-k

Gibt das Schlüssel-Handle des zu löschenden Schlüssels an. Um die Schlüssel-Handles der Schlüssel im HSM zu finden, verwenden Sie [findKey](#).

Erforderlich: Ja

Verwandte Themen

- [findKey](#)

Beschreiben Sie einen AWS CloudHSM Fehler mit KMU

Verwenden Sie den Error2String Hilfsbefehl in key_mgmt_util, um den Fehler zurückzugeben, der einem AWS CloudHSM hexadezimalen Key_mgmt_util-Fehlercode entspricht. Verwenden Sie diesen Befehl bei der Fehlerbehebung von Befehlen und Skripten.

Bevor Sie einen key_mgmt_util-Befehl ausführen, müssen Sie [key_mgmt_util starten](#) und sich am HSM als Crypto-Benutzer (CU) [anmelden](#).

Syntax

```
Error2String -h
```



```
Error2String -r <response-code>
```

Beispiele

Diese Beispiele zeigen, wie Sie mit Error2String den Fehlerstring für einen key_mgmt_util-Fehlercode abrufen können.

Example; Anfordern einer Fehlerbeschreibung

Dieser Befehl ruft die Fehlerbeschreibung für den 0xdb-Fehlercode auf. Die Beschreibung erklärt, dass ein Anmeldeversuch bei key_mgmt_util fehlgeschlagen ist, da der Benutzer nicht über den richtigen Benutzertypen verfügt. Nur Crypto-Benutzer (CU) können sich bei key_mgmt_util anmelden.

```
Command: Error2String -r 0xdb
```

```
Error Code db maps to HSM Error: Invalid User Type.
```

Example: Finden des Fehlercodes

Dieses Beispiel zeigt, wo der Fehlercode in einem key_mgmt_util-Fehler gefunden werden kann. Der Fehlercode 0xc6 wird nach der Zeichenfolge angezeigt: Cfm3*<command-name>* returned: .

[getKeyInfo](#) zeigt in diesem Beispiel an, dass der aktuelle Benutzer (Benutzer 4) den Schlüssel für kryptografische Operationen verwenden kann. Wenn der Benutzer jedoch [deleteKey](#) verwendet, um den Schlüssel zu löschen, gibt der Befehl den Fehlercode 0xc6 zurück.

```
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):
```

Wenn Ihnen der Fehler 0xc6 zurückgegeben wird, können Sie einen Error2String-Befehl wie diesen verwenden, um den Fehler nachzuschlagen. In diesem Fall schlug der Befehl deleteKey aufgrund eines „Zugriff verweigert“-Fehlers fehl, da der Schlüssel für den aktuellen Benutzer freigegeben wurde, der Eigentümer aber ein anderer Benutzer ist. Nur die Eigentümer eines Schlüssels haben die Berechtigung, einen Schlüssel zu löschen.

```
Command: Error2String -r 0xa8
```

```
Error Code c6 maps to HSM Error: Key Access is denied
```

Parameters

-h

Zeigt Hilfe für den Befehl an.

Erforderlich: Ja

-r

Gibt einen hexadezimalen Fehlercode an. Die hexadezimalen Anzeige 0x ist erforderlich.

Erforderlich: Ja

Verlassen Sie die AWS CloudHSM KMU

Verwenden Sie den exit Befehl in AWS CloudHSM key_mgmt_util, um key_mgmt_util zu beenden. Nach erfolgreicher Beendigung kehren Sie zur Standard-Befehlszeile zurück.

Bevor Sie den Befehl key_mgmt_util ausführen, müssen Sie [key_mgmt_util starten](#).

Syntax

```
exit
```

Parameters

Für diesen Befehl gibt es keine Parameter.

Verwandte Themen

- [key_mgmt_util starten](#)

Exportieren eines privaten AWS CloudHSM Schlüssels mit KMU

Verwenden Sie den `exportPrivateKey` Befehl in AWS CloudHSM `key_mgmt_util`, um einen asymmetrischen privaten Schlüssel aus einem Hardware-Sicherheitsmodul (HSM) in eine Datei zu exportieren. Das HSM erlaubt keinen direkten Export von Schlüsseln im Klartext. Der Befehl umschließt den privaten Schlüssel mit einem von Ihnen angegebenen AES-Wrapping-Schlüssel, entschlüsselt die umschlossenen Bytes und kopiert den privaten Klartext-Schlüssel in eine Datei.

Der `exportPrivateKey`-Befehl entfernt den Schlüssel nicht aus dem HSM, ändert nicht seine [Schlüsselattribute](#) und hindert Sie nicht daran, den Schlüssel für weitere kryptografische Operationen zu verwenden. Sie können den gleichen Schlüssel mehrmals exportieren.

Sie können nur private Schlüssel exportieren, die den `OBJ_ATTR_EXTRACTABLE`-Attributwert 1 haben. Sie müssen einen AES-Wrapping-Schlüssel angeben, der `OBJ_ATTR_WRAP`- und `OBJ_ATTR_DECRYPT`-Attributwerte 1 hat. Mit dem Befehl [getAttribute](#) können Sie nach den Attributen eines Schlüssels suchen.

Bevor Sie einen `key_mgmt_util`-Befehl ausführen, müssen Sie [key_mgmt_util starten](#) und sich am HSM als Crypto-Benutzer (CU) [anmelden](#).

Syntax

```
exportPrivateKey -h

exportPrivateKey -k <private-key-handle>
                  -w <wrapping-key-handle>
                  -out <key-file>
                  [-m <wrapping-mechanism>]
                  [-wk <wrapping-key-file>]
```

Beispiele

In diesem Beispiel wird gezeigt, wie Sie mit `exportPrivateKey` einen privaten Schlüssel aus einem HSM exportieren.

Example: Exportieren eines privaten Schlüssels

Dieser Befehl exportiert einen privaten Schlüssel mit dem Handle 15 anhand eines Verpackungsschlüssels mit dem Handle 16 in eine PEM-Datei mit dem Namen `exportKey.pem`. Wird der Befehl erfolgreich ausgeführt, gibt `exportPrivateKey` eine Erfolgsmeldung zurück.

```
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
```

Parameters

Dieser Befehl erfordert die folgenden Parameter.

-h

Zeigt die Befehlszeilenhilfe für den Befehl an.

Erforderlich: Ja

-k

Gibt das Schlüssel-Handle des zu exportierenden privaten Schlüssels an.

Erforderlich: Ja

-w

Gibt das Schlüssel-Handle des Verpackungsschlüssels an. Dieser Parameter muss angegeben werden. Nutzen Sie den Befehl [findKey](#), um Schlüssel-Handles zu suchen.

Ermitteln Sie mithilfe von [getAttribute](#) den Wert des OBJ_ATTR_WRAP-Attributs (262), um zu bestimmen, ob ein Schlüssel als Verpackungsschlüssel verwendet werden kann. Um einen Wrapping-Schlüssel zu erstellen verwenden Sie [genSymKey](#), um einen AES-Schlüssel (Typ 31) zu generieren.

Wenn Sie den Parameter `-wk` zum Angeben eines externen Entpackungsschlüssels verwenden, wird der `-w`-Verpackungsschlüssel während des Exports zum Verpacken, nicht aber zum Entpacken des Schlüssels verwendet.

Erforderlich: Ja

-out

Gibt den Namen der Datei an, in die der exportierte private Schlüssel geschrieben werden soll.

Erforderlich: Ja

-m

Gibt den Verpackungsmechanismus an, mit dem der zu exportierende private Schlüssel verpackt werden soll. Der einzige gültige Wert ist 4. Dieser repräsentiert den NIST_AES_WRAP mechanism.-Mechanismus.

Standard: 4 (NIST_AES_WRAP)

Erforderlich: Nein

-wk

Gibt den Schlüssel an, mit dem der zu exportierende Schlüssel entpackt werden soll. Geben Sie den Pfad und den Namen einer Datei an, die einen Klartext-AES-Schlüssel enthält.

Wenn Sie diesen Parameter einschließen, verwendet `exportPrivateKey` den Schlüssel in der `-w`-Datei, um den zu exportierenden Schlüssel zu verpacken. Zum Entpacken wird der über den `-wk`-Parameter angegebene Schlüssel verwendet.

Standard: Verwenden Sie den im `-w`-Parameter angegebenen Verpackungsschlüssel für das Verpacken und Entpacken von Schlüsseln.

Erforderlich: Nein

Verwandte Themen

- [importPrivateKey](#)
- [wrapKey](#)
- [unWrapKey](#)
- [genSymKey](#)

Exportieren eines öffentlichen AWS CloudHSM Schlüssels mit KMU

Verwenden Sie den `exportPubKey` Befehl in AWS CloudHSM `key_mgmt_util`, um einen öffentlichen Schlüssel in einem HSM in eine Datei zu exportieren. Sie können ihn verwenden, um öffentliche Schlüssel zu exportieren, die Sie in einem HSM erstellt haben. Mit diesem Befehl ist auch das Exportieren öffentlicher Schlüssel möglich, die in ein HSM importiert wurden. Dies betrifft z. B. jene Schlüssel, deren Import über den Befehl [importPubKey](#) erfolgt ist.

Die Operation `exportPubKey` kopiert das Schlüsselmateriale in eine von Ihnen angegebene Datei. Der Befehl entfernt jedoch den Schlüssel nicht aus dem HSM oder ändert dessen [Schlüsselattribute](#). Auch hindert Sie der Befehl nicht daran, den Schlüssel in weiteren kryptografischen Operationen zu verwenden. Sie können den gleichen Schlüssel mehrmals exportieren.

Sie können nur öffentliche Schlüssel exportieren, deren `OBJ_ATTR_EXTRACTABLE`-Wert auf 1 gesetzt ist. Mit dem Befehl [getAttribute](#) können Sie nach den Attributen eines Schlüssels suchen.

Vor dem Ausführen eines `key_mgmt_util`-Befehls müssen Sie [key_mgmt_util starten](#) und sich beim HSM als Crypto-Benutzer (CU) [anmelden](#).

Syntax

```
exportPubKey -h

exportPubKey -k <public-key-handle>
               -out <key-file>
```

Beispiele

In diesem Beispiel wird gezeigt, wie Sie mit `exportPubKey` einen öffentlichen Schlüssel aus einem HSM exportieren.

Example: Exportieren eines öffentlichen Schlüssels

Mit diesem Befehl wird ein öffentlicher Schlüssel mit dem Handle `10` in eine Datei namens `public.pem` exportiert. Wird der Befehl erfolgreich ausgeführt, gibt `exportPubKey` eine Erfolgsmeldung zurück.

```
Command: exportPubKey -k 10 -out public.pem
```

```
PEM formatted public key is written to public.pem
```

```
Cfm3ExportPubKey returned: 0x00 : HSM Return: SUCCESS
```

Parameters

Dieser Befehl erfordert die folgenden Parameter.

-h

Zeigt die Befehlszeilenhilfe für den Befehl an.

Erforderlich: Ja

-k

Gibt das Schlüssel-Handle des zu exportierenden öffentlichen Schlüssels an.

Erforderlich: Ja

-out

Gibt den Namen der Datei an, in die der exportierte öffentliche Schlüssel geschrieben werden soll.

Erforderlich: Ja

Verwandte Themen

- [importPubKey](#)
- [Generieren von Schlüsseln](#)

Exportieren Sie eine Klartextkopie eines AWS CloudHSM Schlüssels mit KMU

Verwenden Sie den `exSymKey` Befehl im Tool AWS CloudHSM `key_mgmt_util`, um eine Klartextkopie eines symmetrischen Schlüssels aus dem Hardware-Sicherheitsmodul (HSM) zu exportieren und in einer Datei auf der Festplatte zu speichern. Um eine verschlüsselte (verpackte) Kopie eines Schlüssels zu exportieren, verwenden Sie [wrapKey](#). Um einen Klartext-Schlüssel zu importieren, wie z. B. Schlüssel, der exportiert wird, verwenden Sie `exSymKey` [imSymKey](#)

Während des Exportvorgangs verwendet `exSymKey` einen AES-Schlüssel, den Sie angeben (der wrapping key (Verpackungsschlüssel)), um den zu exportierenden Schlüssel zu packen

(verschlüsseln) und anschließend zu entpacken (entschlüsseln). Das Ergebnis des Exportvorgangs ist jedoch ein Klartextschlüssel (entpackter Schlüssel) auf einem Datenträger.

Nur der Eigentümer eines Schlüssels, d. h., der CU-Benutzer, der den Schlüssel erstellt hat, kann ihn exportieren. Benutzer, für die der Schlüssel freigegeben ist, können ihn in kryptografischen Vorgängen verwenden, können ihn aber nicht exportieren.

Der Vorgang `exSymKey` kopiert die Schlüsselinformationen in eine Datei, die Sie angeben, entfernt den Schlüssel jedoch nicht aus dem HSM, ändert seine [key attributes \(Schlüsselattribute\)](#) nicht oder verhindert, dass Sie den Schlüssel in kryptografischen Vorgänge verwenden. Sie können den gleichen Schlüssel mehrmals exportieren.

`exSymKey` exportiert nur symmetrische Schlüssel. Zum Exportieren von öffentlichen Schlüsseln verwenden Sie [exportPubKey](#). Zum Exportieren von privaten Schlüsseln verwenden Sie [exportPrivateKey](#).

Bevor Sie einen `key_mgmt_util`-Befehl ausführen, müssen Sie [key_mgmt_util starten](#) und sich am HSM als Crypto-Benutzer (CU) [anmelden](#).

Syntax

```
exSymKey -h

exSymKey -k <key-to-export>
          -w <wrapping-key>
          -out <key-file>
          [-m 4]
          [-wk <unwrapping-key-file> ]
```

Beispiele

Diese Beispiele zeigen, wie Sie symmetrische SchlüsselexSymKey, die Sie besitzen, aus Ihrem exportieren können. HSMs

Example: Exportieren eines symmetrischen 3DES-Schlüssels

Mit diesem Befehl wird ein symmetrischer Triple DES (3DES)-Schlüssel exportiert (Schlüssel-Handle 7). Er verwendet einen vorhandenen AES-Schlüssel (Schlüssel-Handle 6) im HSM als Verpackungsschlüssel. Anschließend wird der Klartext des 3DES-Schlüssels in die Datei `3DES.key` geschrieben.

Die Ausgabe zeigt, dass der Schlüssel 7 (der 3DES-Schlüssel) erfolgreich verpackt und entpackt und anschließend in die angegebene Datei 3DES.key geschrieben wurde.

 Warning

Obwohl die Ausgabe besagt, dass ein „verpackter symmetrischer Schlüssel“ in die Ausgabedatei geschrieben wurden, enthält die Ausgabedatei einen (entpackten) Klartextschlüssel.

```
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: Exportieren mit einem nur für die aktuelle Sitzung gültigen Verpackungsschlüssel

Das folgende Beispiel zeigt, wie Sie einen Schlüssel verwenden, der nur in dieser Sitzung als Verpackungsschlüssel vorhanden ist. Da der zu exportierende Schlüssel verpackt ist, anschließend sofort entpackt und als Klartext bereitgestellt wird, ist es nicht nötig, den Verpackungsschlüssel aufzubewahren.

Anhand dieser Befehle wird ein AES-Schlüssel mit der Schlüssel-Handle 8 aus dem HSM exportiert. Dabei wird ein AES-Sitzungsschlüssel verwendet, der speziell zu diesem Zweck erstellt wurde.

Der erste Befehl dient [genSymKey](#) zur Erstellung eines 256-Bit-AES-Schlüssels. Dabei wird der `-sess`-Parameter zum Erstellen eines Schlüssels verwendet, der nur in der aktuellen Sitzung vorhanden ist.

Die Ausgabe zeigt, dass das HSM den Schlüssel 262168 erstellt.

```
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

Als Nächstes wird im Beispiel sichergestellt, dass es sich bei dem Schlüssel 8, dem zu exportierenden Schlüssel, um einen symmetrischen Schlüssel handelt, der extrahierbar ist. Außerdem wird überprüft, ob der Verpackungsschlüssel, der Schlüssel 262168, ein AES-Schlüssel ist, der nur in dieser Sitzung vorhanden ist. Sie können den Befehl [findKey](#) verwenden, aber in diesem Beispiel werden die Attribute beider Schlüssel in Dateien exportiert und anschließend grep verwendet, um die relevanten Attributwerte in der Datei zu finden.

Diese Befehle verwenden `getAttribute` mit einem `-a` Wert von 512 (alle), um alle Attribute für die Schlüssel 8 und 262168 aufzurufen. Weitere Informationen zu den Schlüsselattributen finden Sie unter [the section called "Referenz auf das Schlüsselattribut"](#).

```
getAttribute -o 8 -a 512 -out attributes/attr_8
getAttribute -o 262168 -a 512 -out attributes/attr_262168
```

Diese Befehle verwenden `grep`, um die Attribute des zu exportierenden Schlüssels (Schlüssel 8) und des nur für die aktuelle Sitzung gültigen Verpackungsschlüssels (Schlüssel 262168) zu überprüfen.

```
// 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
```

Schließlich verwenden wir den Befehl `exSymKey`, um den Schlüssel 8 mithilfe des Sitzungsschlüssels (Schlüssel 262168) als Verpackungsschlüssel zu exportieren.

Wenn die Sitzung beendet wird, wird der Schlüssel 262168 nicht länger vorhanden sein.

```
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: Verwenden eines externen Entpackungsschlüssels

Das folgende Beispiel zeigt, wie Sie mit einem externen Entpackungsschlüssel einen Schlüssel aus dem HSM exportieren.

Wenn Sie einen Schlüssel aus dem HSM exportieren, geben Sie einen AES-Schlüssel im HSM als Verpackungsschlüssel an. Standardmäßig wird dieser Schlüssel dazu verwendet, den zu exportierenden Schlüssel zu verpacken und zu entpacken. Sie können jedoch mithilfe des `-wk`-Parameters `exSymKey` anweisen, zum Entpacken einen externen Schlüssel in einer Datei auf einem Datenträger zu verwenden. Wenn Sie sich dafür entscheiden, verpackt der vom `-w`-Parameter angegebene Schlüssel den Ziel-Schlüssel, und der Schlüssel, der sich in der vom `-wk`-Parameter angegebenen Datei befindet, entpackt den Schlüssel.

Da der Verpackungsschlüssel ein AES-Schlüssel sein muss, und dieser symmetrisch ist, müssen der Verpackungsschlüssel im HSM und der Entpackungsschlüssel auf dem Datenträger über die gleichen Schlüsselinformationen verfügen. Hierzu müssen Sie vor dem Exportvorgang den Verpackungsschlüssel in das HSM importieren oder den Verpackungsschlüssel aus dem HSM exportieren.

In diesem Beispiel wird ein Schlüssel außerhalb des HSM erstellt und in das HSM importiert. Dabei wird eine interne Kopie des Schlüssels verwendet, um den zu exportierenden symmetrischen

Schlüssel zu verpacken, und anschließend die Kopie des Schlüssels in der Datei dazu benutzt, um ihn zu entpacken.

Der erste Befehl verwendet OpenSSL, um einen 256-Bit-AES-Schlüssel zu generieren. Der Schlüssel wird in der Datei `aes256-forImport.key` gespeichert. Der OpenSSL-Befehl gibt keine Ausgabe zurück, jedoch können Sie mehrere Befehle verwenden, um zu überprüfen, ob der Vorgang erfolgreich war. In diesem Beispiel wird das Tool `wc` (Wortzählung) verwendet, das bestätigt, dass die Datei 32 Byte an Daten enthält.

```
$ openssl rand -out keys/aes256-forImport.key 32
```

```
$ wc keys/aes256-forImport.key
0  2 32 keys/aes256-forImport.key
```

Dieser Befehl verwendet den Befehl [imSymKey](#), um den AES-Schlüssel aus der Datei `aes256-forImport.key` in das HSM zu importieren. Wenn der Befehl abgeschlossen ist, ist der Schlüssel im HSM mit dem Schlüssel-Handle 262167 und in der Datei `aes256-forImport.key` vorhanden.

```
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
```

Dieser Befehl verwendet den Schlüssel in einem Exportvorgang. Der Befehl verwendet `exSymKey`, um den Schlüssel 21, ein 192-Bit-AES-Schlüssel, zu exportieren. Um den Schlüssel zu verpacken, wird der Schlüssel 262167 verwendet. Dabei handelt es sich um die Kopie des Schlüssels, die in das HSM importiert wurde. Um den Schlüssel zu entpacken, werden die gleichen Schlüsselinformationen in der Datei `aes256-forImport.key` verwendet. Nach der Ausführung des Befehls wird der Schlüssel 21 in die Datei `aes192_h21.key` exportiert.

```
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"
```

Parameters

-h

Zeigt Hilfe für den Befehl an.

Erforderlich: Ja

-k

Gibt das Schlüssel-Handle des zu exportierenden Schlüssels an. Dieser Parameter muss angegeben werden. Geben Sie das Schlüssel-Handle eines symmetrischen Schlüssels ein, dessen Eigentümer Sie sind. Dieser Parameter muss angegeben werden. Verwenden Sie den [findKey](#)-Befehl, um Schlüssel-Handles zu finden.

Verwenden Sie den Befehl [getAttribute](#), um sicherzustellen, dass ein Schlüssel exportiert werden kann, und um den Wert des Attributs OBJ_ATTR_EXTRACTABLE abzurufen, der von der Konstanten 354 dargestellt wird. Zudem können Sie nur Schlüssel exportieren, deren Eigentümer Sie sind. Verwenden Sie den [getKeyInfo](#)-Befehl, um den Besitzer eines Schlüssels zu ermitteln.

Erforderlich: Ja

-w

Gibt das Schlüssel-Handle des Verpackungsschlüssels an. Dieser Parameter muss angegeben werden. Verwenden Sie den [findKey](#)-Befehl, um Schlüssel-Handles zu finden.

Ein Verpackungsschlüssel ist ein Schlüssel im HSM, der zum Verschlüsseln (Verpacken) und zum anschließenden Entschlüsseln (Entpacken) des zu exportierenden Schlüssels verwendet wird. Nur AES-Schlüssel können als Verpackungsschlüssel verwendet werden.

Sie können jeden AES-Schlüssel (in jeder beliebigen Größe) als Verpackungsschlüssel verwenden. Da der Verpackungsschlüssel den Zielschlüssel verpackt und danach sofort entpackt, können Sie einen nur für die aktuelle Sitzung gültigen AES-Schlüssel als Verpackungsschlüssel verwenden. Verwenden Sie [getAttribute](#), um zu überprüfen, ob ein Schlüssel als Verpackungsschlüssel verwendet werden kann, und um den Wert des

OBJ_ATTR_WRAP-Attributs abzurufen, der von der Konstanten 262 dargestellt wird. Um einen Wrapping-Schlüssel zu erstellen, verwenden Sie [genSymKey](#), um einen AES-Schlüssel (Typ 31) zu generieren.

Wenn Sie den Parameter `-wk` zum Angeben eines externen Entschlüsselungsschlüssels verwenden, wird der `-w`-Schlüssel während des Exportvorgangs zum Verpacken, aber nicht zum Entpacken des Schlüssels verwendet.

 Note

Schlüssel 4 stellt einen nicht unterstützten internen Schlüssel dar. Wir empfehlen, dass Sie einen AES-Schlüssel als Umhüllungsschlüssel verwenden, den Sie erstellen und verwalten.

Erforderlich: Ja

`-out`

Gibt den Pfad und Namen der Ausgabedatei an. Wenn der Befehl erfolgreich ist, enthält diese Datei den exportierten Schlüssel als Klartext. Wenn die Datei bereits vorhanden ist, überschreibt der Befehl sie ohne Warnung.

Erforderlich: Ja

`-m`

Gibt den Verpackungsmechanismus an. Der einzige gültige Wert ist 4, der den NIST_AES_WRAP-Mechanismus darstellt.

Erforderlich: Nein

Standard: 4

`-wk`

Verwenden Sie den AES-Schlüssel in der angegebenen Datei, um den Schlüssel, der exportiert wird, zu entpacken. Geben Sie den Pfad und den Namen einer Datei an, die einen Klartext-AES-Schlüssel enthält.

Wenn Sie diesen Parameter angeben, verwendet `exSymKey` den Schlüssel im HSM, der durch den `-w`-Parameter angegeben wird, um den zu exportierenden Schlüssel zu verpacken, und

verwendet den Schlüssel in der -wk-Datei, um ihn zu entpacken. Die Parameterwerte -w und -wk müssen durch denselben Klartextschlüssel aufgelöst werden.

Erforderlich: Nein

Standard: Verwenden Sie den Verpackungsschlüssel auf dem HSM zum Entpacken.

Verwandte Themen

- [genSymKey](#)
- [imSymKey](#)
- [wrapKey](#)

Extrahieren Sie einen AWS CloudHSM Schlüssel mit KMU

Verwenden Sie den `extractMaskedObject` Befehl in AWS CloudHSM `key_mgmt_util`, um einen Schlüssel aus einem Hardware-Sicherheitsmodul (HSM) zu extrahieren und ihn als maskiertes Objekt in einer Datei zu speichern. Maskierte Objekte sind geklonte Objekte, die nur verwendet werden können, nachdem sie anhand des Befehls [insertMaskedObject](#) wieder in den ursprünglichen Cluster eingefügt wurden. Sie können ein maskiertes Objekt nur in den Cluster einfügen, aus dem es erstellt wurde (oder in eine geklonte Version dieses Clusters). Dies umfasst alle geklonten Clusterversionen, die durch eine [regionsübergreifende Sicherungskopie](#) und die anschließende [Verwendung dieser Sicherung zum Erstellen eines neuen Clusters](#) generiert wurden.

Maskierte Objekte sind eine effiziente Möglichkeit zum Auslagern und Synchronisieren von Schlüsseln, auch von nicht extrahierbaren Schlüsseln (d. h. Schlüssel mit einem `OBJ_ATTR_EXTRACTABLE`-Wert von 0). [Auf diese Weise können Schlüssel sicher zwischen verwandten Clustern in verschiedenen Regionen synchronisiert werden, ohne dass die Konfigurationsdatei aktualisiert werden muss. AWS CloudHSM](#)

Important

Nach dem Einfügen werden maskierte Objekte entschlüsselt und mit einem Schlüssel-Handle versehen, das sich vom Schlüssel-Handle des ursprünglichen Schlüssels unterscheidet. Ein maskiertes Objekt enthält alle dem ursprünglichen Schlüssel zugeordneten Metadaten, einschließlich Attributen, Eigentümerschaft, der gemeinsamen Nutzung von Informationen

und Quorum-Einstellungen. Wenn Sie in einer Anwendung Schlüssel zwischen Clustern synchronisieren müssen, verwenden Sie stattdessen [syncKey](#) im `cloudhsm_mgmt_util`.

Bevor Sie einen `key_mgmt_util`-Befehl ausführen, müssen Sie [key_mgmt_util](#) starten und sich beim [HSM](#) anmelden. Der Befehl `extractMaskedObject` kann entweder vom CU, der den Schlüssel besitzt, oder von einem beliebigen CO genutzt werden.

Syntax

```
extractMaskedObject -h

extractMaskedObject -o <object-handle>
                    -out <object-file>
```

Beispiele

In diesem Beispiel wird gezeigt, wie Sie `extractMaskedObject` verwenden, um einen Schlüssel als maskiertes Objekt aus einem HSM zu extrahieren.

Example: Extrahieren eines maskierten Objekts

Dieser Befehl extrahiert in einem HSM aus einem Schlüssel mit dem Handle 524295 ein maskiertes Objekt und speichert es als Datei mit dem Namen `maskedObj`. Wird der Befehl erfolgreich ausgeführt, gibt `extractMaskedObject` eine Erfolgsmeldung zurück.

```
Command: extractMaskedObject -o 524295 -out maskedObj

Object was masked and written to file "maskedObj"

Cfm3ExtractMaskedObject returned: 0x00 : HSM Return: SUCCESS
```

Parameters

Dieser Befehl erfordert die folgenden Parameter.

-h

Zeigt die Befehlszeilenhilfe für den Befehl an.

Erforderlich: Ja

-o

Gibt das Handle des Schlüssels an, der als maskiertes Objekt extrahiert werden soll.

Erforderlich: Ja

-out

Gibt den Namen der Datei an, in die das maskierte Objekt gespeichert werden soll.

Erforderlich: Ja

Verwandte Themen

- [insertMaskedObject](#)
- [syncKey](#)
- [Regionsübergreifendes Kopieren einer Sicherung](#)
- [Einen AWS CloudHSM Cluster aus einem vorherigen Backup erstellen](#)

Suchen Sie mithilfe von KMU nach AWS CloudHSM Schlüsseln anhand von Attributen

Verwenden Sie den findKey Befehl in AWS CloudHSM key_mgmt_util, um anhand der Werte der Schlüsselattribute nach Schlüsseln zu suchen. Wenn ein Schlüssel allen Kriterien entspricht, die Sie festlegen, gibt findKey das Schlüssel-Handle zurück. Ohne Parameter gibt findKey die Schlüssel-Handles aller Schlüssel zurück, die Sie im HSM verwenden können. Verwenden Sie für die Suche der Attributwerte eines bestimmten Schlüssels den Befehl [getAttribute](#).

Wie alle key_mgmt_util-Befehle ist findKey benutzerspezifisch. Der Befehl gibt nur die Schlüssel zurück, die der aktuelle Benutzer in kryptografischen Operationen verwenden kann. Dies umfasst Schlüssel, die der aktuelle Benutzer besitzt und Schlüssel, die für ihn freigegeben wurden.

Bevor Sie einen key_mgmt_util-Befehl ausführen, müssen Sie [key_mgmt_util starten](#) und sich am HSM als Crypto-Benutzer (CU) [anmelden](#).

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>]
```

Beispiele

Diese Beispiele zeigen, wie Sie Schlüssel in Ihrem finden und findKey identifizieren können. HSMs

Example: Suchen nach allen Schlüsseln

Mit diesem Befehl werden alle Schlüssel für den aktuellen Benutzer im HSM gesucht. Die Ausgabe umfasst Schlüssel, die der Benutzer besitzt und teilt, sowie alle öffentlichen Schlüssel in der HSMs.

Verwenden Sie zum Abrufen der Attribute eines Schlüssels mit einem bestimmten Schlüssel-Handle den Befehl [getAttribute](#). Um festzustellen, ob der aktuelle Benutzer einen bestimmten Schlüssel besitzt oder mit anderen teilt, verwenden Sie [getKeyInfo](#) oder [findAllKeys](#) in cloudhsm_mgmt_util.

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: Suchen von Schlüsseln nach Typ, Benutzer und Sitzung

Mit diesem Befehl werden persistente AES-Schlüssel gesucht, die vom aktuellen Benutzer und von Benutzer 3 verwendet werden können. (Benutzer 3 kann möglicherweise andere Schlüssel verwenden, die der aktuelle Benutzer nicht sehen kann.)

Command: **findKey -t 31 -sess 0 -u 3**

Example: Suchen von Schlüsseln nach Klasse und Beschriftung

Mit diesem Befehl werden alle öffentlichen Schlüssel für den aktuellen Benutzer mit der Beschriftung `2018-sept` gesucht.

```
Command: findKey -c 2 -l 2018-sept
```

Example: Suchen von RSA-Schlüsseln nach Modulus

Mit diesem Befehl können Sie RSA-Schlüssel (Typ 0) für den aktuellen Benutzer suchen, die mit dem Modulus in der Datei `m4.txt` erstellt wurden.

```
Command: findKey -t 0 -m m4.txt
```

Parameters

-h

Zeigt Hilfe für den Befehl an.

Erforderlich: Ja

-t

Sucht Schlüssel des angegebenen Typs. Geben Sie die Konstante ein, die die Schlüsselklasse darstellt. Geben Sie beispielsweise `-t 21` ein, um 3DES-Schlüssel zu suchen.

Zulässige Werte:

- 0: [RSA](#)
- 1: [DSA](#)
- 3: [EC](#)
- 16: [GENERIC_SECRET](#)
- 18: [RC4](#)
- 21: [Triple DES \(3DES\)](#)
- 31: [AES](#)

Erforderlich: Nein

-c

Sucht Schlüssel in der angegebenen Klasse. Geben Sie die Konstante ein, die die Schlüsselklasse darstellt. Geben Sie beispielsweise -c 2 ein, um öffentliche Schlüssel zu suchen.

Gültige Werte für jeden Schlüsseltyp:

- 2: Public. Diese Klasse enthält die öffentlichen Schlüssel der öffentlich-privaten Schlüsselpaare.
- 3: Private. Diese Klasse enthält die privaten Schlüssel der öffentlich-privaten Schlüsselpaare.
- 4: Secret. Diese Klasse enthält alle symmetrischen Schlüssel.

Erforderlich: Nein

-l

Sucht Schlüssel mit der angegebenen Beschriftung. Geben Sie die genaue Beschriftung ein. Sie können keine Platzhalter oder regulären Ausdrücke im Wert -l verwenden.

Erforderlich: Nein

-id

Sucht den Schlüssel mit der angegebenen ID. Geben Sie die genaue ID-Zeichenfolge ein. Sie können keine Platzhalter oder regulären Ausdrücke im Wert -id verwenden.

Erforderlich: Nein

-sess

Sucht Schlüssel nach Sitzungsstatus. Geben Sie für die Suche nach Schlüsseln, die nur in der aktuellen Sitzung gültig sind, 1 ein. Geben Sie für die Suche nach persistenten Schlüsseln 0 ein.

Erforderlich: Nein

-u

Sucht Schlüssel, die von den angegebenen Benutzern und dem aktuellen Benutzer gemeinsam verwendet werden. Geben Sie eine durch Kommas getrennte Liste von HSM-Benutzern ein IDs, z. B. oder. -u 3 -u 4, 7 Verwenden Sie IDs [ListUsers](#), um die Benutzer auf einem HSM zu finden.

Wenn Sie eine Benutzer-ID angeben, gibt findKey den Schlüssel für diesen Benutzer zurück. Wenn Sie mehrere Benutzer angeben IDs, werden die Schlüssel findKey zurückgegeben, die alle angegebenen Benutzer verwenden können.

Da `findKey` nur Schlüssel zurückgibt, die der aktuelle Benutzer verwenden kann, sind die -u-Ergebnisse immer identisch oder eine Teilmenge der Schlüssel des aktuellen Benutzers. Um alle Schlüssel abzurufen, die einem Benutzer gehören oder mit diesem geteilt werden, können Crypto Officers (COs) [findAllKeys](#) in `cloudhsm_mgmt_util` verwenden.

Erforderlich: Nein

-m

Sucht Schlüssel, die mit dem RSA-Modulus in der angegebenen Datei erstellt wurden. Geben Sie den Pfad zur Datei ein, in der der Modulus gespeichert ist.

-m gibt die Binärdatei an, die den RSA-Modul enthält, mit dem abgeglichen werden soll (optional).

Erforderlich: Nein

-kcv

Sucht Schlüssel mit dem angegebenen Schlüsselprüfwert.

Der Schlüsselprüfwert (KCV) ist ein 3-Byte-Hash oder eine Prüfsumme eines Schlüssels, der generiert wird, wenn das HSM einen Schlüssel importiert oder generiert. Sie können einen KCV auch außerhalb des HSM berechnen, z. B. nachdem Sie einen Schlüssel exportiert haben. Anschließend können Sie die KCVs vergleichen, um die Identität und Integrität des Schlüssels zu bestätigen. Um den KCV eines Schlüssels abzurufen, verwenden Sie [getAttribute](#).

AWS CloudHSM verwendet die folgende Standardmethode, um einen Schlüsselprüfwert zu generieren:

- Symmetrische Schlüssel: Die ersten 3 Byte des Ergebnisses der Verschlüsselung eines Nullblocks mit dem Schlüssel.
- Asymmetrische Schlüsselpaare: Die ersten 3 Byte des SHA-1-Hashs des öffentlichen Schlüssels.
- HMAC-Schlüssel: KCV für HMAC-Schlüssel wird derzeit nicht unterstützt.

Erforderlich: Nein

Ausgabe

Die Ausgabe des Befehls `findKey` enthält die Gesamtanzahl der übereinstimmenden Schlüssel und ihrer Schlüssel-Handles.

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
```

Verwandte Themen

- [findSingleKey](#)
- [getKeyInfo](#)
- [getAttribute](#)
- [findAllKeys](#) in cloudhsm_mgmt_util
- [Schlüsselattributreferenz](#)

Verifizieren Sie einen AWS CloudHSM Schlüssel mit KMU

Verwenden Sie den findSingleKey Befehl im Tool AWS CloudHSM key_mgmt_util, um zu überprüfen, ob ein Schlüssel auf allen Hardware-Sicherheitsmodulen (HSM) im Cluster vorhanden ist. AWS CloudHSM

Bevor Sie einen key_mgmt_util-Befehl ausführen, müssen Sie [key_mgmt_util starten](#) und sich am HSM als Crypto-Benutzer (CU) [anmelden](#).

Syntax

```
findSingleKey -h
```

```
findSingleKey -k <key-handle>
```

Beispiel

Example

Mit diesem Befehl wird überprüft, ob der Schlüssel 252136 auf allen drei im Cluster vorhandenen HSMs

```
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
```

Parameters

-h

Zeigt Hilfe für den Befehl an.

Erforderlich: Ja

-k

Gibt das Schlüssel-Handle eines Schlüssels im HSM an. Dieser Parameter muss angegeben werden.

Verwenden Sie den [findKey](#)-Befehl, um Schlüssel-Handles zu finden.

Erforderlich: Ja

Verwandte Themen

- [findKey](#)
- [getKeyInfo](#)
- [getAttribute](#)

Generieren Sie ein AWS CloudHSM DSA-Schlüsselpaar mit KMU

Verwenden Sie den `genDSAKeyPair` Befehl im Tool AWS CloudHSM `key_mgmt_util`, um ein DSA-Schlüsselpaar ([Digital Signing Algorithm](#)) in Ihren Hardware-Sicherheitsmodulen (HSM) zu generieren. Sie müssen die Modullänge angeben, der Befehl generiert den Modulwert. Sie können auch eine ID zuweisen, den Schlüssel für andere HSM-Benutzer freigeben sowie nicht extrahierbare Schlüssel und Schlüssel, die bei Sitzungsende ablaufen, erstellen. Wenn der Befehl erfolgreich ausgeführt wurde, werden die Schlüssel-Handles zurückgegeben, die das HSM den öffentlichen und

privaten ECC-Schlüsseln zuweist. Sie können die Schlüssel-Handles nutzen, damit die Schlüssel für andere Befehle identifizierbar sind.

Bevor Sie einen `key_mgmt_util`-Befehl ausführen, müssen Sie [key_mgmt_util starten](#) und sich am HSM als Crypto-Benutzer (CU) [anmelden](#).

Tip

Um die Attribute eines von Ihnen erstellten Schlüssels wie Typ, Länge, Bezeichnung und ID zu finden, verwenden Sie [getAttribute](#). Um die Schlüssel für einen bestimmten Benutzer zu finden, verwenden Sie [getKeyInfo](#). Verwenden Sie [findKey](#), um Schlüssel anhand ihrer Attributwerte zu finden.

Syntax

```
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]
```

Beispiele

Diese Beispiele verdeutlichen, wie mit `genDSAKeyPair` ein DSA-Schlüsselpaar erstellt wird.

Example: Erstellen eines DSA-Schlüsselpaares

Mit diesem Befehl wird ein DSA-Schlüsselpaar mit einem DSA-Label erstellt. Die Ausgabe zeigt, dass das Schlüssel-Handle des öffentlichen Schlüssels 19 ist, und dass das Schlüssel-Handle des privaten Schlüssels 21 lautet.

```
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: Erstellen eines DSA-Schlüsselpaares, das nur für eine Sitzung gültig ist

Mit diesem Befehl wird ein DSA-Schlüsselpaar erstellt, das nur für die aktuelle Sitzung gültig ist. Der Befehl weist zusätzlich zu der erforderlichen (nicht eindeutigen) Bezeichnung eine eindeutige ID von DSA_temp_pair zu. Sie können ein solches Schlüsselpaar erstellen, um ein nur für diese Sitzung gültiges Token zu signieren und zu überprüfen. Die Ausgabe zeigt, dass das Schlüssel-Handle des öffentlichen Schlüssels 12 ist, und dass das Schlüssel-Handle des privaten Schlüssels 14 lautet.

```
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
```

Um zu überprüfen, ob das Schlüsselpaar nur in der Sitzung vorhanden ist, verwenden Sie den -sess-Parameter von [findKey](#) mit einem Wert von 1 (true).

```
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: Erstellen eines freigegebenen, unextrahierbaren DSA-Schlüsselpaares

Mit diesem Befehl wird ein DSA-Schlüsselpaar erstellt. Der private Schlüssel wird mit drei anderen Benutzern geteilt und kann nicht aus dem HSM exportiert werden. Öffentliche Schlüssel können von jedem Benutzer verwendet und immer extrahiert werden.

```
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: Erstellen eines Quorum-kontrollierten Schlüsselpaares

Mit diesem Befehl wird ein DSA-Schlüsselpaar mit der Bezeichnung DSA-mV2 erstellt. Der Befehl verwendet den `-u`-Parameter, um den privaten Schlüssel für die Benutzer 4 und 6 freizugeben. Er verwendet den `-m_value`-Parameter, damit für alle kryptografischen Vorgänge, die den privaten Schlüssel verwenden, ein Quorum aus mindestens zwei Genehmigungen erforderlich ist. Der Befehl verwendet auch den `-attest`-Parameter, um die Integrität der Firmware zu überprüfen, auf der das Schlüsselpaar generiert wird.

Die Ausgabe zeigt, dass der Befehl einen öffentlichen Schlüssel mit dem Schlüssel-Handle 12 und einen privaten Schlüssel mit dem Schlüssel-Handle 17 generiert, und dass die Bescheinigungsprüfung auf der Cluster-Firmware erfolgreich bestanden wurde.

```
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
```

Dieser Befehl verwendet [getKeyInfo](#) den privaten Schlüssel (Schlüsselname17). Die Ausgabe bestätigt, dass es sich bei dem Eigentümer um den aktuellen Benutzer (Benutzer 3) handelt, und dass der Schlüssel mit den Benutzern 4 und 6 (und keinen anderen) geteilt wird. Die Ausgabe zeigt zudem, dass die Quorum-Authentifizierung aktiviert ist und die Quorumgröße 2 beträgt.

```
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
```

Parameters

-h

Zeigt Hilfe für den Befehl an.

Erforderlich: Ja

-m

Gibt die Länge des Moduls in Bits an. Der einzige gültige Wert ist 2048.

Erforderlich: Ja

-l

Gibt eine benutzerdefinierte Bezeichnung für das Schlüsselpaar an. Geben Sie eine Zeichenfolge ein. Dieselbe Bezeichnung gilt für beide Schlüssel im Paar. Die maximal zulässige Größe für `label` beträgt 127 Zeichen.

Sie können eine beliebige Phrase verwenden, die Ihnen bei der Identifizierung des Schlüssels hilft. Da die Bezeichnung nicht eindeutig sein muss, können Sie sie verwenden, um Schlüssel zu gruppieren und zu kategorisieren.

Erforderlich: Ja

-id

Gibt einen benutzerdefinierten Bezeichner für das Schlüsselpaar an. Geben Sie eine Zeichenfolge ein, die im Cluster eindeutig ist. Der Standardwert ist eine leere Zeichenfolge. Die von Ihnen angegebene ID gilt für beide Schlüssel im Paar.

Standard : Kein ID-Wert.

Erforderlich: Nein

-min_srv

Gibt die Mindestanzahl HSMs an, mit der der Schlüssel synchronisiert wird, bevor der Wert des -timeout Parameters abläuft. Falls der Schlüssel nicht in der zulässigen vorgegebenen Zeit mit der angegebenen Anzahl von Servern synchronisiert wird, wird er nicht erstellt.

AWS CloudHSM synchronisiert automatisch jeden Schlüssel mit jedem HSM im Cluster. Um Ihren Prozess zu beschleunigen, setzen Sie den Wert von min_srv auf weniger als die Anzahl von HSMs im Cluster und legen Sie einen niedrigen Timeout-Wert fest. Beachten Sie jedoch, dass einige Anfragen möglicherweise keinen Schlüssel generieren.

Standard: 1

Erforderlich: Nein

-m_value

Gibt die Anzahl der Benutzer an, die jede kryptografische Operation genehmigen müssen, die den privaten Schlüssel des Paares verwendet. Geben Sie einen Wert von 0 bis 8 ein.

Dieser Parameter legt eine Quorum-Authentifizierungsanforderung für den privaten Schlüssel fest. Der Standardwert, 0, deaktiviert die Quorum-Authentifizierungsfunktion für den Schlüssel. Wenn die Quorumauthentifizierung aktiviert ist, muss die angegebene Anzahl von Benutzern ein Token signieren, um kryptografische Operationen, bei denen der private Schlüssel verwendet wird, sowie Operationen, bei denen der private Schlüssel gemeinsam genutzt oder die gemeinsame Nutzung aufgehoben wird, zu genehmigen.

Um den Wert m_value eines Schlüssels zu ermitteln, verwenden Sie [getKeyInfo](#).

Dieser Parameter ist nur gültig, wenn der -u-Parameter im Befehl das Schlüsselpaar für ausreichend Benutzer freigibt, um die m_value-Anforderung zu erfüllen.

Standard: 0

Erforderlich: Nein

-nex

Macht den privaten Schlüssel nicht extrahierbar. Der generierte private Schlüssel kann nicht [aus dem HSM exportiert](#) werden. Öffentliche Schlüssel sind immer extrahierbar.

Standard: Sowohl der öffentliche als auch der private Schlüssel im Schlüsselpaar können extrahiert werden.

Erforderlich: Nein

-sess

Erstellt einen Schlüssel, der nur in der aktuellen Sitzung existiert. Der Schlüssel kann nach Ende der Sitzung nicht wiederhergestellt werden.

Verwenden Sie diesen Parameter, wenn Sie einen Schlüssel zum Packen nur für kurze Zeit benötigen, z. B. einen Schlüssel, der einen anderen Schlüssel verschlüsselt und dann schnell entschlüsselt. Verwenden Sie keinen Sitzungsschlüssel, um Daten zu verschlüsseln, die Sie nach dem Ende der Sitzung möglicherweise entschlüsseln müssen.

Um einen Sitzungsschlüssel in einen persistenten (Token-)Schlüssel zu ändern, verwenden Sie [setAttribute](#).

Standard: Der Schlüssel ist persistent.

Erforderlich: Nein

-timeout

Gibt an, wie lange (in Sekunden) der Befehl darauf wartet, dass ein Schlüssel mit der im `min_srv` Parameter HSMs angegebenen Anzahl von synchronisiert wird.

Dieser Parameter ist nur gültig, wenn der `min_srv`-Parameter auch im Befehl verwendet wird.

Voreinstellung: Keine Zeitüberschreitung. Der Befehl wartet auf unbestimmte Zeit und kehrt erst zurück, wenn der Schlüssel mit der Mindestanzahl von Servern synchronisiert ist.

Erforderlich: Nein

-u

Teilt den privaten Schlüssel des Paares mit den angegebenen Benutzern. Dieser Parameter gibt anderen HSM-Kryptobenzutzern (CUs) die Erlaubnis, den privaten Schlüssel für kryptografische Operationen zu verwenden. Öffentliche Schlüssel können von jedem Benutzer verwendet werden, ohne sie zu teilen.

Geben Sie eine durch Kommas getrennte Liste von HSM-Benutzern IDs ein, z. B. -u 5,6 Fügen Sie die HSM-Benutzer-ID des aktuellen Benutzers nicht ein. [Um den HSM-Benutzer IDs von CUs auf dem HSM zu finden, verwenden Sie ListUsers](#). Nutzen Sie zum Freigeben oder zum Aufheben der Freigabe vorhandener Schlüssel [shareKey](#) in cloudhsm_mgmt_util.

Standard: Nur der aktuelle Benutzer kann den privaten Schlüssel verwenden.

Erforderlich: Nein

-attest

Führt eine Integritätsprüfung durch, die sicherstellt, dass die Firmware, auf der der Cluster läuft, nicht manipuliert wurde.

Standard: Keine Bescheinigungsprüfung.

Erforderlich: Nein

Verwandte Themen

- [RSAKeyGen-Paar](#)
- [genSymKey](#)
- [ECCKeyGen-Paar](#)

Generieren Sie ein AWS CloudHSM ECC-Schlüsselpaar mit KMU

Verwenden Sie den `genECCKeypair` Befehl im Tool AWS CloudHSM `key_mgmt_util`, um ein ECC-Schlüsselpaar ([Elliptic Curve Cryptography](#)) in Ihren Hardware-Sicherheitsmodulen (HSM) zu generieren. Beim Ausführen des `genECCKeypair`-Befehls müssen Sie die elliptische Kurvenkennung und eine Beschriftung für das Schlüsselpaar angeben. Sie können den privaten Schlüssel auch mit anderen CU-Benutzern teilen sowie nicht extrahierbare Schlüssel, Quorum-kontrollierte Schlüssel und Schlüssel, die bei Sitzungsende ablaufen, erstellen. Wenn der Befehl

erfolgreich ausgeführt wurde, werden die Schlüssel-Handle zurückgegeben, die das HSM zu den öffentlichen und privaten ECC-Schlüsseln zuweist. Sie können die Schlüssel-Handles nutzen, damit die Schlüssel für andere Befehle identifizierbar sind.

Bevor Sie einen `key_mgmt_util`-Befehl ausführen, müssen Sie [key_mgmt_util starten](#) und sich am HSM als Crypto-Benutzer (CU) [anmelden](#).

Tip

Um die Attribute eines von Ihnen erstellten Schlüssels wie Typ, Länge, Bezeichnung und ID zu finden, verwenden Sie [getAttribute](#). Um die Schlüssel für einen bestimmten Benutzer zu finden, verwenden Sie [getKeyInfo](#). Verwenden Sie [findKey](#), um Schlüssel anhand ihrer Attributwerte zu finden.

Syntax

```
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]
```

Beispiele

Die folgenden Beispiele zeigen, wie Sie `genECCKeyPair` ECC-Schlüsselpaare in Ihrem HSMs erstellen können.

Example: Erstellen und Untersuchen eines ECC-Schlüsselpaars

Mit diesem Befehl wird ein ECC-Schlüsselpaar mithilfe einer elliptischen Kurve vom Typ `NID_secp384r1` und der Bezeichnung `ecc14` erstellt. Die Ausgabe zeigt, dass das Schlüssel-Handle des privaten Schlüssels 262177 ist und das Schlüssel-Handle des öffentlichen Schlüssels 262179. Die Bezeichnung gilt sowohl für den öffentlichen als auch den privaten Schlüssel.

```
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
```

Nach dem Generieren des Schlüssels können Sie seine Attribute untersuchen. Verwenden Sie [getAttribute](#), um alle Attribute (dargestellt durch die Konstante 512) des neuen privaten ECC-Schlüssels in die Datei `attr_262177` zu schreiben.

```
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
```

Verwenden Sie dann den `cat`-Befehl, um den Inhalt der `attr_262177`-Attributdatei anzuzeigen. Die Ausgabe zeigt, dass der Schlüssel ein privater Ellipsenkurvenschlüssel ist, der zum Signieren, aber nicht zum Verschlüsseln, Entschlüsseln, Verpacken, Entpacken oder Verifizieren verwendet werden kann. Der Schlüssel ist persistent und exportierbar.

```
$ 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 Verwenden einer ungültigen EEC-Kurve

Mit diesem Befehl wird ein ECC-Schlüsselpaar mithilfe einer NID_X9_62_prime192v1-Kurve erstellt. Da diese elliptische Kurve für den FIPS-Modus HSMs nicht gültig ist, schlägt der Befehl fehl. In der Meldung wird gemeldet, dass ein Server im Cluster nicht verfügbar ist. Dies deutet jedoch normalerweise nicht auf ein Problem mit dem im HSMs Cluster hin.

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

```

Parameters

-h

Zeigt Hilfe für den Befehl an.

Erforderlich: Ja

-i

Gibt die ID für die elliptische Kurve an. Geben Sie eine ID ein.

Zulässige Werte:

- 2: NID_X9_62_prime256v1
- 14: NID_secp384r1
- 16: NID_secp256k1

Erforderlich: Ja

-l

Gibt eine benutzerdefinierte Bezeichnung für das Schlüsselpaar an. Geben Sie eine Zeichenfolge ein. Dieselbe Bezeichnung gilt für beide Schlüssel im Paar. Die maximal zulässige Größe für `label` beträgt 127 Zeichen.

Sie können eine beliebige Phrase verwenden, die Ihnen bei der Identifizierung des Schlüssels hilft. Da die Bezeichnung nicht eindeutig sein muss, können Sie sie verwenden, um Schlüssel zu gruppieren und zu kategorisieren.

Erforderlich: Ja

-id

Gibt einen benutzerdefinierten Bezeichner für das Schlüsselpaar an. Geben Sie eine Zeichenfolge ein, die im Cluster eindeutig ist. Der Standardwert ist eine leere Zeichenfolge. Die von Ihnen angegebene ID gilt für beide Schlüssel im Paar.

Standard : Kein ID-Wert.

Erforderlich: Nein

`-min_srv`

Gibt die Mindestanzahl HSMs an, für die der Schlüssel synchronisiert wird, bevor der Wert des `-timeout` Parameters abläuft. Falls der Schlüssel nicht in der zulässigen vorgegebenen Zeit mit der angegebenen Anzahl von Servern synchronisiert wird, wird er nicht erstellt.

AWS CloudHSM synchronisiert automatisch jeden Schlüssel mit jedem HSM im Cluster. Um Ihren Prozess zu beschleunigen, setzen Sie den Wert von `min_srv` auf weniger als die Anzahl von HSMs im Cluster und legen Sie einen niedrigen Timeout-Wert fest. Beachten Sie jedoch, dass einige Anfragen möglicherweise keinen Schlüssel generieren.

Standard: 1

Erforderlich: Nein

`-m_value`

Gibt die Anzahl der Benutzer an, die jede kryptografische Operation genehmigen müssen, die den privaten Schlüssel des Paares verwendet. Geben Sie einen Wert von 0 bis 8 ein.

Dieser Parameter legt eine Quorum-Authentifizierungsanforderung für den privaten Schlüssel fest. Der Standardwert, 0, deaktiviert die Quorum-Authentifizierungsfunktion für den Schlüssel. Wenn die Quorumauthentifizierung aktiviert ist, muss die angegebene Anzahl von Benutzern ein Token signieren, um kryptografische Operationen, bei denen der private Schlüssel verwendet wird, sowie Operationen, bei denen der private Schlüssel gemeinsam genutzt oder die gemeinsame Nutzung aufgehoben wird, zu genehmigen.

Um den Wert `m_value` eines Schlüssels zu ermitteln, verwenden Sie [getKeyInfo](#).

Dieser Parameter ist nur gültig, wenn der `-u`-Parameter im Befehl das Schlüsselpaar für ausreichend Benutzer freigibt, um die `m_value`-Anforderung zu erfüllen.

Standard: 0

Erforderlich: Nein

`-nex`

Macht den privaten Schlüssel nicht extrahierbar. Der generierte private Schlüssel kann nicht [aus dem HSM exportiert](#) werden. Öffentliche Schlüssel sind immer extrahierbar.

Standard: Sowohl der öffentliche als auch der private Schlüssel im Schlüsselpaar können extrahiert werden.

Erforderlich: Nein

-sess

Erstellt einen Schlüssel, der nur in der aktuellen Sitzung existiert. Der Schlüssel kann nach Ende der Sitzung nicht wiederhergestellt werden.

Verwenden Sie diesen Parameter, wenn Sie einen Schlüssel zum Packen nur für kurze Zeit benötigen, z. B. einen Schlüssel, der einen anderen Schlüssel verschlüsselt und dann schnell entschlüsselt. Verwenden Sie keinen Sitzungsschlüssel, um Daten zu verschlüsseln, die Sie nach dem Ende der Sitzung möglicherweise entschlüsseln müssen.

Um einen Sitzungsschlüssel in einen persistenten (Token-)Schlüssel zu ändern, verwenden Sie [setAttribute](#).

Standard: Der Schlüssel ist persistent.

Erforderlich: Nein

-timeout

Gibt an, wie lange (in Sekunden) der Befehl darauf wartet, dass ein Schlüssel mit der im `min_srv` Parameter HSMs angegebenen Anzahl von synchronisiert wird.

Dieser Parameter ist nur gültig, wenn der `min_srv`-Parameter auch im Befehl verwendet wird.

Voreinstellung: Keine Zeitüberschreitung. Der Befehl wartet auf unbestimmte Zeit und kehrt erst zurück, wenn der Schlüssel mit der Mindestanzahl von Servern synchronisiert ist.

Erforderlich: Nein

-u

Teilt den privaten Schlüssel des Paares mit den angegebenen Benutzern. Dieser Parameter gibt anderen HSM-Kryptobenzern (CUs) die Erlaubnis, den privaten Schlüssel für kryptografische Operationen zu verwenden. Öffentliche Schlüssel können von jedem Benutzer verwendet werden, ohne sie zu teilen.

Geben Sie eine durch Kommas getrennte Liste von HSM-Benutzern IDs ein, z. B. `-u 5,6`. Fügen Sie die HSM-Benutzer-ID des aktuellen Benutzers nicht ein. [Um den HSM-Benutzer IDs von CUs auf dem HSM zu finden, verwenden Sie ListUsers](#). Nutzen Sie zum Freigeben oder zum Aufheben der Freigabe vorhandener Schlüssel [shareKey](#) in `cloudhsm_mgmt_util`.

Standard: Nur der aktuelle Benutzer kann den privaten Schlüssel verwenden.

Erforderlich: Nein

-attest

Führt eine Integritätsprüfung durch, die sicherstellt, dass die Firmware, auf der der Cluster läuft, nicht manipuliert wurde.

Standard: Keine Bescheinigungsprüfung.

Erforderlich: Nein

Verwandte Themen

- [genSymKey](#)
- [RSAKeyGen-Paar](#)
- [DSAKeyGen-Paar](#)

Generieren Sie ein AWS CloudHSM RSA-Schlüsselpaar mit KMU

[Verwenden Sie den genRSAKeyPair Befehl im Tool AWS CloudHSM key_mgmt_util, um ein asymmetrisches RSA-Schlüsselpaar zu generieren.](#) Sie geben den Schlüsseltyp, die Modullänge und einen öffentlichen Exponenten an. Der Befehl generiert ein Modul der angegebenen Länge und erstellt das Schlüsselpaar. Sie können eine ID zuweisen, den Schlüssel mit anderen HSM-Benutzern teilen und nicht extrahierbare Schlüssel sowie Schlüssel, die bei Sitzungsende ablaufen, erstellen. Wenn der Befehl erfolgreich ausgeführt wurde, wird ein Schlüssel-Handle zurückgegeben, das das HSM zum Schlüssel zuweist. Sie können das Schlüssel-Handle nutzen, damit der Schlüssel für andere Befehle identifizierbar ist.

Bevor Sie einen key_mgmt_util-Befehl ausführen, müssen Sie [key_mgmt_util starten](#) und sich am HSM als Crypto-Benutzer (CU) [anmelden](#).

 Tip

Um die Attribute eines von Ihnen erstellten Schlüssels wie Typ, Länge, Bezeichnung und ID zu finden, verwenden Sie [getAttribute](#). Um die Schlüssel für einen bestimmten Benutzer zu finden, verwenden Sie [getKeyInfo](#). Verwenden Sie [findKey](#), um Schlüssel anhand ihrer Attributwerte zu finden.

Syntax

```
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]
```

Beispiele

Diese Beispiele zeigen, wie Sie `genRSAKeyPair` asymmetrische Schlüsselpaare in Ihrem HSMs erstellen können.

Example: Erstellen und Untersuchen eines RSA-Schlüsselpaars

Mit diesem Befehl wird ein RSA-Schlüsselpaar mit einem 2048-Bit-Modul und einem Exponenten von 65537 erstellt. Die Ausgabe zeigt, dass das Schlüssel-Handle des öffentlichen Schlüssels 2100177 ist und das Schlüssel-Handle des privaten Schlüssels 2100426.

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

Der nächste Befehl nutzt [getAttribute](#), um die Attribute des öffentlichen Schlüssels abzurufen, den wir gerade erstellt haben. Die Ausgabe wird in die Datei `attr_2100177` geschrieben. Auf diesen folgt ein `cat`-Befehl, der den Inhalt der Attribut-Datei abrufen. Hilfe zur Interpretation der Schlüsselattribute finden Sie unter [AWS CloudHSM Referenz zu Schlüsselattributen für KMU](#).

Die resultierenden Hexadezimalwerte bestätigen, dass es sich um einen öffentlichen Schlüssel (OBJ_ATTR_CLASS 0x02) mit einem RSA-Typ von (OBJ_ATTR_KEY_TYPE 0x00) handelt. Sie können mit diesem öffentlichen Schlüssel (OBJ_ATTR_ENCRYPT 0x01) verschlüsseln, aber nicht (OBJ_ATTR_DECRYPT 0x00) entschlüsseln. Die Ergebnisse können auch die Schlüssellänge (512, 0x200), das Modul, die Modullänge (2048, 0x800) und den öffentlichen Exponenten (65537, 0x10001) enthalten.

```
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
1c0fe6e3b61aa83c205280119452868f66541bbbffacbbe787b8284fc81deaeeef2b8ec0ba25a077d
6983c77a1de7b17cbe8e15b203868704c6452c2810344a7f2736012424cf0703cf15a37183a1d2d0
97240829f8f90b063dd3a41171402b162578d581980976653935431da0c1260bfe756d85dca63857
d9f27a541676cb9c7def0ef6a2a89c9b9304bcac16fdf8183c0a555421f9ad5dfef534cf26b65873
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: Generieren eines freigegebenen RSA-Schlüsselpaars

Dieser Befehl generiert ein RSA-Schlüsselpaar und teilt den privaten Schlüssel mit Benutzer 4, einem anderen CU auf dem HSM. Der Befehl verwendet den Parameter `m_value`, damit mindestens zwei Genehmigungen erforderlich sind, ehe der private Schlüssel des Paares in einer kryptografischen Operation verwendet werden kann. Wenn Sie den Parameter `m_value` verwenden, müssen Sie auch `-u` im Befehl nutzen. Dabei darf der `m_value` nicht die Gesamtanzahl an Benutzern übersteigen (Anzahl der Werte in `-u` + Eigentümer).

```
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
```

Parameters

-h

Zeigt Hilfe für den Befehl an.

Erforderlich: Ja

-m

Gibt die Länge des Moduls in Bits an. Der minimale Wert beträgt 2048.

Erforderlich: Ja

-e

Gibt den öffentlichen Exponenten an. Bei diesem Wert muss es sich eine ungerade Zahl gleich oder größer als 65537 handeln.

Erforderlich: Ja

-l

Gibt eine benutzerdefinierte Bezeichnung für das Schlüsselpaar an. Geben Sie eine Zeichenfolge ein. Dieselbe Bezeichnung gilt für beide Schlüssel im Paar. Die maximal zulässige Größe für `label` beträgt 127 Zeichen.

Sie können eine beliebige Phrase verwenden, die Ihnen bei der Identifizierung des Schlüssels hilft. Da die Bezeichnung nicht eindeutig sein muss, können Sie sie verwenden, um Schlüssel zu gruppieren und zu kategorisieren.

Erforderlich: Ja

-id

Gibt einen benutzerdefinierten Bezeichner für das Schlüsselpaar an. Geben Sie eine Zeichenfolge ein, die im Cluster eindeutig ist. Der Standardwert ist eine leere Zeichenfolge. Die von Ihnen angegebene ID gilt für beide Schlüssel im Paar.

Standard : Kein ID-Wert.

Erforderlich: Nein

`-min_srv`

Gibt die Mindestanzahl HSMs an, mit der der Schlüssel synchronisiert wird, bevor der Wert des `-timeout` Parameters abläuft. Falls der Schlüssel nicht in der zulässigen vorgegebenen Zeit mit der angegebenen Anzahl von Servern synchronisiert wird, wird er nicht erstellt.

AWS CloudHSM synchronisiert automatisch jeden Schlüssel mit jedem HSM im Cluster. Um Ihren Prozess zu beschleunigen, setzen Sie den Wert von `min_srv` auf weniger als die Anzahl von HSMs im Cluster und legen Sie einen niedrigen Timeout-Wert fest. Beachten Sie jedoch, dass einige Anfragen möglicherweise keinen Schlüssel generieren.

Standard: 1

Erforderlich: Nein

`-m_value`

Gibt die Anzahl der Benutzer an, die jede kryptografische Operation genehmigen müssen, die den privaten Schlüssel des Paares verwendet. Geben Sie einen Wert von 0 bis 8 ein.

Dieser Parameter legt eine Quorum-Authentifizierungsanforderung für den privaten Schlüssel fest. Der Standardwert, 0, deaktiviert die Quorum-Authentifizierungsfunktion für den Schlüssel. Wenn die Quorumauthentifizierung aktiviert ist, muss die angegebene Anzahl von Benutzern ein Token signieren, um kryptografische Operationen, bei denen der private Schlüssel verwendet wird, sowie Operationen, bei denen der private Schlüssel gemeinsam genutzt oder die gemeinsame Nutzung aufgehoben wird, zu genehmigen.

Um den Wert `m_value` eines Schlüssels zu ermitteln, verwenden Sie [getKeyInfo](#).

Dieser Parameter ist nur gültig, wenn der `-u`-Parameter im Befehl das Schlüsselpaar für ausreichend Benutzer freigibt, um die `m_value`-Anforderung zu erfüllen.

Standard: 0

Erforderlich: Nein

`-nex`

Macht den privaten Schlüssel nicht extrahierbar. Der generierte private Schlüssel kann nicht [aus dem HSM exportiert](#) werden. Öffentliche Schlüssel sind immer extrahierbar.

Standard: Sowohl der öffentliche als auch der private Schlüssel im Schlüsselpaar können extrahiert werden.

Erforderlich: Nein

-sess

Erstellt einen Schlüssel, der nur in der aktuellen Sitzung existiert. Der Schlüssel kann nach Ende der Sitzung nicht wiederhergestellt werden.

Verwenden Sie diesen Parameter, wenn Sie einen Schlüssel zum Packen nur für kurze Zeit benötigen, z. B. einen Schlüssel, der einen anderen Schlüssel verschlüsselt und dann schnell entschlüsselt. Verwenden Sie keinen Sitzungsschlüssel, um Daten zu verschlüsseln, die Sie nach dem Ende der Sitzung möglicherweise entschlüsseln müssen.

Um einen Sitzungsschlüssel in einen persistenten (Token-)Schlüssel zu ändern, verwenden Sie [setAttribute](#).

Standard: Der Schlüssel ist persistent.

Erforderlich: Nein

-timeout

Gibt an, wie lange (in Sekunden) der Befehl darauf wartet, dass ein Schlüssel mit der im `min_srv` Parameter HSMs angegebenen Anzahl von synchronisiert wird.

Dieser Parameter ist nur gültig, wenn der `min_srv`-Parameter auch im Befehl verwendet wird.

Voreinstellung: Keine Zeitüberschreitung. Der Befehl wartet auf unbestimmte Zeit und kehrt erst zurück, wenn der Schlüssel mit der Mindestanzahl von Servern synchronisiert ist.

Erforderlich: Nein

-u

Teilt den privaten Schlüssel des Paares mit den angegebenen Benutzern. Dieser Parameter gibt anderen HSM-Kryptobenzutzern (CUs) die Erlaubnis, den privaten Schlüssel für kryptografische Operationen zu verwenden. Öffentliche Schlüssel können von jedem Benutzer verwendet werden, ohne sie zu teilen.

Geben Sie eine durch Kommas getrennte Liste von HSM-Benutzern IDs ein, z. B. `-u 5,6` Fügen Sie die HSM-Benutzer-ID des aktuellen Benutzers nicht ein. [Um den HSM-Benutzer IDs von CUs](#)

[auf dem HSM zu finden, verwenden Sie ListUsers](#). Nutzen Sie zum Freigeben oder zum Aufheben der Freigabe vorhandener Schlüssel [shareKey](#) in `cloudhsm_mgmt_util`.

Standard: Nur der aktuelle Benutzer kann den privaten Schlüssel verwenden.

Erforderlich: Nein

-attest

Führt eine Integritätsprüfung durch, die sicherstellt, dass die Firmware, auf der der Cluster läuft, nicht manipuliert wurde.

Standard: Keine Bescheinigungsprüfung.

Erforderlich: Nein

Verwandte Themen

- [genSymKey](#)
- [DSAKeyGen-Paar](#)
- [ECCKeyGen-Paar](#)

Generieren Sie einen AWS CloudHSM symmetrischen Schlüssel mit KMU

Verwenden Sie den `genSymKey` Befehl im Tool AWS CloudHSM `key_mgmt_util`, um einen symmetrischen Schlüssel in Ihren Hardware-Sicherheitsmodulen (HSM) zu generieren. Sie können Schlüsseltyp und Größe festlegen, eine ID und Bezeichnung zuweisen und den Schlüssel für andere HSM-Benutzer freigeben. Sie können auch nicht extrahierbare Schlüssel erstellen sowie Schlüssel, die ablaufen, wenn die Sitzung endet. Wenn der Befehl erfolgreich ausgeführt wurde, wird ein Schlüssel-Handle zurückgegeben, das das HSM zum Schlüssel zuweist. Sie können das Schlüssel-Handle nutzen, damit der Schlüssel für andere Befehle identifizierbar ist.

Bevor Sie einen `key_mgmt_util`-Befehl ausführen, müssen Sie [key_mgmt_util starten](#) und sich am HSM als Crypto-Benutzer (CU) [anmelden](#).

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]

```

Beispiele

Diese Beispiele zeigen, wie Sie symmetrische Schlüssel in Ihrem erstellen können. `genSymKey` HSMs

Tip

Um die Schlüssel, die Sie mit diesen Beispielen erstellen, für HMAC-Operationen zu verwenden, müssen Sie nach dem Generieren des Schlüssels den Wert `OBJ_ATTR_SIGN` und `OBJ_ATTR_VERIFY` auf `TRUE` festlegen. Verwenden Sie `setAttribute` in CloudHSM Management Utility (CMU), um diese Werte festzulegen. Weitere Informationen finden Sie unter [setAttribute](#).

Example: Generieren eines AES-Schlüssels

Mit diesem Befehl wird ein 256-Bit-AES-Schlüssel mit der Bezeichnung `aes256` erstellt. Die Ausgabe zeigt, dass das Schlüssel-Handle des neuen Schlüssels 6 ist.

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: Erstellen eines Sitzungsschlüssels

Mit diesem Befehl wird ein nicht extrahierbarer 192-Bit-AES-Schlüssel erstellt, der nur für die aktuelle Sitzung gültig ist. Sie können einen solchen Schlüssel erstellen, um einen Schlüssel, der exportiert wird, zu verpacken (und dann sofort zu entpacken).

```
Command: genSymKey -t 31 -s 24 -l tmpAES -id wrap01 -nex -sess
```

Example: Schnelles Zurückgeben

Mit diesem Befehl wird ein generischer 512-Byte-Schlüssel mit der Bezeichnung `IT_test_key` erstellt. Der Befehl wartet nicht darauf, dass der Schlüssel mit allen HSMs im Cluster synchronisiert ist. Stattdessen wird er zurückgegeben, sobald der Schlüssel in einem der HSMs erstellt ist (`-min_srv 1`) oder nach Ablauf von 1 Sekunde (`-timeout 1`), je nachdem, was kürzer ist. Wenn der Schlüssel HSMs vor Ablauf des Timeouts nicht mit der angegebenen Mindestanzahl synchronisiert wird, wird er nicht generiert. Sie können einen solchen Befehl in einem Skript verwenden, das zahlreiche Schlüssel erstellt, wie die `for`-Schleife im folgenden Beispiel.

```
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: Erstellen eines allgemeinen Schlüssels mit Quorum-Autorisierung

Mit diesem Befehl wird ein allgemeiner geheimer 2048-Bit-Schlüssel mit der Bezeichnung `generic-mV2` erstellt. Der Befehl verwendet den `-u`-Parameter, um den Schlüssel mit einem anderen CU, Benutzer 6, zu teilen. Er verwendet den `-m_value`-Parameter, damit für alle kryptografischen Vorgänge, die den Schlüssel verwenden, ein Quorum aus mindestens zwei Genehmigungen erforderlich ist. Der Befehl verwendet auch den `-attest`-Parameter, um die Integrität der Firmware zu überprüfen, auf der der Schlüssel generiert wird.

Die Ausgabe zeigt, dass der Befehl einen Schlüssel mit dem Schlüssel-Handle 9 generiert hat und dass die Bescheinigungsprüfung auf der Cluster-Firmware erfolgreich bestanden wurde.

```
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: Erstellen und Untersuchen eines Schlüssels

Mit diesem Befehl wird ein Triple-DES-Schlüssel mit der Bezeichnung 3DES_shared und der ID IT-02 erstellt. Der Schlüssel kann von dem aktuellen Benutzer und den Benutzern 4 und 5 verwendet werden. Der Befehl schlägt fehl, wenn die ID im Cluster nicht eindeutig ist oder wenn der aktuelle Benutzer der Benutzer 4 oder 5 ist.

Die Ausgabe zeigt, dass der neue Schlüssel das Schlüssel-Handle 7 aufweist.

```
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
```

Um zu überprüfen, ob der neue 3DES-Schlüssel im Besitz des aktuellen Benutzers ist und gemeinsam mit den Benutzern 4 und 5 genutzt wird, verwenden Sie [getKeyInfo](#). Der Befehl verwendet das Handle, das dem neuen Schlüssel zugewiesen wurde (Key Handle: 7).

Die Ausgabe bestätigt, dass der Schlüssel im Besitz des Benutzers 3 ist und gemeinsam mit den Benutzern 4 und 5 verwendet wird.

```
Command: getKeyInfo -k 7

Cfm3GetKey returned: 0x00 : HSM Return: SUCCESS

Owned by user 3
```

also, shared to following 2 user(s):

4, 5

Verwenden Sie [getAttribute](#), um die anderen Eigenschaften des Schlüssels zu bestätigen. Der erste Befehl verwendet `getAttribute`, um alle Attribute (`-a 512`) von Schlüssel-Handle 7 (`-o 7`) abzurufen. Sie werden in die Datei `attr_7` geschrieben. Der zweite Befehl verwendet `cat`, um den Inhalt der `attr_7`-Datei abzurufen.

Mit diesem Befehl wird bestätigt, dass Schlüssel 7 ein symmetrischer 192-Bit- (OBJ_ATTR_VALUE_LEN 0x00000018 oder 24-Byte)-3DES (OBJ_ATTR_KEY_TYPE 0x15)-Schlüssel (OBJ_ATTR_CLASS 0x04) mit der Bezeichnung 3DES_shared (OBJ_ATTR_LABEL 3DES_shared) und der ID IT_02 ist (OBJ_ATTR_ID IT-02). Der Schlüssel ist persistent (OBJ_ATTR_TOKEN 0x01) und extrahierbar (OBJ_ATTR_EXTRACTABLE 0x01) und kann für das Verschlüsseln, Entschlüsseln und Verpacken verwendet werden.

Tip

Um die Attribute eines von Ihnen erstellten Schlüssels wie Typ, Länge, Bezeichnung und ID zu finden, verwenden Sie [getAttribute](#). Um die Schlüssel für einen bestimmten Benutzer zu finden, verwenden Sie [getKeyInfo](#). Verwenden Sie [findKey](#), um Schlüssel anhand ihrer Attributwerte zu finden.

Hilfe zur Interpretation der Schlüsselattribute finden Sie unter [AWS CloudHSM Referenz zu Schlüsselattributen für KMU](#).

```
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

Um die Schlüssel, die Sie mit diesen Beispielen erstellen, für HMAC-Operationen zu verwenden, müssen Sie nach dem Generieren des Schlüssels den Wert `OBJ_ATTR_SIGN` und `OBJ_ATTR_VERIFY` auf `TRUE` festlegen. Verwenden Sie `setAttribute` in CMU, um diese Werte festzulegen. Weitere Informationen finden Sie unter [setAttribute](#).

Parameters

-h

Zeigt Hilfe für den Befehl an.

Erforderlich: Ja

-t

Gibt den Typ des symmetrischen Schlüssels an. Geben Sie die Konstante ein, die den Schlüsseltyp darstellt. Zum Erstellen eines AES-Schlüssels geben Sie beispielsweise `-t 31` ein.

Zulässige Werte:

- 16: [GENERIC_SECRET](#). Ein allgemeiner geheimer Schlüssel ist ein Byte-Array, das keinem speziellen Standard entspricht, wie etwa den Anforderungen an einen AES-Schlüssel.
- 18: [RC4](#). RC4 Schlüssel sind im FIPS-Modus nicht gültig HSMs
- 21: [Triple DES \(3DES\)](#). Gemäß den NIST-Richtlinien ist dies für Cluster im FIPS-Modus nach 2023 nicht zulässig. Für Cluster im Nicht-FIPS-Modus ist dies auch nach 2023 zulässig. Details dazu finden Sie unter [FIPS-140-Konformität: Mechanismus 2024 nicht mehr unterstützt](#).
- 31: [AES](#)

Erforderlich: Ja

-s

Gibt die Schlüsselgröße in Byte an. Um beispielsweise einen 192-Bit-Schlüssel zu erstellen, geben Sie 24 ein.

Gültige Werte für jeden Schlüsseltyp:

- AES: 16 (128 Bit), 24 (192 Bit), 32 (256 Bit)
- 3DES: 24 (192 Bit)
- Allgemeiner geheimer Schlüssel: <3584 (28672 Bit)

Erforderlich: Ja

-l

Gibt eine benutzerdefinierte Bezeichnung für den Schlüssel an. Geben Sie eine Zeichenfolge ein.

Sie können eine beliebige Phrase verwenden, die Ihnen bei der Identifizierung des Schlüssels hilft. Da die Bezeichnung nicht eindeutig sein muss, können Sie sie verwenden, um Schlüssel zu gruppieren und zu kategorisieren.

Erforderlich: Ja

-attest

Führt eine Integritätsprüfung durch, die sicherstellt, dass die Firmware, auf der der Cluster läuft, nicht manipuliert wurde.

Standard: Keine Bescheinigungsprüfung.

Erforderlich: Nein

-id

Gibt einen benutzerdefinierten Bezeichner für den Schlüssel an. Geben Sie eine Zeichenfolge ein, die im Cluster eindeutig ist. Der Standardwert ist eine leere Zeichenfolge.

Standard : Kein ID-Wert.

Erforderlich: Nein

-min_srv

Gibt die Mindestanzahl HSMs an, für die der Schlüssel synchronisiert wird, bevor der Wert des `-timeout` Parameters abläuft. Falls der Schlüssel nicht in der zulässigen vorgegebenen Zeit mit der angegebenen Anzahl von Servern synchronisiert wird, wird er nicht erstellt.

AWS CloudHSM synchronisiert automatisch jeden Schlüssel mit jedem HSM im Cluster. Um Ihren Prozess zu beschleunigen, setzen Sie den Wert von `min_srv` auf weniger als die Anzahl von HSMs im Cluster und legen Sie einen niedrigen Timeout-Wert fest. Beachten Sie jedoch, dass einige Anfragen möglicherweise keinen Schlüssel generieren.

Standard: 1

Erforderlich: Nein

-m_value

Gibt die Anzahl der Benutzer an, die einen kryptografischen Vorgang genehmigen müssen, der den Schlüssel verwendet. Geben Sie einen Wert von 0 bis 8 ein.

Dieser Parameter legt eine Quorum-Authentifizierungsanforderung für den Schlüssel fest. Der Standardwert, 0, deaktiviert die Quorum-Authentifizierungsfunktion für den Schlüssel. Wenn

die Quorumauthentifizierung aktiviert ist, muss die angegebene Anzahl von Benutzern ein Token signieren, um kryptografische Operationen, bei denen der Schlüssel verwendet wird, sowie Operationen, bei denen der Schlüssel gemeinsam genutzt oder die gemeinsame Nutzung aufgehoben wird, zu genehmigen.

Um den Wert `m_value` eines Schlüssels zu ermitteln, verwenden Sie [getKeyInfo](#).

Dieser Parameter ist nur gültig, wenn der `-u`-Parameter im Befehl den Schlüssel für ausreichend Benutzer freigibt, um die `m_value`-Anforderung zu erfüllen.

Standard: 0

Erforderlich: Nein

`-nex`

Macht den Schlüssel nicht extrahierbar. Der generierte Schlüssel kann nicht [aus dem HSM exportiert](#) werden.

Standard: Der Schlüssel ist extrahierbar.

Erforderlich: Nein

`-sess`

Erstellt einen Schlüssel, der nur in der aktuellen Sitzung existiert. Der Schlüssel kann nach Ende der Sitzung nicht wiederhergestellt werden.

Verwenden Sie diesen Parameter, wenn Sie einen Schlüssel zum Packen nur für kurze Zeit benötigen, z. B. einen Schlüssel, der einen anderen Schlüssel verschlüsselt und dann schnell entschlüsselt. Verwenden Sie keinen Sitzungsschlüssel, um Daten zu verschlüsseln, die Sie nach dem Ende der Sitzung möglicherweise entschlüsseln müssen.

Um einen Sitzungsschlüssel in einen persistenten (Token-)Schlüssel zu ändern, verwenden Sie [setAttribute](#).

Standard: Der Schlüssel ist persistent.

Erforderlich: Nein

`-timeout`

Gibt an, wie lange (in Sekunden) der Befehl darauf wartet, dass ein Schlüssel mit der im `min_srv` Parameter HSMs angegebenen Anzahl von synchronisiert wird.

Dieser Parameter ist nur gültig, wenn der `min_srv`-Parameter auch im Befehl verwendet wird.

Voreinstellung: Keine Zeitüberschreitung. Der Befehl wartet auf unbestimmte Zeit und kehrt erst zurück, wenn der Schlüssel mit der Mindestanzahl von Servern synchronisiert ist.

Erforderlich: Nein

-u

Gibt den Schlüssel für die angegebenen Benutzer frei. Dieser Parameter gibt anderen HSM-Kryptobenzutzern (CUs) die Erlaubnis, diesen Schlüssel für kryptografische Operationen zu verwenden.

Geben Sie eine durch Kommas getrennte Liste von HSM-Benutzern IDs ein, z. B. `-u 5,6`. Fügen Sie die HSM-Benutzer-ID des aktuellen Benutzers nicht ein. [Um den HSM-Benutzer IDs von CUs auf dem HSM zu finden, verwenden Sie ListUsers](#). Nutzen Sie zum Freigeben oder zum Aufheben der Freigabe vorhandener Schlüssel [shareKey](#) in `cloudhsm_mgmt_util`.

Standard: Nur der aktuelle Benutzer kann den Schlüssel verwenden.

Erforderlich: Nein

Verwandte Themen

- [exSymKey](#)
- [RSAKeyGen-Paar](#)
- [DSAKeyGen-Paar](#)
- [ECCKeyGen-Paar](#)
- [setAttribute](#)

Rufen Sie ein AWS CloudHSM identifizierendes Attribut mit KMU ab

Verwenden Sie den `getAttribute` Befehl in AWS CloudHSM `key_mgmt_util`, um einen oder alle Attributwerte für einen Schlüssel in eine Datei zu schreiben. AWS CloudHSM Wenn das von Ihnen angegebene Attribut für den Schlüsseltyp nicht existiert (z. B. der Modulus eines AES-Schlüssels), gibt `getAttribute` einen Fehler zurück.

Schlüsselattribute sind Eigenschaften eines Schlüssels. Dazu zählen Merkmale wie der Schlüsseltyp, Klasse, Beschriftung und ID sowie Werte, die Aktionen darstellen, die Sie mit dem Schlüssel ausführen können, wie verschlüsseln, entschlüsseln, packen, signieren und überprüfen.

Sie können `getAttribute` nur für Schlüssel ausführen, die Ihnen gehören oder die für Sie freigegeben wurden. Sie können diesen Befehl oder den Befehl [getAttribute](#) in `cloudhsm_mgmt_util` ausführen, wodurch ein Attributwert eines Schlüssels von allen HSMs in einem Cluster abgerufen und in die Standardausgabe oder in eine Datei geschrieben wird.

Zum Abrufen einer Liste der Attribute und Konstanten, die sie darstellen, verwenden Sie den Befehl [listAttributes](#). Um die Attributwerte von vorhandenen Schlüsseln zu ändern, verwenden Sie [setAttribute](#) in `key_mgmt_util` und [setAttribute](#) in `cloudhsm_mgmt_util`. Hilfe zur Interpretation der Schlüsselattribute finden Sie unter [AWS CloudHSM Referenz zu Schlüsselattributen für KMU](#).

Bevor Sie einen `key_mgmt_util`-Befehl ausführen, müssen Sie [key_mgmt_util starten](#) und sich am HSM als Crypto-Benutzer (CU) [anmelden](#).

Syntax

```
getAttribute -h

getAttribute -o <key handle>
              -a <attribute constant>
              -out <file>
```

Beispiele

Diese Beispiele zeigen, wie Sie die Attribute von Schlüsseln in Ihrem abrufen können. `getAttribute` HSMs

Example: Ruft den Schlüsseltyp ab

In diesem Beispiel wird der Schlüsseltyp abgerufen (z. B. ein AES- oder 3DES-Schlüssel, ein generischer Schlüssel oder ein RSA-Schlüsselpaar oder elliptisches Kurvenschlüsselpaar).

Der erste Befehl führt [listAttributes](#) aus, das die Schlüsselattribute und deren Konstanten abruft. Die Ausgabe zeigt, dass die Konstante für den Schlüsseltyp 256 lautet. Hilfe zur Interpretation der Schlüsselattribute finden Sie unter [AWS CloudHSM Referenz zu Schlüsselattributen für KMU](#).

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_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

Der zweite Befehl führt `getAttribute` aus. Er fordert den Schlüsseltyp (Attribut 256) für das Schlüsselhandle 524296 an und schreibt ihn in die `attribute.txt`-Datei.

Command: **`getAttribute -o 524296 -a 256 -out attribute.txt`**
 Attributes dumped into `attribute.txt` file

Mit dem letzten Befehl wird der Inhalt der Schlüsseldatei abgerufen. Die Ausgabe zeigt, dass es sich bei dem Schlüsseltyp um `0x15` oder 21 handelt – ein Triple DES-Schlüssel (3DES). Definitionen der Klassen- und Typwerte finden Sie in der [Schlüsselattributreferenz](#).

```
$ cat attribute.txt
OBJ_ATTR_KEY_TYPE
0x00000015
```

Example: Ruft alle Attribute eines Schlüssels ab

Dieser Befehl ruft alle Attribute des Schlüssels mit dem Schlüsselhandle 6 ab und schreibt sie in die `attr_6`-Datei. Er verwendet den Attributwert 512, der alle Attribute repräsentiert.

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>
```

Dieser Befehl zeigt den Inhalt einer Beispiel-Attributdatei mit allen Attributwerten an. Neben den Werten zeigt er auch an, dass es sich bei dem Schlüssel um einen 256-Bit-AES-Schlüssel mit der ID `test_01` von und der Bezeichnung `aes256` handelt. Der Schlüssel ist extrahierbar und persistent, d. h., es handelt sich nicht um einen Session-only-Schlüssel. Hilfe zur Interpretation der Schlüsselattribute finden Sie unter [AWS CloudHSM Referenz zu Schlüsselattributen für KMU](#).

```
$ 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
```

Parameters

-h

Zeigt Hilfe für den Befehl an.

Erforderlich: Ja

-o

Gibt das Schlüssel-Handle des Zielschlüssels an. Sie können nur jeweils einen Schlüssel in den einzelnen Befehlen angeben. Verwenden Sie zum Abrufen des Schlüssel-Handles eines Schlüssels [findKey](#).

Sie müssen außerdem Besitzer des angegebenen Schlüssels sein oder er muss für Sie freigegeben sein. Um die Benutzer eines Schlüssels zu finden, verwenden Sie [getKeyInfo](#).

Erforderlich: Ja

-a

Bezeichnet das Attribut. Geben Sie eine Konstante, die ein Attribut darstellt, oder den Wert 512 ein, der alle Attribute repräsentiert. Zum Abrufen des Schlüsseltyps geben Sie z. B. 256 ein. Dies ist die Konstante für das Attribut OBJ_ATTR_KEY_TYPE.

Verwenden Sie [listAttributes](#), um die Attribute und deren Konstanten aufzulisten. Hilfe zur Interpretation der Schlüsselattribute finden Sie unter [AWS CloudHSM Referenz zu Schlüsselattributen für KMU](#).

Erforderlich: Ja

-out

Schreibt die Ausgabe in die angegebene Datei. Geben Sie einen Dateipfad ein. Sie können die Ausgabe nicht in die Datei stdout schreiben.

Wenn die angegebene Datei vorhanden ist, überschreibt getAttribute die Datei ohne Warnung.

Erforderlich: Ja

Verwandte Themen

- [getAttribute](#) in cloudhsm_mgmt_util
- [listAttributes](#)
- [setAttribute](#)
- [findKey](#)
- [Schlüsselattributreferenz](#)

Exportieren Sie einen AWS CloudHSM Schlüssel mit KMU in ein gefälschtes PEM-Format

Verwenden Sie den Befehl `getCaviumPrivKey` in der Datei AWS CloudHSM `key_mgmt_util`, um einen privaten Schlüssel aus einem Hardware-Sicherheitsmodul (HSM) im gefälschten PEM-Format zu exportieren. Die gefälschte PEM-Datei, die nicht das eigentliche Material des privaten Schlüssels enthält, sondern stattdessen auf den privaten Schlüssel im HSM verweist, kann dann verwendet werden, um die Übertragung von Ihrem Webserver zu einzurichten. SSL/TLS AWS CloudHSM Weitere Informationen finden Sie unter [SSL/TLS-Offload unter Linux mit Tomcat oder SSL/TLS-Offload unter Linux mit NGINX oder Apache](#).

Bevor Sie einen `key_mgmt_util`-Befehl ausführen, müssen Sie [key_mgmt_util starten](#) und sich am HSM als Crypto-Benutzer (CU) [anmelden](#).

Syntax

```
getCaviumPrivKey -h

getCaviumPrivKey -k <private-key-handle>
                  -out <fake-PEM-file>
```

Beispiele

In diesem Beispiel wird gezeigt, wie Sie mit `getCaviumPrivKey` einen privaten Schlüssel im gefälschten PEM-Format exportieren.

Example: Exportieren einer gefälschten PEM-Datei

Dieser Befehl erstellt und exportiert eine gefälschte PEM-Version eines privaten Schlüssels mit dem Handle 15 und speichert sie in einer Datei namens `cavKey.pem`. Wird der Befehl erfolgreich ausgeführt, gibt `exportPrivateKey` eine Erfolgsmeldung zurück.

```
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
```

Parameters

Dieser Befehl erfordert die folgenden Parameter.

-h

Zeigt die Befehlszeilenhilfe für den Befehl an.

Erforderlich: Ja

-k

Gibt das Schlüssel-Handle des privaten Schlüssels an, der im gefälschten PEM-Format exportiert werden soll.

Erforderlich: Ja

-out

Gibt den Namen der Datei an, in die der gefälschte PEM-Schlüssel geschrieben werden soll.

Erforderlich: Ja

Verwandte Themen

- [importPrivateKey](#)
- [SSL/TLS Offload unter Linux mit Tomcat](#)
- [SSL/TLS-Offload unter Linux mit NGINX oder Apache](#)

Holen Sie sich HSM-Partitionszertifikate mit KMU AWS CloudHSM

Verwenden Sie den `getCert` Befehl in AWS CloudHSM `key_mgmt_util`, um die Partitionszertifikate eines Hardware-Sicherheitsmoduls (HSM) abzurufen und sie in einer Datei zu speichern. Wenn Sie den Befehl ausführen, geben Sie den Typ des abzurufenden Zertifikats an. Verwenden Sie dazu eine der entsprechenden ganzzahligen Werte, wie im folgenden Abschnitt [Parameter](#) beschrieben. Weitere Informationen über die Rolle der einzelnen Zertifikate finden Sie unter [Überprüfen der HSM-Identität](#).

Bevor Sie einen `key_mgmt_util`-Befehl ausführen, müssen Sie [key_mgmt_util starten](#) und sich am HSM als Crypto-Benutzer (CU) [anmelden](#).

Syntax

```
getCert -h

getCert -f <file-name>
        -t <certificate-type>
```

Beispiel

Dieses Beispiel zeigt, wie man `getCert` verwendet, um das Kundenstammzertifikat eines Clusters abzurufen und als Datei zu speichern.

Example: Abrufen eines Kundenstammzertifikats

Dieser Befehl exportiert ein Kundenstammzertifikat (dargestellt durch die Ganzzahl 4) und speichert es in einer Datei namens `userRoot.crt`. Wird der Befehl erfolgreich ausgeführt, gibt `getCert` eine Erfolgsmeldung zurück.

```
Command: getCert -f userRoot.crt -s 4

Cfm3GetCert() returned 0 :HSM Return: SUCCESS
```

Parameters

Dieser Befehl erfordert die folgenden Parameter.

-h

Zeigt die Befehlszeilenhilfe für den Befehl an.

Erforderlich: Ja

-f

Gibt den Namen der Datei an, in der das abgerufene Zertifikat gespeichert wird.

Erforderlich: Ja

-s

Eine Ganzzahl, die den Typ des abzurufenden Partitionszertifikats angibt. Die Ganzzahlen und ihre entsprechenden Zertifikattypen sind:

- 1 – Hersteller-Stammzertifikat
- 2 – Hersteller-Hardwarezertifikat
- 4 – Kunden-Stammzertifikat
- 8 – Clusterzertifikat (signiert vom Kunden-Stammzertifikat)
- 16 – Clusterzertifikat (verkettet mit dem Hersteller-Stammzertifikat)

Erforderlich: Ja

Verwandte Themen

- [HSM-Identität überprüfen](#)

Holen Sie sich die Benutzer eines AWS CloudHSM Schlüssels mithilfe von KMU

Verwenden Sie den `getKeyInfo` Befehl in AWS CloudHSM `key_mgmt_util`, um den Benutzer des Hardware-Sicherheitsmoduls (HSM) von Benutzern zurückzugeben, die den Schlüssel verwenden können, einschließlich des Besitzers und IDs der Crypto-Benutzer (CU), mit denen der Schlüssel gemeinsam genutzt wird. Wenn die Quorum-Authentifizierung für einen Schlüssel aktiviert ist, gibt `getKeyInfo` auch die Anzahl der Benutzer zurück, die kryptografische Operationen genehmigen müssen, von denen der Schlüssel verwendet wird. Sie können `getKeyInfo` nur auf Schlüsseln ausführen, die Ihnen gehören oder die für Sie freigegeben wurden.

Wenn Sie `getKeyInfo` auf öffentliche Schlüsseln ausführen, gibt `getKeyInfo` nur den Schlüsseleigentümer zurück, auch wenn alle HSM-Benutzer den öffentlichen Schlüssel verwenden können. Um den HSM-Benutzer IDs von Benutzern in Ihrem zu finden HSMs, verwenden Sie [ListUsers](#). Um nach den Schlüsseln für einen bestimmten Benutzer zu suchen, verwenden Sie [findKey](#) -u.

Ihren gehören die Schlüssel, die Sie erstellen. Sie können einen Schlüssel für andere Benutzer freigeben, wenn Sie ihn erstellen. Nutzen Sie dann zum Freigeben oder zum Aufheben der Freigabe [shareKey](#) in `cloudhsm_mgmt_util`.

Bevor Sie einen `key_mgmt_util`-Befehl ausführen, müssen Sie [key_mgmt_util starten](#) und sich am HSM als Crypto-Benutzer (CU) [anmelden](#).

Syntax

```
getKeyInfo -h  
  
getKeyInfo -k <key-handle>
```

Beispiele

Diese Beispiele veranschaulichen die Verwendung von `getKeyInfo` zum Abrufen von Informationen über die Benutzer eines Schlüssels.

Example: Abfragen der Benutzer eines symmetrischen Schlüssels

Über diesen Befehl erhalten Sie die Benutzer, die den (symmetrischen) AES-Schlüssel mit dem Schlüssel-Handle 9 verwenden können. Die Ausgabe zeigt, dass Benutzer 3 den Schlüssel besitzt und mit Benutzer 4 geteilt hat.

```
Command: getKeyInfo -k 9  
  
Cfm3GetKey returned: 0x00 : HSM Return: SUCCESS  
  
Owned by user 3  
  
also, shared to following 1 user(s):  
  
4
```

Example: Abfragen der Benutzer eines asymmetrischen Schlüsselpaars

Diese Befehle verwenden `getKeyInfo`, um die Benutzer abzurufen, die die Schlüssel in einem (asymmetrischen) RSA-Schlüsselpaar verwenden können. Das Schlüssel-Handle des öffentlichen Schlüssels ist 21. Das Schlüssel-Handle des privaten Schlüssels ist 20.

Wenn Sie `getKeyInfo` den privaten Schlüssel (20) verwenden, werden der Schlüsselbesitzer (3) und die Krypto-Benutzer (CUs) 4 und 5 zurückgegeben, mit denen der Schlüssel geteilt wird.

Command: **getKeyInfo -k 20**

Cfm3GetKey returned: 0x00 : HSM Return: SUCCESS

Owned by user 3

also, shared to following 2 user(s):

4

5

Wenn Sie `getKeyInfo` auf dem öffentlichen Schlüssel (21) ausführen, wird nur der Schlüsseleigentümer (3) zurückgegeben.

Command: **getKeyInfo -k 21**

Cfm3GetKey returned: 0x00 : HSM Return: SUCCESS

Owned by user 3

Um zu bestätigen, dass Benutzer 4 den öffentlichen Schlüssel (und alle öffentlichen Schlüssel im HSM) verwenden kann, verwenden Sie den `-u`-Parameter von [findKey](#).

Die Ausgabe zeigt, dass Benutzer 4 sowohl den öffentlichen (21) als auch den privaten Schlüssel (20) im Schlüsselpaar verwenden kann. Benutzer 4 kann zudem alle anderen öffentlichen und privaten Schlüssel nutzen, die sie erstellt haben oder die für sie freigegeben wurden.

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: Abrufen des Quorum-Authentifizierungswerts (m_value) für einen Schlüssel

Mit diesem Beispiel wird gezeigt, wie der `m_value` für einen Schlüssel abgerufen werden kann, d. h. die Anzahl der Benutzer im Quorum, die alle kryptografischen Vorgänge genehmigen müssen, bei denen der Schlüssel verwendet wird.

Wenn die Quorum-Authentifizierung für einen Schlüssel aktiviert ist, muss das Quorum der Benutzer alle kryptografischen Operationen genehmigen, bei denen der Schlüssel verwendet wird. Zum Aktivieren der Quorum-Authentifizierung und Festlegen der Quorum-Größe nutzen Sie beim Erstellen des Schlüssels den Parameter `-m_value`.

Dieser Befehl verwendet [gen RSAKey Pair](#), um ein RSA-Schlüsselpaar zu erstellen, das mit Benutzer 4 geteilt wird. Er verwendet den Parameter `m_value` zur Aktivierung der Quorum-Authentifizierung für den privaten Schlüssel im Paar und zur Festlegung der Quorum-Größe auf zwei Benutzer. Die Anzahl der Benutzer muss groß genug sein, um die erforderlichen Genehmigungen bereitstellen zu können.

Die Ausgabe zeigt, dass der Befehl den öffentlichen Schlüssel 27 und den privaten Schlüssel 28 erstellt hat.

```
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
```

Dieser Befehl nutzt `getKeyInfo` zum Abrufen von Informationen über die Benutzer des privaten Schlüssels. Die Ausgabe zeigt, dass der Schlüssel im Besitz des Benutzers 3 ist und gemeinsam mit Benutzer 4 verwendet wird. Sie zeigt auch, dass ein Quorum von zwei Benutzern jede kryptografische Operation genehmigen muss, bei der der Schlüssel verwendet wird.

```
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

Parameters

-h

Zeigt die Befehlszeilenhilfe für den Befehl an.

Erforderlich: Ja

-k

Gibt das Schlüssel-Handle eines Schlüssels im HSM an. Geben Sie das Schlüssel-Handle eines Schlüssels ein, den Sie besitzen oder teilen. Dieser Parameter muss angegeben werden.

Verwenden Sie den [findKey](#)-Befehl, um Schlüssel-Handles zu finden.

Erforderlich: Ja

Verwandte Themen

- [getKeyInfo](#) in cloudhsm_mgmt_util
- [listUsers](#)
- [findKey](#)
- [findAllKeys](#) in cloudhsm_mgmt_util

Hilfeinformationen für AWS CloudHSM KMU anzeigen

Verwenden Sie den Befehl in key_mgmt_util, um Informationen zu allen verfügbaren help key_mgmt_util-Befehlen anzuzeigen. AWS CloudHSM

Vor dem Ausführen von help müssen Sie [key_mgmt_util starten](#).

Syntax

```
help
```

Beispiel

Dieses Beispiel zeigt die Ausgabe des `help`-Befehls.

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

<code>createPublicKey</code>	Creates an RSA public key
<code>importPubKey</code>	Imports RSA/DSA/EC Public key
<code>exportPubKey</code>	Exports RSA/DSA/EC Public key
<code>importPrivateKey</code>	Imports RSA/DSA/EC private key
<code>exportPrivateKey</code>	Exports RSA/DSA/EC private key
<code>imSymKey</code>	Imports a Symmetric key
<code>exSymKey</code>	Exports a Symmetric key
<code>wrapKey</code>	Wraps a key from from HSM using the specified handle
<code>unwrapKey</code>	UnWraps a key into HSM using the specified handle

Key Management Commands

<code>deleteKey</code>	Delete Key
<code>setAttribute</code>	Sets an attribute of an object
<code>getKeyInfo</code>	Get Key Info about shared users/sessions
<code>findKey</code>	Find Key
<code>findSingleKey</code>	Find single Key
<code>getAttribute</code>	Reads an attribute from an object

Certificate Setup Commands

<code>getCert</code>	Gets Partition Certificates stored on HSM

Key Transfer Commands

<code>insertMaskedObject</code>	Inserts a masked object
<code>extractMaskedObject</code>	Extracts a masked object

Management Crypto Commands

<code>sign</code>	Generates a signature
<code>verify</code>	Verifies a signature
<code>aesWrapUnwrap</code>	Does NIST AES Wrap/Unwrap

Helper Commands

<code>Error2String</code>	Converts Error codes to Strings
<code>save key handle in fake PEM format</code>	
<code>getCaviumPrivKey</code>	Saves an RSA private key handle in fake PEM format
<code>IsValidKeyHandlefile</code>	Checks if private key file has an HSM key handle or a real key
<code>listAttributes</code>	List all attributes for getAttributes
<code>listECCCurveIds</code>	List HSM supported ECC CurveIds

Parameters

Für diesen Befehl gibt es keine Parameter.

Verwandte Themen

- [loginHSM und logoutHSM](#)

Importieren Sie einen privaten Schlüssel mit AWS CloudHSM KMU

Verwenden Sie den `importPrivateKey` Befehl in AWS CloudHSM `key_mgmt_util`, um einen asymmetrischen privaten Schlüssel aus einer Datei in ein Hardware-Sicherheitsmodul (HSM) zu importieren. Das HSM erlaubt keinen direkten Import von Schlüsseln im Klartext. Der Befehl verschlüsselt den privaten Schlüssel mit einem von Ihnen angegebenen AES-Wrapping-Schlüssel und entpackt den Schlüssel innerhalb des HSM. [Wenn Sie versuchen, einem Zertifikat einen AWS CloudHSM Schlüssel zuzuordnen, finden Sie weitere Informationen in diesem Thema.](#)

Note

Sie können einen passwortgeschützten PEM-Schlüssel nicht mithilfe eines symmetrischen oder privaten Schlüssels importieren.

Sie müssen einen AES-Wrapping-Schlüssel angeben, der `OBJ_ATTR_UNWRAP`- und `OBJ_ATTR_ENCRYPT`-Attributwerte 1 hat. Mit dem Befehl [getAttribute](#) können Sie nach den Attributen eines Schlüssels suchen.

Note

Dieser Befehl bietet keine Option, um den importierten Schlüssel als nicht exportierbar zu markieren.

Bevor Sie einen `key_mgmt_util`-Befehl ausführen, müssen Sie [key_mgmt_util starten](#) und sich am HSM als Crypto-Benutzer (CU) [anmelden](#).

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]

```

Beispiele

In diesem Beispiel wird gezeigt, wie Sie mit `importPrivateKey` einen privaten Schlüssel in ein HSM importieren.

Example: Importieren eines privaten Schlüssels

Dieser Befehl importiert den privaten Schlüssel aus einer Datei mit dem Namen `rsa2048.key`, der Bezeichnung `rsa2048-imported` und einem Verpackungsschlüssel mit dem Handle `524299`. Wenn der Befehl erfolgreich ausgeführt wurde, gibt `importPrivateKey` ein Schlüssel-Handle für den importierten Schlüssel sowie eine Erfolgsmeldung zurück.

```
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
```

Parameters

Dieser Befehl erfordert die folgenden Parameter.

-h

Zeigt die Befehlszeilenhilfe für den Befehl an.

Erforderlich: Ja

-l

Gibt die benutzerdefinierte Bezeichnung des privaten Schlüssels an.

Erforderlich: Ja

-f

Gibt den Dateinamen des zu importierenden Schlüssels an.

Erforderlich: Ja

-w

Gibt das Schlüssel-Handle des Verpackungsschlüssels an. Dieser Parameter muss angegeben werden. Nutzen Sie den Befehl [findKey](#), um Schlüssel-Handles zu suchen.

Ermitteln Sie mithilfe von [getAttribute](#) den Wert des OBJ_ATTR_WRAP-Attributs (262), um zu bestimmen, ob ein Schlüssel als Verpackungsschlüssel verwendet werden kann. Um einen Wrapping-Schlüssel zu erstellen, verwenden Sie [genSymKey](#), um einen AES-Schlüssel (Typ 31) zu generieren.

Wenn Sie den Parameter `-wk` zum Angeben eines externen Entpackungsschlüssels verwenden, wird der `-w`-Verpackungsschlüssel während des Imports zum Verpacken, nicht aber zum Entpacken des Schlüssels verwendet.

Erforderlich: Ja

-sess

Gibt den importierten Schlüssel als Sitzungsschlüssel an.

Standard: Der importierte Schlüssel wird im Cluster als persistenter Schlüssel (Token) bereitgehalten.

Erforderlich: Nein

-id

Gibt die ID des zu importierenden Schlüssels an.

Standard : Kein ID-Wert.

Erforderlich: Nein

-m_value

Gibt die Anzahl der Benutzer an, die einen kryptografischen Vorgang genehmigen müssen, der den importierten Schlüssel verwendet. Geben Sie einen Wert zwischen **0** und **8** ein.

Dieser Parameter ist nur gültig, wenn der -u-Parameter im Befehl den Schlüssel für ausreichend Benutzer freigibt, um die m_value-Anforderung zu erfüllen.

Standard: 0

Erforderlich: Nein

-min_srv

Gibt die Mindestanzahl HSMs an, mit der der importierte Schlüssel synchronisiert wird, bevor der Wert des -timeout Parameters abläuft. Falls der Schlüssel nicht in der zulässigen vorgegebenen Zeit mit der angegebenen Anzahl von Servern synchronisiert wird, wird er nicht erstellt.

AWS CloudHSM synchronisiert automatisch jeden Schlüssel mit jedem HSM im Cluster. Um Ihren Prozess zu beschleunigen, setzen Sie den Wert von min_srv auf weniger als die Anzahl von HSMs im Cluster und legen Sie einen niedrigen Timeout-Wert fest. Beachten Sie jedoch, dass einige Anfragen möglicherweise keinen Schlüssel generieren.

Standard: 1

Erforderlich: Nein

-timeout

Gibt an, wie viele Sekunden auf die Synchronisierung des Schlüssels gewartet werden soll, HSMs wenn der min-serv Parameter enthalten ist. Wenn keine Anzahl angegeben ist, wird die Abfrage ohne zeitliche Einschränkung fortgesetzt.

Standard: keine Einschränkung

Erforderlich: Nein

-u

Gibt die Liste der Benutzer an, für die der importierte private Schlüssel freigegeben werden soll. Dieser Parameter gibt anderen HSM-Kryptobenzutzern (CUs) die Erlaubnis, den importierten Schlüssel für kryptografische Operationen zu verwenden.

Geben Sie eine durch Kommas getrennte Liste von HSM-Benutzern ein, z. B. IDs -u 5,6. Fügen Sie die HSM-Benutzer-ID des aktuellen Benutzers nicht ein. [Um den HSM-Benutzer IDs von CUs auf dem HSM zu finden, verwenden Sie ListUsers.](#)

Standard: Nur der aktuelle Benutzer kann den importierten Schlüssel verwenden.

Erforderlich: Nein

-wk

Gibt den Schlüssel an, mit dem der Schlüssel, der importiert wird, verpackt werden soll. Geben Sie den Pfad und den Namen einer Datei an, die einen Klartext-AES-Schlüssel enthält.

Wenn Sie diesen Parameter einschließen, verwendet importPrivateKey den Schlüssel in der -wk-Datei, um den zu importierenden Schlüssel zu verpacken. Außerdem wird der vom -w-Parameter angegebene Schlüssel zum Entpacken verwendet.

Standard: Verwenden Sie den im -w-Parameter angegebenen Verpackungsschlüssel für das Verpacken und Entpacken von Schlüsseln.

Erforderlich: Nein

-attest

Überprüft die Firmware-Antwort, um sicherzustellen, dass die Firmware, auf der der Cluster ausgeführt wird, nicht beeinträchtigt wurde.

Erforderlich: Nein

Verwandte Themen

- [wrapKey](#)
- [unWrapKey](#)
- [genSymKey](#)

- [exportPrivateKey](#)

Importieren Sie einen öffentlichen Schlüssel mit AWS CloudHSM KMU

Verwenden Sie den `importPubKey` Befehl in AWS CloudHSM `key_mgmt_util`, um einen öffentlichen Schlüssel im PEM-Format in ein Hardware-Sicherheitsmodul (HSM) zu importieren. Sie können ihn verwenden, um öffentliche und außerhalb des HSM erstellte Schlüssel zu importieren. Mit diesem Befehl ist auch das Importieren von Schlüsseln möglich, die aus einem HSM exportiert wurden, z. B. über den Befehl [exportPubKey](#) exportierte Schlüssel.

Bevor Sie einen `key_mgmt_util`-Befehl ausführen, müssen Sie [key_mgmt_util starten](#) und sich am HSM als Crypto-Benutzer (CU) [anmelden](#).

Syntax

```
importPubKey -h

importPubKey -l <label>
               -f <key-file>
               [-sess]
               [-id <key-id>]
               [min_srv <minimum-number-of-servers>]
               [-timeout <number-of-seconds>]
```

Beispiele

In diesem Beispiel wird gezeigt, wie Sie mit `importPubKey` einen öffentlichen Schlüssel in ein HSM importieren.

Example: Importieren eines öffentlichen Schlüssels

Mit diesem Befehl wird ein öffentlicher Schlüssel aus einer Datei mit dem Namen `public.pem` unter der Bezeichnung `importedPublicKey` importiert. Wenn der Befehl erfolgreich ausgeführt wurde, gibt `importPubKey` ein Schlüssel-Handle für den importierten Schlüssel sowie eine Erfolgsmeldung zurück.

```
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
```

Parameters

Dieser Befehl erfordert die folgenden Parameter.

-h

Zeigt die Befehlszeilenhilfe für den Befehl an.

Erforderlich: Ja

-l

Gibt die benutzerdefinierte Bezeichnung für den öffentlichen Schlüssel an.

Erforderlich: Ja

-f

Gibt den Dateinamen des zu importierenden Schlüssels an.

Erforderlich: Ja

-sess

Bezeichnet den importierten Schlüssel als Sitzungsschlüssel.

Standard: Der importierte Schlüssel wird im Cluster als persistenter Schlüssel (Token) bereitgehalten.

Erforderlich: Nein

-id

Gibt die ID des zu importierenden Schlüssels an.

Standard : Kein ID-Wert.

Erforderlich: Nein

-min_srv

Gibt die Mindestanzahl an, mit der der HSMs importierte Schlüssel synchronisiert wird, bevor der Wert des Parameters abläuft. -timeout Falls der Schlüssel nicht in der zulässigen vorgegebenen Zeit mit der angegebenen Anzahl von Servern synchronisiert wird, wird er nicht erstellt.

AWS CloudHSM synchronisiert automatisch jeden Schlüssel mit jedem HSM im Cluster. Um Ihren Prozess zu beschleunigen, setzen Sie den Wert von min_srv auf weniger als die Anzahl von HSMs im Cluster und legen Sie einen niedrigen Timeout-Wert fest. Beachten Sie jedoch, dass einige Anfragen möglicherweise keinen Schlüssel generieren.

Standard: 1

Erforderlich: Nein

-timeout

Gibt an, wie viele Sekunden auf die Synchronisierung des Schlüssels gewartet werden soll, HSMs wenn der min-serv Parameter enthalten ist. Wenn keine Anzahl angegeben ist, wird die Abfrage ohne zeitliche Einschränkung fortgesetzt.

Standard: keine Einschränkung

Erforderlich: Nein

Verwandte Themen

- [exportPubKey](#)
- [Generieren von Schlüsseln](#)

Importieren Sie einen symmetrischen Klartext-Schlüssel mit KMU AWS CloudHSM

Verwenden Sie den imSymKey Befehl im Tool AWS CloudHSM key_mgmt_util, um eine Klartextkopie eines symmetrischen Schlüssels aus einer Datei in das Hardware-Sicherheitsmodul (HSM) zu importieren. Sie können damit Schlüssel importieren, die Sie mit einer beliebigen Methode außerhalb des HSM generiert haben, sowie Schlüssel, die aus einem HSM exportiert wurden, z. B. die Schlüssel, die der Befehl, in eine Datei schreibt. [exSymKey](#)

Während des Importvorgangs verwendet imSymKey einen AES-Schlüssel, den Sie auswählen (Schlüssel zum Packen), um den zu importierenden Schlüssel zu packen (verschlüsseln) und zu

entpacken (entschlüsseln). Allerdings funktioniert `imSymKey` nur für Dateien, die Klartextschlüssel enthalten. Um verschlüsselte Schlüssel zu exportieren und zu importieren, verwenden Sie den [WrapKey](#) und [unWrapKey](#) die Befehle.

Darüber hinaus exportiert der Befehl `imSymKey` nur symmetrische Schlüssel. Zum Importieren von öffentlichen Schlüsseln verwenden Sie [importPubKey](#). Um private Schlüssel zu importieren, verwenden Sie [importPrivateKey](#) oder [WrapKey](#).

Note

Sie können einen passwortgeschützten PEM-Schlüssel nicht mithilfe eines symmetrischen oder privaten Schlüssels importieren.

Importierte Schlüssel können genauso wie im HSM generierte Schlüssel verwendet werden. Der Wert des Attributs [OBJ_ATTR_LOCAL](#) ist allerdings Null, was darauf hinweist, dass es nicht lokal generiert wurde. Mit dem folgenden Befehl können Sie einen symmetrischen Schlüssel freigeben, während Sie ihn importieren. Mit dem Befehl `shareKey` in [cloudhsm_mgmt_util](#) können Sie den Schlüssel freigeben, nachdem er importiert wurde.

```
imSymKey -l aesShared -t 31 -f kms.key -w 3296 -u 5
```

Vergewissern Sie sich, dass Sie die Schlüsseldatei nach dem Importieren des Schlüssels markieren oder entfernen. Dieser Befehl verhindert nicht das wiederholte Importieren der gleichen Schlüsselinformationen. Das Ergebnis, mehrere Schlüssel mit eindeutigen Schlüssel-Handles und den gleichen Schlüsselinformationen, erschwert es, die Verwendung der Schlüsselinformationen nachzuverfolgen und eine Überschreitung der kryptografischen Grenzwerte zu verhindern.

Bevor Sie einen `key_mgmt_util`-Befehl ausführen, müssen Sie [key_mgmt_util starten](#) und sich am HSM als Crypto-Benutzer (CU) [anmelden](#).

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>]

```

Beispiele

Diese Beispiele zeigen, wie Sie symmetrische Schlüssel imSymKey in Ihre importieren können. HSMs

Example: Importieren eines symmetrischen AES-Schlüssels

In diesem Beispiel wird imSymKey ein symmetrischer AES-Schlüssel in den importiert. HSMs

Der erste Befehl verwendet OpenSSL, um einen symmetrischen 256-Bit-AES-Schlüssel nach dem Zufallsprinzip zu generieren. Der Schlüssel wird in der Datei `aes256.key` gespeichert.

```
$ openssl rand -out aes256.key 32
```

Der zweite Befehl dient imSymKey zum Importieren des AES-Schlüssels aus der `aes256.key` Datei in die HSMs. Dabei wird Schlüssel 20, ein AES-Schlüssel im HSM, als Schlüssel zum Packen eingesetzt und die Beschriftung `imported` festgelegt. Im Gegensatz zur ID muss die Beschriftung im Cluster nicht eindeutig sein. Der Wert des Parameters (Typs) `-t` ist 31, der AES darstellt.

Die Ausgabe zeigt, dass der Schlüssel in der Datei verpackt und entpackt und dann in das HSM importiert wurde, wo ihm das Schlüssel-Handle 262180 zugewiesen wurde.

```
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
```

Der nächste Befehl verwendet [getAttribute](#), um das Attribut OBJ_ATTR_LOCAL ([Attribut 355](#)) des neu importierten Schlüssels abzurufen und in der Datei attr_262180 zu speichern.

```
Command: getAttribute -o 262180 -a 355 -out attributes/attr_262180
Attributes dumped into attributes/attr_262180_imported file

Cfm3GetAttribute returned: 0x00 : HSM Return: SUCCESS
```

Wenn Sie die Attributdatei untersuchen, sehen Sie, dass der Wert des Attributs OBJ_ATTR_LOCAL null ist. Dies bedeutet, dass die Schlüsselinformationen nicht im HSM generiert wurden.

```
$ cat attributes/attr_262180_local
OBJ_ATTR_LOCAL
0x00000000
```

Example: Verschieben eines symmetrischen Schlüssels zwischen Clustern

Das folgende Beispiel zeigt, wie Sie mit [exSymKey](#) und `imSymKey` einen AES-Klartextschlüssel zwischen Clustern verschieben. Sie könnten einen Prozess wie diesen verwenden, um ein AES-Wrapping zu erstellen, das auf HSMs beider Clustern vorhanden ist. Sobald der gemeinsame Wrapping-Schlüssel vorhanden ist, können Sie [WrapKey](#) verwenden und [unWrapKey](#) verschlüsselte Schlüssel zwischen den Clustern verschieben.

Der CU-Benutzer, der diesen Vorgang ausführt, muss über die Berechtigung verfügen, sich HSMs auf beiden Clustern anzumelden.

Der erste Befehl verwendet [exSymKey](#) zum Exportieren von Schlüssel 14, einem 32-Bit-AES-Schlüssel, aus Cluster 1 in die Datei `aes.key`. Es verwendet Schlüssel 6, einen AES-Schlüssel auf dem HSMs In-Cluster 1, als Umschließungsschlüssel.

```
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"
```

Der Benutzer meldet sich dann bei `key_mgmt_util` in Cluster 2 an und führt einen `imSymKey` Befehl aus, um den Schlüssel in der `aes.key` Datei in Cluster 2 zu importieren. HSMs Dieser Befehl verwendet den Schlüssel 252152, einen AES-Schlüssel HSMs in Cluster 2, als Umschließungsschlüssel.

Da die Schlüssel zum Packen, die [exSymKey](#) und `imSymKey` verwenden, die Zielschlüssel packen und sofort entpacken, müssen die Schlüssel zum Packen in den verschiedenen Clustern nicht identisch sein.

Die Ausgabe zeigt, dass der Schlüssel erfolgreich in Cluster 2 importiert und das Schlüssel-Handle 21 zugewiesen wurde.

```
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
```

Um nachzuweisen, dass Schlüssel 14 in Cluster 1 und Schlüssel 21 in Cluster 2 über dieselben Schlüsselinformationen verfügen, rufen Sie den Schlüsselprüfwert (Key Check Value, KCV) der einzelnen Schlüssel ab. Bei identischen KCV-Werten sind die Schlüsselinformationen gleich.

Im folgenden Befehl wird [getAttribute](#) in Cluster 1 verwendet, um den Wert des KCV-Attributs (Attribut 371) von Schlüssel 14 in die Datei `attr_14_kcv` zu schreiben. Anschließend wird mit dem Befehl `cat` der Inhalt der Datei `attr_14_kcv` abgerufen.

```
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
```

Dieser ganz ähnliche Befehl verwendet [getAttribute](#) in Cluster 2, um den Wert des KCV-Attributs (Attribut 371) von Schlüssel 21 in die Datei `attr_21_kcv` zu schreiben. Anschließend wird mit dem Befehl `cat` der Inhalt der Datei `attr_21_kcv` abgerufen.

```
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
```

Die Ausgabe zeigt, dass die KCV-Werte der beiden Schlüssel gleich sind, was beweist, dass die Schlüsselinformationen identisch sind.

Da in beiden Clustern dasselbe Schlüsselmateriale vorhanden ist, können Sie jetzt verschlüsselte Schlüssel zwischen den Clustern gemeinsam nutzen, ohne jemals den Klartext-Schlüssel preiszugeben. HSMs Beispielsweise können Sie den Befehl `wrapKey` mit dem Schlüssel 14 zum Packen verwenden, um einen verschlüsselten Schlüssel aus Cluster 1 zu exportieren, und anschließend `unwrapKey` mit Schlüssel 21 zum Packen verwenden, um den verschlüsselten Schlüssel in Cluster 2 zu importieren.

Example: Importieren eines Sitzungsschlüssels

Dieser Befehl verwendet die Parameter `-sess` von `imSymKey` zum Importieren eines 192-Bit-3DES-Schlüssels, der nur für die aktuelle Sitzung gültig ist.

Der Befehl verwendet den Parameter `-f`, um die Datei mit dem zu importierenden Schlüssel anzugeben, den Parameter `-t` zur Angabe des Schlüsseltyps und den Parameter `-w`, um den Schlüssel zum Packen festzulegen. Er verwendet den Parameter `-l` zum Angeben einer Beschriftung, die den Schlüssel kategorisiert, und den Parameter `-id` zum Erstellen eines benutzerfreundlichen, aber eindeutigen Bezeichners für den Schlüssel. Mit dem Parameter `-attest` wird außerdem die Firmware, die den Schlüssel importiert, überprüft.

Die Ausgabe zeigt, dass der Schlüssel erfolgreich verpackt und entpackt und dann in das HSM importiert wurde, wo ihm das Schlüssel-Handle 37 zugewiesen wurde. Außerdem wurde die Bescheinigungsprüfung bestanden, was angibt, dass die Firmware nicht unbefugt geändert wurde.


```
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
```

Anschließend können Sie den Befehl [getAttribute](#) oder [findKey](#) verwenden, um die Attribute der neu importierten Schlüssel zu überprüfen. Der folgende Befehl verwendet `findKey`, um zu überprüfen, dass der Schlüssel 37 über die vom Befehl angegebenen Typ-, Beschriftungs- und ID-Werte verfügt, und dass es sich um einen Sitzungsschlüssel handelt. Wie in Zeile 5 der Ausgabe gezeigt, meldet `findKey`, dass der einzige Schlüssel, der allen Attributen entspricht, Schlüssel 37 ist.

```
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
```

Parameters

-attest

Führt eine Integritätsprüfung durch, die sicherstellt, dass die Firmware, auf der der Cluster läuft, nicht manipuliert wurde.

Standard: Keine Bescheinigungsprüfung.

Erforderlich: Nein

-f

Gibt die Datei an, die den zu importierenden Schlüssel enthält.

Die Datei muss eine Klartextkopie eines AES- oder Triple-DES-Schlüssels der angegebenen Länge enthalten. RC4 und DES-Schlüssel sind im HSMs FIPS-Modus nicht gültig.

- AES: 16, 24 oder 32 Bytes
- Triple-DES (3DES): 24 Byte

Erforderlich: Ja

-h

Zeigt Hilfe für den Befehl an.

Erforderlich: Ja

-id

Gibt einen benutzerdefinierten Bezeichner für den Schlüssel an. Geben Sie eine Zeichenfolge ein, die im Cluster eindeutig ist. Der Standardwert ist eine leere Zeichenfolge.

Standard : Kein ID-Wert.

Erforderlich: Nein

-l

Gibt eine benutzerdefinierte Bezeichnung für den Schlüssel an. Geben Sie eine Zeichenfolge ein.

Sie können eine beliebige Phrase verwenden, die Ihnen bei der Identifizierung des Schlüssels hilft. Da die Bezeichnung nicht eindeutig sein muss, können Sie sie verwenden, um Schlüssel zu gruppieren und zu kategorisieren.

Erforderlich: Ja

-min_srv

Gibt die Mindestanzahl HSMs an, mit der der Schlüssel synchronisiert wird, bevor der Wert des - timeout Parameters abläuft. Falls der Schlüssel nicht in der zulässigen vorgegebenen Zeit mit der angegebenen Anzahl von Servern synchronisiert wird, wird er nicht erstellt.

AWS CloudHSM synchronisiert automatisch jeden Schlüssel mit jedem HSM im Cluster. Um Ihren Prozess zu beschleunigen, setzen Sie den Wert von min_srv auf weniger als die Anzahl von

HSMs im Cluster und legen Sie einen niedrigen Timeout-Wert fest. Beachten Sie jedoch, dass einige Anfragen möglicherweise keinen Schlüssel generieren.

Standard: 1

Erforderlich: Nein

-sess

Erstellt einen Schlüssel, der nur in der aktuellen Sitzung existiert. Der Schlüssel kann nach Ende der Sitzung nicht wiederhergestellt werden.

Verwenden Sie diesen Parameter, wenn Sie einen Schlüssel zum Packen nur für kurze Zeit benötigen, z. B. einen Schlüssel, der einen anderen Schlüssel verschlüsselt und dann schnell entschlüsselt. Verwenden Sie keinen Sitzungsschlüssel, um Daten zu verschlüsseln, die Sie nach dem Ende der Sitzung möglicherweise entschlüsseln müssen.

Um einen Sitzungsschlüssel in einen persistenten (Token-)Schlüssel zu ändern, verwenden Sie [setAttribute](#).

Standard: Der Schlüssel ist persistent.

Erforderlich: Nein

-timeout

Gibt an, wie lange (in Sekunden) der Befehl darauf wartet, dass eine Taste mit der im Parameter HSMS angegebenen Anzahl von synchronisiert wird. `min_srv`

Dieser Parameter ist nur gültig, wenn der `min_srv`-Parameter auch im Befehl verwendet wird.

Voreinstellung: Keine Zeitüberschreitung. Der Befehl wartet auf unbestimmte Zeit und kehrt erst zurück, wenn der Schlüssel mit der Mindestanzahl von Servern synchronisiert ist.

Erforderlich: Nein

-t

Gibt den Typ des symmetrischen Schlüssels an. Geben Sie die Konstante ein, die den Schlüsseltyp darstellt. Zum Erstellen eines AES-Schlüssels geben Sie beispielsweise `-t 31` ein.

Zulässige Werte:

- 21: [Triple DES \(3DES\)](#).
- 31: [AES](#)

Erforderlich: Ja

-u

Gibt den zu importierenden Schlüssel für die angegebenen Benutzer frei. Dieser Parameter gibt anderen HSM-Kryptobenzern (CUs) die Erlaubnis, diesen Schlüssel für kryptografische Operationen zu verwenden.

Geben Sie eine ID oder eine durch Kommas getrennte Liste von HSM-Benutzern IDs ein, z. B. -u. 5,6 Fügen Sie die HSM-Benutzer-ID des aktuellen Benutzers nicht ein. Zum Ermitteln einer ID können Sie den Befehl [listUsers](#) im Befehlszeilen-Tool `cloudhsm_mgmt_util` oder den Befehl [listUsers](#) im Befehlszeilen-Tool `key_mgmt_util` verwenden.

Erforderlich: Nein

-w

Gibt das Schlüssel-Handle des Verpackungsschlüssels an. Dieser Parameter muss angegeben werden. Verwenden Sie den [findKey](#)-Befehl, um Schlüssel-Handles zu finden.

Ein Schlüssel zum Packen ist ein Schlüssel im HSM, der zum Verschlüsseln („Verpacken“) und zum Entschlüsseln („Entpacken“) des Schlüssels während des Importvorgangs dient. Nur AES-Schlüssel können als Verpackungsschlüssel verwendet werden.

Sie können jeden AES-Schlüssel (in jeder beliebigen Größe) als Verpackungsschlüssel verwenden. Da der Verpackungsschlüssel den Zielschlüssel verpackt und danach sofort entpackt, können Sie einen nur für die aktuelle Sitzung gültigen AES-Schlüssel als Verpackungsschlüssel verwenden. Um zu bestimmen, ob ein Schlüssel als Schlüssel zum Packen verwendet werden kann, führen Sie [getAttribute](#) aus, um den Wert des OBJ_ATTR_WRAP-Attributs (262) abzurufen. Um einen Wrapping-Schlüssel zu erstellen, verwenden Sie [genSymKey](#), um einen AES-Schlüssel (Typ 31) zu generieren.

Wenn Sie den Parameter `-wk` zum Angeben eines externen Schlüssels zum Packen verwenden, wird der `-w`-Schlüssel zum Packen zwar zum Entpacken, jedoch nicht zum Verpacken des Schlüssels während des Importvorgangs verwendet.

 Note

Schlüssel 4 ist ein nicht unterstützter, interner Schlüssel. Wir empfehlen, dass Sie einen AES-Schlüssel als Umhüllungsschlüssel verwenden, den Sie erstellen und verwalten.

Erforderlich: Ja

-wk

Verwenden Sie den AES-Schlüssel in der angegebenen Datei, um den Schlüssel, der importiert wird, zu verpacken. Geben Sie den Pfad und den Namen einer Datei an, die einen Klartext-AES-Schlüssel enthält.

Wenn Sie diesen Parameter angeben, verwendet `imSymKey` den Schlüssel in der Datei `-wk`, um den zu importierenden Schlüssel zu verpacken, und den Schlüssel im HSM, der durch den `-w`-Parameter festgelegt ist, um ihn zu entpacken. Die Parameterwerte `-w` und `-wk` müssen durch denselben Klartextschlüssel aufgelöst werden.

Standard: Verwenden Sie den Verpackungsschlüssel auf dem HSM zum Entpacken.

Erforderlich: Nein

Verwandte Themen

- [genSymKey](#)
- [exSymKey](#)
- [wrapKey](#)
- [unWrapKey](#)
- [exportPrivateKey](#)
- [exportPubKey](#)

Fügen Sie ein maskiertes Objekt mit AWS CloudHSM KMU ein

Verwenden Sie den `insertMaskedObject` Befehl in AWS CloudHSM `key_mgmt_util`, um ein maskiertes Objekt aus einer Datei in ein designiertes Hardware-Sicherheitsmodul (HSM) einzufügen. Maskierte Objekte sind geklonte Objekte, die mit dem Befehl [extractMaskedObject](#) aus einem HSM extrahiert werden. Sie können nur verwendet werden, nachdem sie wieder in den ursprünglichen Cluster eingefügt wurden. Sie können ein maskiertes Objekt nur in den Cluster einfügen, aus dem es erstellt wurde (oder in eine geklonte Version dieses Clusters). Dies umfasst alle geklonten Versionen des ursprünglichen Clusters, die durch eine [regionsübergreifende Sicherungskopie](#) und die anschließende [Verwendung dieser Sicherung zum Erstellen eines neuen Clusters](#) generiert wurden.

Maskierte Objekte sind eine effiziente Möglichkeit zum Auslagern und Synchronisieren von Schlüsseln, auch von nicht extrahierbaren Schlüsseln (d. h. Schlüssel mit einem

[OBJ_ATTR_EXTRACTABLE](#)-Wert von 0). [Auf diese Weise können Schlüssel sicher zwischen verwandten Clustern in verschiedenen Regionen synchronisiert werden, ohne dass die Konfigurationsdatei aktualisiert werden muss.](#) [AWS CloudHSM](#)

Bevor Sie einen `key_mgmt_util`-Befehl ausführen, müssen Sie [key_mgmt_util starten](#) und sich am HSM als Crypto-Benutzer (CU) [anmelden](#).

Syntax

```
insertMaskedObject -h

insertMaskedObject -f <filename>
                    [-min_srv <minimum-number-of-servers>]
                    [-timeout <number-of-seconds>]
```

Beispiele

In diesem Beispiel wird gezeigt, wie Sie mit `insertMaskedObject` ein maskiertes Objekt in ein HSM einfügen.

Example: Einfügen eines maskierten Objekts

Dieser Befehl fügt ein maskiertes Objekt aus einer Datei mit dem Namen `maskedObj` in ein HSM ein. Wenn der Befehl erfolgreich ausgeführt wurde, gibt `insertMaskedObject` ein Key-Handle für den aus dem maskierten Objekt entschlüsselten Schlüssel sowie eine Erfolgsmeldung zurück.

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
```

Parameters

Dieser Befehl erfordert die folgenden Parameter.

-h

Zeigt die Befehlszeilenhilfe für den Befehl an.

Erforderlich: Ja

-f

Gibt den Dateinamen des einzufügenden maskierten Objekts an.

Erforderlich: Ja

-min_srv

Gibt die Mindestanzahl der Server an, auf denen das eingefügte maskierte Objekt synchronisiert wird, bevor der Wert des `-timeout`-Parameters verfällt. Falls das Objekt nicht in der vorgegebenen Zeit mit der angegebenen Anzahl von Servern synchronisiert ist, wird es nicht eingefügt.

Standard: 1

Erforderlich: Nein

-timeout

Gibt an, wie viele Sekunden bei Einschluss des Parameters `min-serv` gewartet wird, bis der Schlüssel serverübergreifend synchronisiert ist. Wenn keine Anzahl angegeben ist, wird die Abfrage ohne zeitliche Einschränkung fortgesetzt.

Standard: keine Einschränkung

Erforderlich: Nein

Verwandte Themen

- [extractMaskedObject](#)
- [syncKey](#)
- [Regionsübergreifendes Kopieren einer Sicherung](#)
- [Einen AWS CloudHSM Cluster aus einem vorherigen Backup erstellen](#)

Schlüsseldatei mit AWS CloudHSM KMU validieren

Verwenden Sie den `IsValidKeyHandlefile` Befehl in AWS CloudHSM `key_mgmt_util`, um herauszufinden, ob eine Schlüsseldatei einen echten privaten Schlüssel oder einen gefälschten RSA-PEM-Schlüssel enthält. Eine gefälschte PEM-Datei enthält nicht das Material des tatsächlichen privaten Schlüssels, sondern verweist auf den privaten Schlüssel im HSM. Eine solche Datei

kann verwendet werden, um das Offloading von Ihrem Webserver zu einzurichten. SSL/TLS AWS CloudHSM Weitere Informationen finden Sie unter [SSL/TLS-Offload unter Linux mit Tomcat oder SSL/TLS-Offload unter Linux mit NGINX](#) oder Apache.

 Note

IsValidKeyHandlefile funktioniert nur für RSA-Schlüssel.

Bevor Sie einen key_mgmt_util-Befehl ausführen, müssen Sie [key_mgmt_util starten](#) und sich am HSM als Crypto-Benutzer (CU) [anmelden](#).

Syntax

```
IsValidKeyHandlefile -h  
  
IsValidKeyHandlefile -f <rsa-private-key-file>
```

Beispiele

Diese Beispiele verdeutlichen, wie mit IsValidKeyHandlefile ermittelt werden kann, ob eine bestimmte Schlüsseldatei das Material des echten Schlüssels oder Material eines gefälschten PEM-Schlüssels enthält.

Example: Validieren eines echten privaten Schlüssels

Mit diesem Befehl wird bestätigt, dass die Datei namens privateKey.pem Material des echten Schlüssels enthält.

```
Command: IsValidKeyHandlefile -f privateKey.pem
```

```
Input key file has real private key
```

Example: Entwerten eines gefälschten PEM-Schlüssels

Mit diesem Befehl wird bestätigt, dass die Datei namens caviumKey.pem Material eines gefälschten PEM-Schlüssels enthält, das aus dem Schlüssel-Handle 15 erstellt wurde.

```
Command: IsValidKeyHandlefile -f caviumKey.pem
```


Input file has invalid key handle: 15

Parameters

Dieser Befehl erfordert die folgenden Parameter.

-h

Zeigt die Befehlszeilenhilfe für den Befehl an.

Erforderlich: Ja

-f

Gibt die Datei mit dem privaten RSA-Schlüssel an, die auf gültiges Schlüsselmateriale überprüft werden soll.

Erforderlich: Ja

Verwandte Themen

- [getCaviumPrivKey](#) (Schlüssel)
- [SSL/TLS-Offload unter Linux mit Tomcat](#)
- [SSL/TLS-Offload unter Linux mit NGINX oder Apache](#)

Die Attribute eines AWS CloudHSM Schlüssels mithilfe von KMU auflisten

Verwenden Sie den listAttributes Befehl im AWS CloudHSM key_mgmt_util, um die Attribute eines Schlüssels und die Konstanten, die sie repräsentieren, aufzulisten. AWS CloudHSM Sie nutzen die Konstanten zum Identifizieren der Attribute in den Befehlen [getAttribute](#) und [setAttribute](#). Hilfe zur Interpretation der Schlüsselattribute finden Sie unter [AWS CloudHSM Referenz zu Schlüsselattributen für KMU](#).

Bevor Sie einen key_mgmt_util-Befehl ausführen, müssen Sie [key_mgmt_util starten](#) und sich am HSM als Crypto-Benutzer (CU) [anmelden](#).

Syntax

Dieser Befehl hat keine Parameter.

`listAttributes`

Beispiel

Mit diesem Befehl werden die Schlüsselattribute aufgelistet, die Sie mit `key_mgmt_util` abrufen und ändern können, sowie die Konstanten, die sie darstellen. Hilfe zur Interpretation der Schlüsselattribute finden Sie unter [AWS CloudHSM Referenz zu Schlüsselattributen für KMU](#).

Um alle Attribute im Befehl [getAttribute](#) in `key_mgmt_util` darzustellen, verwenden Sie 512.

Command: **listAttributes**

Following are the possible attribute values for `getAttribute`:

<code>OBJ_ATTR_CLASS</code>	= 0
<code>OBJ_ATTR_TOKEN</code>	= 1
<code>OBJ_ATTR_PRIVATE</code>	= 2
<code>OBJ_ATTR_LABEL</code>	= 3
<code>OBJ_ATTR_KEY_TYPE</code>	= 256
<code>OBJ_ATTR_ENCRYPT</code>	= 260
<code>OBJ_ATTR_DECRYPT</code>	= 261
<code>OBJ_ATTR_WRAP</code>	= 262
<code>OBJ_ATTR_UNWRAP</code>	= 263
<code>OBJ_ATTR_SIGN</code>	= 264
<code>OBJ_ATTR_VERIFY</code>	= 266
<code>OBJ_ATTR_LOCAL</code>	= 355
<code>OBJ_ATTR_MODULUS</code>	= 288
<code>OBJ_ATTR_MODULUS_BITS</code>	= 289
<code>OBJ_ATTR_PUBLIC_EXPONENT</code>	= 290
<code>OBJ_ATTR_VALUE_LEN</code>	= 353
<code>OBJ_ATTR_EXTRACTABLE</code>	= 354
<code>OBJ_ATTR_KCV</code>	= 371

Verwandte Themen

- [listAttributes](#) in `cloudhsm_mgmt_util`
- [getAttribute](#)
- [setAttribute](#)
- [Schlüsselattributreferenz](#)

Alle AWS CloudHSM Benutzer auflisten, die KMU verwenden

Verwenden Sie den `listUsers` Befehl in der Datei AWS CloudHSM `key_mgmt_util`, um die Benutzer in den Hardware-Sicherheitsmodulen (HSM) zusammen mit ihrem Benutzertyp und anderen Attributen abzurufen.

In `key_mgmt_util` gibt `ListUsers` eine Ausgabe zurück, die alles HSMs im Cluster darstellt, auch wenn sie nicht konsistent sind. Verwenden Sie zum Abrufen von Informationen über die Benutzer in jedem HSM den Befehl [listUsers](#) in `cloudhsm_mgmt_util`.

Die Benutzerbefehle in `key_mgmt_util` `listUsers` und, sind schreibgeschützte Befehle, zu deren Ausführung Crypto-Benutzer () berechtigt [getKeyInfos](#) sind. CUs Die restlichen Benutzerverwaltungsbefehle sind Teil von `cloudhsm_mgmt_util`. Sie werden von Verschlüsselungsverantwortlichen (CO) mit Benutzerverwaltungsberechtigungen ausgeführt.

Bevor Sie einen `key_mgmt_util`-Befehl ausführen, müssen Sie [key_mgmt_util starten](#) und sich am HSM als Crypto-Benutzer (CU) [anmelden](#).

Syntax

```
listUsers
```

```
listUsers -h
```

Beispiel

Dieser Befehl listet die Benutzer im Cluster und ihre Attribute auf. HSMs Sie können das `User ID` Attribut verwenden, um Benutzer in anderen Befehlen wie [findKey](#), [GetAttribute](#) und zu identifizieren. [getKeyInfo](#)

Command: `listUsers`

Number Of Users found 4

Index	User ID	User Type	User Name	MofnPubKey
0	1	PCO	admin	NO
0	2	AU	app_user	NO
0	3	CU	alice	YES

0	4	NO	4	CU	bob	NO
0	5	NO	5	CU	trent	YES

Cfm3ListUsers returned: 0x00 : HSM Return: SUCCESS

Die Ausgabe umfasst die folgenden Benutzerattribute:

- Benutzer-ID: Identifiziert den Benutzer in den Befehlen `key_mgmt_util` und [cloudhsm_mgmt_util](#).
- [User type](#): Bestimmt die Operationen, die der Benutzer im HSM ausführen kann.
- User Name: Zeigt den benutzerdefinierten Anzeigenamen für den Benutzer an.
- MofnPubKey: Gibt an, ob der Benutzer ein key pair zum Signieren von [Quorum-Authentifizierungstoken](#) registriert hat.
- LoginFailureCnt: Gibt an, wie oft sich der Benutzer erfolglos angemeldet hat.
- 2FA: Gibt an, dass der Benutzer Multifaktor-Authentifizierung aktiviert hat.

Parameters

-h

Zeigt Hilfe für den Befehl an.

Erforderlich: Ja

Verwandte Themen

- [listUsers](#) in `cloudhsm_mgmt_util`
- [findKey](#)
- [getAttribute](#)
- [getKeyInfo](#)

Mit AWS CloudHSM KMU bei einem HSM an- und abmelden

Verwenden Sie die `logoutHSM` Befehle `loginHSM` und in der Datei AWS CloudHSM `key_mgmt_util`, um sich bei den Hardware-Sicherheitsmodulen (HSM) in einem Cluster an- und abzumelden. Sobald Sie bei dem angemeldet sind HSMs, können Sie `key_mgmt_util` verwenden, um eine Vielzahl von

Schlüsselverwaltungsvorgängen durchzuführen, einschließlich Generierung, Synchronisation und Wrapping von öffentlichen und privaten Schlüsseln.

Bevor Sie den Befehl `key_mgmt_util` ausführen, müssen Sie [key_mgmt_util starten](#). [Um Schlüssel mit key_mgmt_util zu verwalten, müssen Sie sich als Crypto-Benutzer \(CU\) anmelden. HSMs](#)

Note

Wenn Sie mehr als fünf falsche Anmeldeversuche tätigen, wird Ihr Konto gesperrt.

Wenn Sie Ihren Cluster vor Februar 2018 erstellt haben, wird Ihr Konto nach 20 falschen Anmeldeversuchen gesperrt. Ein Crypto Officer (CO) muss Ihr Passwort mit dem Befehl [changePswd](#) in `cloudhsm_mgmt_util` zurücksetzen, um das Konto zu entsperren.

Wenn Sie mehr als ein HSM in Ihrem Cluster haben, sind Ihnen möglicherweise weitere falsche Anmeldeversuche gestattet, bevor Ihr Konto gesperrt wird. Dies liegt daran, dass der CloudHSM-Client die Last auf verschiedene verteilt. HSMs Aus diesem Grund beginnt der Anmeldeversuch möglicherweise nicht jedes Mal auf demselben HSM. Wenn Sie diese Funktion testen, empfehlen wir Ihnen, dies auf einem Cluster mit nur einem aktiven HSM zu tun.

Syntax

```
loginHSM -h

loginHSM -u <user type>
          { -p | -hpswd } <password>
          -s <username>
```

Beispiel

Dieses Beispiel zeigt, wie Sie sich mit den Befehlen und HSMs in einem Cluster an `loginHSM` - und `logoutHSM` abmelden.

Example: Melden Sie sich bei der an HSMs

Dieser Befehl meldet Sie HSMs als Krypto-Benutzer (CU) mit dem Benutzernamen `example_user` und dem Passwort `anaws`. Die Ausgabe zeigt, dass Sie sich bei allen HSMs im Cluster angemeldet haben.

```
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: Melden Sie sich mit einem versteckten Passwort an

Dieser Befehl entspricht dem obigen Beispiel, außer dass Sie dieses Mal angeben, dass das System das Passwort verbergen soll.

```
Command: loginHSM -u CU -s example_user -hpswd
```

Sie werden vom System aufgefordert, Ihr Passwort einzugeben. Sie geben das Passwort ein, das System versteckt das Passwort und die Ausgabe zeigt, dass der Befehl erfolgreich war und dass Sie sich mit dem HSMs verbunden haben.

```
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: Loggen Sie sich aus dem ab HSMs

Mit diesem Befehl melden Sie sich von der ab HSMs. Die Ausgabe zeigt, dass Sie sich von allen HSMs im Cluster abgemeldet haben.

```
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
```

Parameters

-h

Zeigt Hilfe für diesen Befehl an.

-u

Gibt den Anmeldebenutzertyp an. Zum Verwenden von `key_mgmt_util` müssen Sie als CU angemeldet sein.

Erforderlich: Ja

-s

Gibt den Anmeldebenutzernamen an.

Erforderlich: Ja

{ -p | -hpswd }

Geben Sie das Login-Passwort mit `-p` an. Das Passwort wird in Klartext angezeigt, wenn Sie es eingeben. Um Ihr Passwort zu verbergen, verwenden Sie den optionalen `-hpswd`-Parameter anstelle von `-p` und folgen Sie der Aufforderung.

Erforderlich: Ja

Verwandte Themen

- [exit](#)

Legen Sie die Attribute von AWS CloudHSM Schlüsseln mithilfe von KMU fest

Verwenden Sie den `setAttribute` Befehl in AWS CloudHSM `key_mgmt_util`, um einen Schlüssel, der nur in der aktuellen Sitzung gültig ist, in einen dauerhaften Schlüssel zu konvertieren, der solange existiert, bis Sie ihn löschen. Dazu wird der Wert des Token-Attributs des Schlüssels (`OBJ_ATTR_TOKEN`) von „false“ (0) in „true“ (1) geändert. Sie können nur die Attribute der Schlüssel ändern, deren Eigentümer Sie sind.

Sie können auch den `setAttribute`-Befehl in `cloudhsm_mgmt_util` verwenden, um die Attribute `label`, `wrap`, `unwrap`, `encrypt` und `decrypt` zu ändern.

Bevor Sie einen `key_mgmt_util`-Befehl ausführen, müssen Sie [key_mgmt_util starten](#) und sich am HSM als Crypto-Benutzer (CU) [anmelden](#).

Syntax

```
setAttribute -h

setAttribute -o <object handle>
               -a 1
```

Beispiel

In diesem Beispiel wird gezeigt, wie Sie einen Sitzungsschlüssel in einen persistenten Schlüssel konvertieren.

Der erste Befehl verwendet den `-sess` Parameter von, um einen 192-Bit-AES-Schlüssel [genSymKey](#) zu erstellen, der nur in der aktuellen Sitzung gültig ist. Die Ausgabe zeigt, dass das Schlüssel-Handle des neuen Sitzungsschlüssels 262154 lautet.

```
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
```

Dieser Befehl verwendet [findKey](#), um die Sitzungsschlüssel in der aktuellen Sitzung zu suchen. Die Ausgabe verifiziert, dass der Schlüssel 262154 ein Sitzungsschlüssel ist.

```
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
```

Dieser Befehl verwendet `setAttribute`, um den Schlüssel 262154 von einem Sitzungsschlüssel in einen persistenten Schlüssel zu konvertieren. Dazu wird der Wert des Token-Attributs (`OBJ_ATTR_TOKEN`) des Schlüssels von 0 („false“) in 1 („true“) geändert. Hilfe zur Interpretation der Schlüsselattribute finden Sie unter [AWS CloudHSM Referenz zu Schlüsselattributen für KMU](#).

Der Befehl verwendet den Parameter `-o` zum Angeben des Schlüssel-Handles (262154) und den Parameter `-a` zum Festlegen der Konstanten, die das Token-Attribut (1) darstellt. Wenn Sie den Befehl ausführen, werden Sie zur Eingabe eines Werts für das Token-Attribut aufgefordert. Der einzige gültige Wert ist 1 (true); der Wert für einen persistenten Schlüssel.

```
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
```

Um sich zu vergewissern, dass der Schlüssel 262154 nun persistent ist, verwendet dieser Befehl `findKey` für die Suche nach Sitzungsschlüsseln (`-sess 1`) und persistenten Schlüsseln (`-sess 0`). In diesem Fall findet der Befehl zwar keine Sitzungsschlüssel, gibt jedoch 262154 in der Liste der persistenten Schlüssel zurück.

```
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
```

Parameters

-h

Zeigt Hilfe für den Befehl an.

Erforderlich: Ja

-o

Gibt das Schlüssel-Handle des Zielschlüssels an. Sie können nur jeweils einen Schlüssel in den einzelnen Befehlen angeben. Verwenden Sie zum Abrufen des Schlüssel-Handles eines Schlüssels [findKey](#).

Erforderlich: Ja

-a

Gibt die Konstante an, die das Attribut darstellt, das Sie ändern möchten. Der einzige gültige Wert ist 1, der das Token-Attribut OBJ_ATTR_TOKEN darstellt.

Verwenden Sie [listAttributes](#), um die Attribute und deren Ganzzahlwerte abzurufen.

Erforderlich: Ja

Verwandte Themen

- [setAttribute](#) in cloudhsm_mgmt_util
- [getAttribute](#)
- [listAttributes](#)

- [Schlüsselattributreferenz](#)

Generieren Sie eine Signatur mit AWS CloudHSM KMU

Verwenden Sie den `sign` Befehl in AWS CloudHSM `key_mgmt_util`, um einen ausgewählten privaten Schlüssel zu verwenden, um eine Signatur für eine Datei zu generieren.

Um `sign` verwenden zu können, müssen Sie zunächst über einen privaten Schlüssel in Ihrem HSM verfügen. Sie können einen privaten Schlüssel mit dem Befehl [genSymKey](#), [genRSAKeyPair](#) oder [genECCKeypair](#) erstellen. Sie können einen privaten Schlüssel auch mit dem Befehl [importPrivateKey](#) importieren. Weitere Informationen finden Sie unter [Generieren von Schlüsseln](#).

Der Befehl `sign` verwendet einen benutzerdefinierten Signierungsmechanismus (dargestellt durch eine Ganzzahl) zum Signieren einer Nachricht. Eine Liste mit möglichen Signierungsmechanismen finden Sie unter [Parameter](#).

Bevor Sie einen `key_mgmt_util`-Befehl ausführen, müssen Sie [key_mgmt_util starten](#) und sich am HSM als Crypto-Benutzer (CU) [anmelden](#).

Syntax

```
sign -h

sign -f <file name>
    -k <private key handle>
    -m <signature mechanism>
    -out <signed file name>
```

Beispiel

In diesem Beispiel wird gezeigt, wie Sie den Befehl `sign` verwenden, um eine Datei zu signieren.

Example: Signieren einer Datei

Dieser Befehl signiert eine Datei mit dem Namen `messageFile` anhand eines privaten Schlüssels mit dem Handle `266309`. Er verwendet den `SHA256_RSA_PKCS (1)`-Signierungsmechanismus und speichert die signierte Datei als `signedFile`.

```
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
```

Parameters

Dieser Befehl erfordert die folgenden Parameter.

-f

Der Name der zu signierenden Datei.

Erforderlich: Ja

-k

Das Handle des privaten Schlüssels, der zum Signieren verwendet werden soll.

Erforderlich: Ja

-m

Eine Ganzzahl, die den zum Signieren verwendeten Signierungsmechanismus repräsentiert. Die möglichen Mechanismen entsprechen den folgenden Ganzzahlen:

Signierungsmechanismus	Entsprechende Ganzzahl
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

Signierungsmechanismus	Entsprechende Ganzzahl
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

Erforderlich: Ja

-out

Gibt den Namen der Datei an, in die die signierte Datei gespeichert wird.

Erforderlich: Ja

Verwandte Themen

- [verify](#)
- [importPrivateKey](#)
- [RSAKeyGen-Paar](#)
- [ECCKeyGen-Paar](#)
- [genSymKey](#)
- [Generieren von Schlüsseln](#)

Entpacken Sie einen AWS CloudHSM Schlüssel mit KMU

Verwenden Sie den `unWrapKey` Befehl im Tool AWS CloudHSM `key_mgmt_util`, um einen verpackten (verschlüsselten) symmetrischen oder privaten Schlüssel aus einer Datei in das HSM

zu importieren. Wurde entwickelt, um verschlüsselte Schlüssel zu importieren, die mit dem Befehl [wrapKey](#) in key_mgmt_util verpackt wurden. Kann aber auch verwendet werden, um Schlüssel zu entpacken, die mit anderen Tools verpackt wurden. In diesen Situationen empfehlen wir jedoch, die Softwarebibliotheken [PKCS#11](#) oder [JCE](#) zu verwenden, um den Schlüssel zu entpacken.

Importierte Schlüssel funktionieren wie Schlüssel, die von generiert wurden. AWS CloudHSM Der Wert ihres [Attributs OBJ_ATTR_LOCAL](#) ist jedoch Null, was bedeutet, dass sie nicht lokal generiert wurden.

Stellen Sie nach dem Importieren eines Schlüssels sicher, dass Sie die Schlüsseldatei markieren oder löschen. Dieser Befehl verhindert nicht das wiederholte Importieren der gleichen Schlüsselinformationen. Das Ergebnis – mehrere Schlüssel mit unterschiedlichen Schlüsselhandles und demselben Schlüsselmaterial – macht es schwierig, die Verwendung des Schlüsselmaterials zu verfolgen und zu verhindern, dass die kryptografischen Grenzen überschritten werden.

Bevor Sie einen key_mgmt_util-Befehl ausführen, müssen Sie [key_mgmt_util starten](#) und sich am HSM als Crypto-Benutzer (CU) [anmelden](#).

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>]
```

Beispiel

Diese Beispiele zeigen, wie Sie `unWrapKey` einen umschlossenen Schlüssel aus einer Datei in die importieren können HSMs. Im ersten Beispiel entpacken wir einen Schlüssel, der mit dem Befehl [wrapKey](#) `key_mgmt_util` verpackt wurde und daher einen Header hat. Im zweiten Beispiel entpacken wir einen Schlüssel, der außerhalb von `key_mgmt_util` verpackt wurde und daher keinen Header hat.

Example: Einen Schlüssel entpacken (mit Header)

Dieser Befehl importiert eine verpackte Kopie eines symmetrischen 3DES-Schlüssels in ein HSM. Der Schlüssel wird mit einem AES-Schlüssel mit dem Bezeichner 6 entpackt, der mit demjenigen kryptographisch identisch ist, der zum Verpacken des 3DES-Schlüssels verwendet wurde. Die Ausgabe zeigt, dass der Schlüssel in der Datei entpackt und importiert wurde und dass das Handle des importierten Schlüssels 29 ist.

```
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: Einen Schlüssel entpacken (kein Header)

Dieser Befehl importiert eine verpackte Kopie eines symmetrischen 3DES-Schlüssels in ein HSM. Der Schlüssel wird mit einem AES-Schlüssel mit dem Bezeichner 6 entpackt, der mit demjenigen kryptographisch identisch ist, der zum Verpacken des 3DES-Schlüssels verwendet wurde. Da dieser 3DES-Schlüssel nicht mit `key_mgmt_util` verpackt wurde, wird der `noheader`-Parameter zusammen mit den erforderlichen Begleitparametern angegeben: ein Schlüssellabel (`unwrapped3DES`), eine Schlüsselklasse (4) und ein Schlüsseltyp (21). Die Ausgabe zeigt, dass der Schlüssel in der Datei entpackt und importiert wurde und dass das Handle des importierten Schlüssels 8 ist.

```
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
```

Parameters

-h

Zeigt Hilfe für den Befehl an.

Erforderlich: Ja

-f

Gibt den Pfad und Namen der Datei an, die den verpackten Schlüssel enthält.

Erforderlich: Ja

-w

Gibt den Wrapping-Schlüssel an. Geben Sie das Schlüssel-Handle eines AES-Schlüssels oder RSA-Schlüssels auf dem HSM ein. Dieser Parameter muss angegeben werden. Verwenden Sie den [findKey](#)-Befehl, um Schlüssel-Handles zu finden.

Um einen Wrapping-Schlüssel zu erstellen, verwenden Sie, [genSymKey](#)um einen AES-Schlüssel (Typ 31) oder ein [RSAKeyGen-Paar](#) zu generieren, um ein RSA-Schlüsselpaar (Typ 0) zu generieren. Wenn Sie ein RSA-Schlüsselpaar verwenden, achten Sie darauf, den Schlüssel mit einem der Schlüssel zu umwickeln und ihn mit dem anderen zu entpacken. Verwenden Sie [getAttribute](#), um zu überprüfen, ob ein Schlüssel als Verpackungsschlüssel verwendet werden kann, und um den Wert des OBJ_ATTR_WRAP-Attributs abzurufen, der von der Konstanten 262 dargestellt wird.

Erforderlich: Ja

-sess

Erstellt einen Schlüssel, der nur in der aktuellen Sitzung existiert. Der Schlüssel kann nach Ende der Sitzung nicht wiederhergestellt werden.

Verwenden Sie diesen Parameter, wenn Sie einen Schlüssel zum Packen nur für kurze Zeit benötigen, z. B. einen Schlüssel, der einen anderen Schlüssel verschlüsselt und dann schnell entschlüsselt. Verwenden Sie keinen Sitzungsschlüssel, um Daten zu verschlüsseln, die Sie nach dem Ende der Sitzung möglicherweise entschlüsseln müssen.

Um einen Sitzungsschlüssel in einen persistenten (Token-)Schlüssel zu ändern, verwenden Sie [setAttribute](#).

Standard: Der Schlüssel ist persistent.

Erforderlich: Nein

`-min_srv`

Gibt die Mindestanzahl HSMs an, mit der der Schlüssel synchronisiert wird, bevor der Wert des `-timeout` Parameters abläuft. Falls der Schlüssel nicht in der zulässigen vorgegebenen Zeit mit der angegebenen Anzahl von Servern synchronisiert wird, wird er nicht erstellt.

AWS CloudHSM synchronisiert automatisch jeden Schlüssel mit jedem HSM im Cluster. Um Ihren Prozess zu beschleunigen, setzen Sie den Wert von `min_srv` auf weniger als die Anzahl von HSMs im Cluster und legen Sie einen niedrigen Timeout-Wert fest. Beachten Sie jedoch, dass einige Anfragen möglicherweise keinen Schlüssel generieren.

Standard: 1

Erforderlich: Nein

`-timeout`

Gibt an, wie lange (in Sekunden) der Befehl darauf wartet, dass eine Taste mit der im Parameter HSMs angegebenen Anzahl von synchronisiert wird. `min_srv`

Dieser Parameter ist nur gültig, wenn der `min_srv`-Parameter auch im Befehl verwendet wird.

Voreinstellung: Keine Zeitüberschreitung. Der Befehl wartet auf unbestimmte Zeit und kehrt erst zurück, wenn der Schlüssel mit der Mindestanzahl von Servern synchronisiert ist.

Erforderlich: Nein

`-attest`

Führt eine Integritätsprüfung durch, die sicherstellt, dass die Firmware, auf der der Cluster läuft, nicht manipuliert wurde.

Standard: Keine Bescheinigungsprüfung.

Erforderlich: Nein

-nex

Macht den Schlüssel nicht extrahierbar. Der generierte Schlüssel kann nicht [aus dem HSM exportiert](#) werden.

Standard: Der Schlüssel ist extrahierbar.

Erforderlich: Nein

-m

Der Wert, der den Verschlüsselungsmechanismus bezeichnet. CloudHSM unterstützt die folgenden Mechanismen:

Mechanismus	Wert
AES_KEY_WRAP_PAD_PKCS5	4
NIST_AES_WRAP_NO_PAD	5
NIST_AES_WRAP_PAD	6
RSA_AES	7
RSA_OAEP (zur maximalen Datengröße lesen Sie den Hinweis weiter unten in diesem Abschnitt)	8
AES_GCM	10
CLOUDHSM_AES_GCM	11
RSA_PKCS (Zur maximalen Datengröße lesen Sie den Hinweis weiter unten in diesem Abschnitt.) Eine bevorstehende Änderung finden Sie im Hinweis 1 unten.	12

Erforderlich: Ja

Note

Wenn Sie den RSA_OAEP Umbruchmechanismus verwenden, wird die maximale Schlüsselgröße, die Sie umbrechen können, durch den Modul des RSA-Schlüssels und die Länge des angegebenen Hashs wie folgt bestimmt: Maximale Schlüsselgröße = $\text{modulusLengthIn Bytes} - (2 * \text{hashLengthIn Byte}) - 2$.

Wenn Sie den RSA_PKCS-Wrapping-Mechanismus verwenden, wird die maximale Schlüsselgröße, die Sie umbrechen können, durch das Modul des RSA-Schlüssels wie folgt bestimmt: Maximale Schlüsselgröße = $(\text{Byte} - 11) \cdot \text{modulusLengthIn}$

-t

Hash-Algorithmus	Wert
SHA1	2
SHA256	3
SHA384	4
SHA512	5
SHA224 (gültig für RSA_AES- und RSA_OAEP-Mechanismen)	6

Erforderlich: Nein

-noheader

Wenn Sie einen Schlüssel entpacken, der außerhalb von `key_mgmt_util` verpackt wurde, müssen Sie diesen Parameter und alle anderen zugehörigen Parameter angeben.

Erforderlich: Nein

Note

Wenn Sie diesen Parameter angeben, müssen Sie auch die folgenden `-noheader-` Parameter angeben:

- -l

Gibt den Bezeichner an, der dem entpackten Schlüssel hinzugefügt werden soll.

Erforderlich: Ja

- -kc

Gibt die Klasse des zu entpackenden Schlüssels an. Die folgenden Werte sind zulässig:

3 = privater Schlüssel aus einem öffentlich-privaten Schlüsselpaar

4 = geheimer Schlüssel (symmetrisch)

Erforderlich: Ja

- -kt

Gibt den Typ des zu entpackenden Schlüssels an. Die folgenden Werte sind zulässig:

0 = RSA

1 = DSA

3 = ECC

16 = GENERIC_SECRET

21 = DES3

31 = AES

Erforderlich: Ja

Sie können auch optional die folgenden `-noheader`-Parameter angeben:

- -id

Die ID, die dem entpackten Schlüssel hinzugefügt werden soll.

Erforderlich: Nein

- -i

Erforderlich: Nein

[1] Gemäß den NIST-Richtlinien ist dies für Cluster im FIPS-Modus nach 2023 nicht zulässig. Für Cluster im Nicht-FIPS-Modus ist dies auch nach 2023 zulässig. Details dazu finden Sie unter [FIPS-140-Konformität: Mechanismus 2024 nicht mehr unterstützt](#).

Verwandte Themen

- [wrapKey](#)
- [exSymKey](#)
- [imSymKey](#)

Überprüfen Sie die Signatur einer Datei mit AWS CloudHSM KMU

Verwenden Sie den `verify` Befehl in AWS CloudHSM `key_mgmt_util`, um zu überprüfen, ob eine Datei mit einem bestimmten Schlüssel signiert wurde oder nicht. Hierzu vergleicht der Befehl `verify` eine signierte Datei mit einer Quelldatei und analysiert auf der Grundlage eines bestimmten öffentlichen Schlüssels und eines Signierungsmechanismus, ob in kryptografischer Hinsicht ein Zusammenhang zwischen ihnen besteht. Dateien können mit dem Vorgang angemeldet werden. AWS CloudHSM [sign](#)

Signierungsmechanismen werden durch die im Abschnitt [Parameter](#) aufgeführten Ganzzahlen dargestellt.

Bevor Sie einen `key_mgmt_util`-Befehl ausführen, müssen Sie [key_mgmt_util starten](#) und sich am HSM als Crypto-Benutzer (CU) [anmelden](#).

Syntax

```
verify -h

verify -f <message-file>
       -s <signature-file>
       -k <public-key-handle>
       -m <signature-mechanism>
```

Beispiel

Diese Beispiele zeigen, wie mit `verify` überprüft werden kann, ob ein bestimmter öffentlicher Schlüssel zum Signieren einer bestimmten Datei verwendet wurde.

Example: Überprüfen einer Dateisignatur

Mit diesem Befehl soll überprüft werden, ob eine Datei mit dem Namen `hardwareCert.crt` mit dem öffentlichen Schlüssel 262276 anhand des Signierungsmechanismus `SHA256_RSA_PKCS` signiert wurde, um die signierte Datei `hardwareCertSigned` zu erzeugen. Da es sich bei den angegebenen Parametern um ein echtes Signierungsverhältnis handelt, gibt der Befehl eine Erfolgsmeldung zurück.

```
Command: verify -f hardwareCert.crt -s hardwareCertSigned -k 262276 -m 1
```

```
Signature verification successful
```

```
Cfm3Verify returned: 0x00 : HSM Return: SUCCESS
```

Example: Nachweisen eines ungültigen Signierungsverhältnisses

Dieser Befehl überprüft, ob eine Datei mit dem Namen `hardwareCert.crt` mit dem öffentlichen Schlüssel 262276 anhand des Signierungsmechanismus `SHA256_RSA_PKCS` signiert wurde, um die signierte Datei `userCertSigned` zu erzeugen. Da die angegebenen Parameter kein echtes Signierungsverhältnis darstellen, gibt der Befehl eine Fehlermeldung zurück.

```
Command: verify -f hardwarecert.crt -s usercertsigned -k 262276 -m 1
```

```
Cfm3Verify returned: 0x1b
```

```
CSP Error: ERR_BAD_PKCS_DATA
```

Parameters

Dieser Befehl erfordert die folgenden Parameter.

-f

Der Name der ursprünglichen Nachrichtendatei.

Erforderlich: Ja

-s

Der Name der signierten Datei.

Erforderlich: Ja

-k

Das Handle des öffentlichen Schlüssels, der vermutlich zum Signieren der Datei verwendet wird.

Erforderlich: Ja

-m

Eine Ganzzahl, die den vorgeschlagenen und zum Signieren der Datei verwendeten Signierungsmechanismus repräsentiert. Die möglichen Mechanismen entsprechen den folgenden Ganzzahlen:

Signierungsmechanismus	Entsprechende Ganzzahl
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

Erforderlich: Ja

Verwandte Themen

- [sign](#)
- [getCert](#)
- [Generieren von Schlüsseln](#)

Exportieren Sie einen AWS CloudHSM Schlüssel mit KMU

Verwenden Sie den `wrapKey` Befehl in AWS CloudHSM `key_mgmt_util`, um eine verschlüsselte Kopie eines symmetrischen oder privaten Schlüssels aus dem Hardware-Sicherheitsmodul (HSM) in eine Datei zu exportieren. Wenn Sie `wrapKey` ausführen, geben Sie den zu exportierenden Schlüssel, einen Schlüssel in dem HSM zum Verschlüsseln (Wrap) des zu exportierenden Schlüssels und die Ausgabedatei an.

Der Befehl `wrapKey` schreibt den verschlüsselten Schlüssel in eine von Ihnen angegebene Datei, entfernt den Schlüssel jedoch nicht aus dem HSM oder verhindert, dass Sie ihn in kryptographischen Operationen verwenden. Sie können den gleichen Schlüssel mehrmals exportieren.

Nur der Eigentümer eines Schlüssels, d.h. der Crypto-Benutzer (CU), der den Schlüssel angelegt hat, kann ihn exportieren. Benutzer, für die der Schlüssel freigegeben ist, können ihn in kryptografischen Vorgängen verwenden, können ihn aber nicht exportieren.

Um den verschlüsselten Schlüssel wieder in das HSM zu importieren, verwenden Sie [unWrapKey](#). Um einen Klartext-Schlüssel aus einem HSM zu exportieren, verwenden Sie [exSymKey](#) oder [exportPrivateKey](#) je nach Bedarf. [aesWrapUnwrap](#) Mit dem Befehl können verschlüsselte Schlüssel nicht entschlüsselt (entpackt) werden. `wrapKey`

Bevor Sie einen `key_mgmt_util`-Befehl ausführen, müssen Sie [key_mgmt_util starten](#) und sich am HSM als Crypto-Benutzer (CU) [anmelden](#).

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>]
```

Beispiel

Example

Mit diesem Befehl wird ein symmetrischer 192-Bit-3DES-Schlüssel exportiert (Schlüssel-Handle 7). Der Befehl verwendet einen 256-Bit-AES-Schlüssel im HSM (Schlüssel-Handle 14), um den Schlüssel 7 zu verpacken. Anschließend wird der verschlüsselte 3DES-Schlüssel in die Datei `3DES-encrypted.key` geschrieben.

Die Ausgabe zeigt, dass der Schlüssel 7 (der 3DES-Schlüssel) erfolgreich verpackt und in die angegebene Datei geschrieben wurde. Der verschlüsselte Schlüssel hat eine Länge von 307 Byte.

```
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
```

Parameters

-h

Zeigt Hilfe für den Befehl an.

Erforderlich: Ja

-k

Der Schlüssel-Handle des Schlüssels, den Sie exportieren möchten. Geben Sie das Schlüssel-Handle eines symmetrischen oder privaten Schlüssels ein, der Ihnen gehört. Verwenden Sie den [findKey](#)-Befehl, um Schlüssel-Handles zu finden.

Verwenden Sie den Befehl [getAttribute](#), um sicherzustellen, dass ein Schlüssel exportiert werden kann, und um den Wert des Attributs OBJ_ATTR_EXTRACTABLE abzurufen, der von der Konstanten 354 dargestellt wird. Hilfe zur Interpretation der Schlüsselattribute finden Sie unter [AWS CloudHSM Referenz zu Schlüsselattributen für KMU](#).

Zudem können Sie nur Schlüssel exportieren, deren Eigentümer Sie sind. Verwenden Sie den Befehl, um den Besitzer eines Schlüssels zu ermitteln. [getKeyInfo](#)

Erforderlich: Ja

-w

Gibt den Wrapping-Schlüssel an. Geben Sie das Schlüssel-Handle eines AES-Schlüssels oder RSA-Schlüssels auf dem HSM ein. Dieser Parameter muss angegeben werden. Verwenden Sie den [findKey](#)-Befehl, um Schlüssel-Handles zu finden.

Um einen Wrapping-Schlüssel zu erstellen, verwenden Sie, [genSymKey](#) um einen AES-Schlüssel (Typ 31) oder ein [RSAKeyGen-Paar](#) zu generieren, um ein RSA-Schlüsselpaar (Typ 0) zu generieren. Wenn Sie ein RSA-Schlüsselpaar verwenden, achten Sie darauf, den Schlüssel mit einem der Schlüssel zu umwickeln und ihn mit dem anderen zu entpacken. Verwenden Sie [getAttribute](#), um zu überprüfen, ob ein Schlüssel als Verpackungsschlüssel verwendet werden kann, und um den Wert des OBJ_ATTR_WRAP-Attributs abzurufen, der von der Konstanten 262 dargestellt wird.

Erforderlich: Ja

-out

Pfad und Name der Ausgabedatei. Wenn der Befehl erfolgreich ist, enthält diese Datei eine verschlüsselte Kopie des exportierten Schlüssels. Wenn die Datei bereits vorhanden ist, überschreibt der Befehl sie ohne Warnung.

Erforderlich: Ja

-m

Der Wert, der den Verschlüsselungsmechanismus bezeichnet. CloudHSM unterstützt die folgenden Mechanismen:

Mechanismus	Wert
AES_KEY_WRAP_PAD_PKCS5	4

Mechanismus	Wert
NIST_AES_WRAP_NO_PAD	5
NIST_AES_WRAP_PAD	6
RSA_AES	7
RSA_OAEP (zur maximalen Datengröße lesen Sie den Hinweis weiter unten in diesem Abschnitt)	8
AES_GCM	10
CLOUDHSM_AES_GCM	11
RSA_PKCS (Zur maximalen Datengröße lesen Sie den Hinweis weiter unten in diesem Abschnitt.) Eine bevorstehende Änderung finden Sie im Hinweis 1 unten.	12

Erforderlich: Ja

Note

Wenn Sie den RSA_OAEP Wrapping-Mechanismus verwenden, wird die maximale Schlüsselgröße, die Sie umbrechen können, durch den Modul des RSA-Schlüssels und die Länge des angegebenen Hashs wie folgt bestimmt: Maximale Schlüsselgröße = $(\text{modulusLengthInBytes} - 2 * \text{hashLengthInBytes} - 2)$.

Wenn Sie den RSA_PKCS-Wrapping-Mechanismus verwenden, wird die maximale Schlüsselgröße, die Sie umbrechen können, durch das Modul des RSA-Schlüssels wie folgt bestimmt: Maximale Schlüsselgröße = $(\text{Byte} - 11) * \text{modulusLengthIn}$

-t

Der Wert, der den Hash-Algorithmus darstellt. CloudHSM unterstützt die folgenden Algorithmen:

Hash-Algorithmus	Wert
SHA1	2
SHA256	3
SHA384	4
SHA512	5
SHA224 (gültig für RSA_AES- und RSA_OAEP-Mechanismen)	6

Erforderlich: Nein

AAD

Der Dateiname mit AAD.

Note

Gültig nur für AES_GCM- und CLOUDHSM_AES_GCM-Mechanismen.

Erforderlich: Nein

-noheader

Lässt den Header weg, der CloudHSM-spezifische [Schlüsselattribute](#) angibt. Verwenden Sie diesen Parameter nur, wenn Sie planen, den Schlüssel mit Tools außerhalb von key_mgmt_util zu entpacken.

Erforderlich: Nein

-i

Der Initialisierungsvektor (IV) (Hex-Wert).

 Note

Gültig nur, wenn mit dem `-noheader`-Parameter für CLOUDHSM_AES_KEY_WRAP- und NIST_AES_WRAP-Mechanismen übergeben.

Erforderlich: Nein

`-iv_file`

Die Datei, in die Sie den als Antwort erhaltenen IV-Wert schreiben möchten.

 Note

Gültig nur, wenn mit dem `-noheader`-Parameter für den AES_GCM-Mechanismus übergeben.

Erforderlich: Nein

`-tag_size`

Die Größe des Tags, das zusammen mit verschlüsselttem Blob gespeichert werden soll.

 Note

Gültig nur, wenn mit dem `-noheader`-Parameter für AES_GCM- und CLOUDHSM_AES_GCM-Mechanismen übergeben. Die Mindestgröße der Tags ist acht.

Erforderlich: Nein

[1] Gemäß den NIST-Richtlinien ist dies für Cluster im FIPS-Modus nach 2023 nicht zulässig. Für Cluster im Nicht-FIPS-Modus ist dies auch nach 2023 zulässig. Details dazu finden Sie unter [FIPS-140-Konformität: Mechanismus 2024 nicht mehr unterstützt](#).

Verwandte Themen

- [exSymKey](#)

- [imSymKey](#)
- [unWrapKey](#)

AWS CloudHSM Referenz zu Schlüsselattributen für KMU

Die AWS CloudHSM `key_mgmt_util`-Befehle verwenden Konstanten, um die Attribute von Schlüsseln in einem Hardware-Sicherheitsmodul (HSM) darzustellen. Dieses Thema kann Ihnen helfen, die Attribute zu identifizieren, die Konstanten zu finden, die sie in Befehlen darstellen, und deren Werte zu verstehen.

Sie legen die Attribute eines Schlüssels fest, wenn Sie ihn erstellen. Um das Token-Attribut zu ändern, das angibt, ob ein Schlüssel persistent ist oder nur in der Sitzung existiert, verwenden Sie den Befehl [setAttribute](#) in `key_mgmt_util`. Um die Attribute `label`, `wrap`, `unwrap`, `encrypt` oder `decrypt` zu ändern, verwenden Sie den `setAttribute`-Befehl in `cloudhsm_mgmt_util`.

Verwenden Sie [listAttributes](#), um eine Liste der Attribute und deren Konstanten abzurufen. Um die Attributwerte für einen Schlüssel abzurufen, verwenden Sie [getAttribute](#).

In der folgenden Tabelle finden Sie die Schlüsselattribute, ihre Konstanten und ihre gültigen Werte.

Attribut	Konstante	Werte
OBJ_ATTR_ALL	512	Repräsentiert alle Attribute.
OBJ_ATTR_ALWAYS_SENSITIVE	357	0: False. 1: True.
OBJ_ATTR_CLASS	0	2: Öffentlicher Schlüssel in einem öffentlich-privaten Schlüsselpaar. 3: Privater Schlüssel in einem öffentlich-privaten Schlüssel paar. 4: Geheimer (symmetrischer) Schlüssel.
OBJ_ATTR_DECRYPT	261	0: False.

Attribut	Konstante	Werte
		1: True. Der Schlüssel kann zum Entschlüsseln von Daten verwendet werden.
OBJ_ATTR_DERIVE	268	0: False. 1: True. Die Funktion leitet den Schlüssel ab.
OBJ_ATTR_DESTROYABLE	370	0: False. 1: True.
OBJ_ATTR_ENCRYPT	260	0: False. 1: True. Der Schlüssel kann zum Verschlüsseln von Daten verwendet werden.
OBJ_ATTR_EXTRACTABLE	354	0: False. 1: True. Der Schlüssel kann aus dem exportiert werden. HSMs
OBJ_ATTR_ID	258	Benutzerdefinierte Zeichenfolge. Muss im Cluster eindeutig sein. Der Standardwert ist eine leere Zeichenfolge.
OBJ_ATTR_KCV	371	Schlüsselprüfwert des Schlüssels. Weitere Informationen finden Sie unter Weitere Details .

Attribut	Konstante	Werte
OBJ_ATTR_KEY_TYPE	256	0: RSA. 1: DSA. 3: EC. 16: Allgemeiner geheimer Schlüssel. 18: RC4. 21: Triple DES (3DES). 31: AES.
OBJ_ATTR_LABEL	3	Benutzerdefinierte Zeichenfolge. Muss im Cluster nicht eindeutig sein.
OBJ_ATTR_LOCAL	355	0. Falsch. Der Schlüssel wurde in den importiert HSMs. 1: True.
OBJ_ATTR_MODULUS	288	Der Modulus, der zum Generieren eines RSA-Schlüsselpaars verwendet wurde. Bei EC-Schlüsseln steht dieser Wert für die DER-Kodierung des ANSI ECPoint X9.62-Werts „Q“ in einem Hexadezimalformat. Für andere Schlüsseltypen ist dieses Attribut nicht vorhanden.

Attribut	Konstante	Werte
OBJ_ATTR_MODULUS_BITS	289	Die Länge des Modulus, der zum Generieren eines RSA-Schlüsselpaars verwendet wurde. Bei EC-Schlüsseln steht dies für die ID der elliptischen Kurve, die zur Generierung des Schlüssels verwendet wurde. Für andere Schlüsseltypen ist dieses Attribut nicht vorhanden.
OBJ_ATTR_NEVER_EXPORTABLE	356	0: False. 1: True. Der Schlüssel kann nicht exportiert werden. HSMs
OBJ_ATTR_PUBLIC_EXPONENT	290	Der öffentliche Exponent, der zum Generieren eines RSA-Schlüsselpaars verwendet wurde. Für andere Schlüsseltypen ist dieses Attribut nicht vorhanden.

Attribut	Konstante	Werte
OBJ_ATTR_PRIVATE	2	0: False. 1: True. Dieses Attribut gibt an, ob nicht authentifizierte Benutzer die Attribute des Schlüssels auflisten können. Da der CloudHSM PKCS # 11-Provider derzeit keine öffentlichen Sitzungen unterstützt, ist bei allen Schlüsseln (einschließlich öffentlicher Schlüssel in einem öffentlich-privaten Schlüsselpaar) dieses Attribut auf 1 festgelegt.
OBJ_ATTR_SENSITIVE	259	0: False. Öffentlicher Schlüssel in einem öffentlich-privaten Schlüsselpaar. 1: True.
OBJ_ATTR_SIGN	264	0: False. 1: True. Der Schlüssel kann zum Signieren verwendet werden (private Schlüssel).
OBJ_ATTR_TOKEN	1	0: False. Sitzungsschlüssel. 1: True. Persistenter Schlüssel.
OBJ_ATTR_TRUSTED	134	0: False. 1: True.

Attribut	Konstante	Werte
OBJ_ATTR_UNWRAP	263	0: False. 1: True. Der Schlüssel kann zum Entschlüsseln von Schlüsseln verwendet werden.
OBJ_ATTR_UNWRAP_TEMPLATE	1073742354	Für die Werte sollte die Attributvorlage verwendet werden, die auf jeden Schlüssel angewendet wird, der mit diesem Schlüssel zum Packen entpackt wird.
OBJ_ATTR_VALUE_LEN	353	Schlüssellänge in Bytes.
OBJ_ATTR_VERIFY	266	0: False. 1: True. Der Schlüssel für die Verifizierung verwendet werden (öffentliche Schlüssel).
OBJ_ATTR_WRAP	262	0: False. 1: True. Der Schlüssel kann zum Verschlüsseln von Schlüsseln verwendet werden.
OBJ_ATTR_WRAP_TEMPLATE	1073742353	Für die Werte sollte die Attributvorlage verwendet werden, die dem Schlüssel entspricht, der mit diesem Schlüssel zum Packen gepackt wurde.
OBJ_ATTR_WRAP_WITH_TRUSTED	528	0: False. 1: True.

Weitere Details

Schlüsselprüfwert (Key Check Value, KCV)

Der Schlüsselprüfwert (KCV) ist ein 3-Byte-Hash oder eine Prüfsumme eines Schlüssels, der generiert wird, wenn das HSM einen Schlüssel importiert oder generiert. Sie können einen KCV auch außerhalb des HSM berechnen, z. B. nachdem Sie einen Schlüssel exportiert haben. Anschließend können Sie die KCVs vergleichen, um die Identität und Integrität des Schlüssels zu bestätigen. Um den KCV eines Schlüssels abzurufen, verwenden Sie [getAttribute](#).

AWS CloudHSM verwendet die folgende Standardmethode, um einen Schlüsselprüfwert zu generieren:

- Symmetrische Schlüssel: Die ersten 3 Byte des Ergebnisses der Verschlüsselung eines Nullblocks mit dem Schlüssel.
- Asymmetrische Schlüsselpaare: Die ersten 3 Byte des SHA-1-Hashs des öffentlichen Schlüssels.
- HMAC-Schlüssel: KCV für HMAC-Schlüssel wird derzeit nicht unterstützt.

Offload-Operationen mit dem Client AWS CloudHSM SDKs

Verwenden Sie ein Client-SDK, um kryptografische Operationen von plattform- oder sprachbasierten Anwendungen auf Hardware-Sicherheitsmodule auszulagern (). HSMs

AWS CloudHSM bietet zwei Hauptversionen, und Client SDK 5 ist die neueste Version. Es bietet eine Reihe von Vorteilen gegenüber Client-SDK 3 (der vorherigen Serie). Weitere Informationen finden Sie unter [Vorteile von Client-SDK 5](#). Informationen zur Plattformunterstützung finden Sie unter [AWS CloudHSM Client SDK 5 unterstützte Plattformen](#).

In den folgenden Themen wird beschrieben, wie Sie mit dem AWS CloudHSM Client arbeiten SDKs.

AWS CloudHSM unterstützt die folgenden Komponenten:

[the section called “PKCS-#11-Bibliothek”](#)

PKCS #11 ist ein Standard für die Ausführung kryptografischer Operationen an Hardware-Sicherheitsmodulen ()HSMs. AWS CloudHSM bietet Implementierungen der PKCS #11 - Bibliothek, die mit PKCS #11, Version 2.40, kompatibel sind.

[the section called “OpenSSL Dynamic Engine”](#)

Mit der AWS CloudHSM OpenSSL Dynamic Engine können Sie kryptografische Operationen über die OpenSSL-API auf Ihren CloudHSM-Cluster auslagern.

[the section called “OpenSSL-Anbieter”](#)

Mit dem AWS CloudHSM OpenSSL Dynamic Engine Provider können Sie kryptografische Operationen über die OpenSSL Provider-API auf Ihren CloudHSM-Cluster auslagern.

[the section called “JCE-Anbieter”](#)

Der AWS CloudHSM JCE-Anbieter ist mit der Java Cryptographic Architecture (JCA) kompatibel. Der Anbieter ermöglicht es Ihnen, kryptografische Operationen auf dem HSM durchzuführen.

[the section called “Schlüsselspeicheranbieter \(KSP\)”](#)

Der AWS CloudHSM Client für Windows umfasst CNG- und KSP-Anbieter.

Themen

- [Überprüfen Sie Ihre AWS CloudHSM Client-SDK-Version](#)

- [Vergleichen Sie die Unterstützung von AWS CloudHSM Client SDK-Komponenten](#)
- [Migration von AWS CloudHSM Client SDK 3 zu Client SDK 5](#)
- [Verwenden Sie das Client SDK 5 für die Arbeit mit AWS CloudHSM](#)
- [Verwenden der vorherigen SDK-Version zum Arbeiten mit AWS CloudHSM](#)

Überprüfen Sie Ihre AWS CloudHSM Client-SDK-Version

Verwenden Sie die folgenden Befehle, um die Version des Client-SDK zu überprüfen, die Sie mit verwenden AWS CloudHSM.

Amazon Linux

Verwenden Sie den folgenden Befehl:

```
rpm -qa | grep ^cloudhsm
```

Amazon Linux 2

Verwenden Sie den folgenden Befehl:

```
rpm -qa | grep ^cloudhsm
```

CentOS 6

Verwenden Sie den folgenden Befehl:

```
rpm -qa | grep ^cloudhsm
```

CentOS 7

Verwenden Sie den folgenden Befehl:

```
rpm -qa | grep ^cloudhsm
```

CentOS 8

Verwenden Sie den folgenden Befehl:

```
rpm -qa | grep ^cloudhsm
```

RHEL 6

Verwenden Sie den folgenden Befehl:

```
rpm -qa | grep ^cloudhsm
```

RHEL 7

Verwenden Sie den folgenden Befehl:

```
rpm -qa | grep ^cloudhsm
```

RHEL 8

Verwenden Sie den folgenden Befehl:

```
rpm -qa | grep ^cloudhsm
```

Ubuntu 16.04 LTS

Verwenden Sie den folgenden Befehl:

```
apt list --installed | grep ^cloudhsm
```

Ubuntu 18.04 LTS

Verwenden Sie den folgenden Befehl:

```
apt list --installed | grep ^cloudhsm
```

Ubuntu 20.04 LTS

Verwenden Sie den folgenden Befehl:

```
apt list --installed | grep ^cloudhsm
```

Windows Server

Verwenden Sie den folgenden Befehl:

```
wmic product get name,version
```

Vergleichen Sie die Unterstützung von AWS CloudHSM Client SDK-Komponenten

Zusätzlich zu den Befehlszeilentools enthält das Client-SDK 3 Komponenten, die es ermöglichen, kryptografische Operationen von verschiedenen plattform- oder sprachbasierten Anwendungen auf das HSM auszulagern. Das Client-SDK 5 entspricht dem Client-SDK 3, unterstützt jedoch noch keine CNG- und KSP-Anbieter. In der folgenden Tabelle wird die Verfügbarkeit der Komponenten in Client-SDK 3 und Client-SDK 5 verglichen.

Komponente	Client-SDK 5	Client-SDK 3
PKCS-#11-Bibliothek	Ja	Ja
JCE-Anbieter	Ja	Ja
OpenSSL Dynamic Engine	Ja	Ja
Key Storage Provider (KSP)	Ja	Ja
CloudHSM-Verwaltungs- gsdienstprogramm (CMU) ¹	Ja	Ja
Schlüsselverwaltungsdienstp rogramm (KMU) ¹	Ja	Ja
Configure-Tool	Ja	Ja

[1] CMU- und KMU-Komponenten sind in der CloudHSM-CLI mit Client SDK 5 enthalten.

In den folgenden Abschnitten werden die Komponenten beschrieben.

PKCS-#11-Bibliothek

PKCS #11 ist ein Standard für die Ausführung kryptografischer Operationen an Hardware-Sicherheitsmodulen (HSMs). AWS CloudHSM bietet Implementierungen der PKCS #11 -Bibliothek, die mit PKCS #11, Version 2.40, kompatibel sind.

- Für das Client-SDK 3 ist die PKCS #11 -Bibliothek eine reine Linux-Komponente, die der Linux-Basisunterstützung entspricht. Weitere Informationen finden Sie unter [the section called “Linux-Unterstützung für Client SDK 3”](#).
- Für das Client-SDK 5 ist die PKCS #11 -Bibliothek eine plattformübergreifende Komponente, die der Basisunterstützung von Linux und Windows Client SDK 5 entspricht. Weitere Informationen erhalten Sie unter [the section called “Linux-Unterstützung für Client-SDK 5”](#) und [the section called “Windows-Unterstützung für Client-SDK 5”](#).

CloudHSM-Verwaltungsdienstprogramm (CMU)

Das Befehlszeilentool CloudHSM Management Utility (CMU) hilft Kryptobeamten bei der Verwaltung von Benutzern in der HSMs. Es umfasst Tools zum Erstellen, Löschen und Auflisten von Benutzern sowie zum Ändern von Benutzerpasswörtern. Weitere Informationen finden Sie unter [AWS CloudHSM Verwaltungsdienstprogramm \(CMU\)](#).

Schlüsselverwaltungsdienstprogramm (KMU)

Das Key Management Utility (KMU) ist ein Befehlszeilentool, das Krypto-Benutzern (CU) hilft, Schlüssel auf den Hardware-Sicherheitsmodulen (HSM) zu verwalten. Weitere Informationen finden Sie unter [AWS CloudHSM Schlüsselverwaltungsprogramm \(KMU\)](#).

JCE-Anbieter

Der AWS CloudHSM JCE-Anbieter ist mit der Java Cryptographic Architecture (JCA) kompatibel. Der Anbieter ermöglicht es Ihnen, kryptografische Operationen auf dem HSM durchzuführen.

Der JCE-Anbieter ist eine reine Linux-Komponente, die der Linux-Basisunterstützung entspricht. Weitere Informationen finden Sie unter [the section called “Linux-Unterstützung für Client SDK 3”](#).

- Für Client SDK 3 ist OpenJDK 1.8 erforderlich

OpenSSL Dynamic Engine

Mit der AWS CloudHSM OpenSSL Dynamic Engine können Sie kryptografische Operationen über die OpenSSL-API auf Ihren CloudHSM-Cluster auslagern.

- Für Client SDK 3 ist die OpenSSL Dynamic Engine eine reine Linux-Komponente, die nicht der Linux-Basisunterstützung entspricht. Sehen Sie sich die folgenden Ausnahmen an.
 - Benötigt OpenSSL 1.0.2[f+]

Nicht unterstützte Plattformen:

- CentOS 8
- Red Hat Enterprise Linux (RHEL) 8
- Ubuntu 18.04 LTS

Diese Plattformen werden mit einer Version von OpenSSL ausgeliefert, die mit OpenSSL Dynamic Engine für Client-SDK 3 nicht kompatibel ist. AWS CloudHSM unterstützt diese Plattformen mit OpenSSL Dynamic Engine für Client-SDK 5.

- Für Client SDK 5 ist die OpenSSL Dynamic Engine eine reine Linux-Komponente, die OpenSSL 1.0.2, 1.1.1 oder 3.x erfordert.

Schlüsselspeicheranbieter (KSP)

Key Storage Provider (KSP) ist eine kryptografische API, die für das Microsoft Windows-Betriebssystem spezifisch ist.

Beim Client SDK 3 handelt es sich bei den CNG- und KSP-Providern um eine reine Windows-Komponente, die der Windows-Basisunterstützung entspricht. Weitere Informationen finden Sie unter [Windows-Unterstützung für AWS CloudHSM Client SDK 3](#).

Für das Client-SDK 5 ist der Key Storage Provider (KSP) eine reine Windows-Komponente, die der Windows-Basisunterstützung entspricht. Weitere Informationen finden Sie unter [Windows-Unterstützung für AWS CloudHSM Client SDK 5](#).

Migration von AWS CloudHSM Client SDK 3 zu Client SDK 5

In AWS CloudHSM führen Kundenanwendungen mithilfe des AWS CloudHSM Client Software Development Kit (SDK) kryptografische Operationen durch. Das Client SDK 5 ist das primäre SDK, das weiterhin um neue Funktionen und Plattformunterstützung erweitert wird.

Das Client SDK 3 umfasst zwei separate Befehlszeilentools: die CMU für die Verwaltung von Benutzern und die KMU für die Verwaltung von Schlüsseln und die Ausführung von Vorgängen mit Schlüsseln. Das Client SDK 5 konsolidiert die Funktionen von CMU und KMU (Tools, die mit Client SDK 3 angeboten wurden) in einem einzigen Tool, dem. [AWS CloudHSM Befehlszeilenschnittstelle \(CLI\)](#) Benutzerverwaltungsoperationen finden Sie unter den Unterbefehlen und. [Die Benutzerkategorie in CloudHSM CLI](#) [Die Quorum-Kategorie in CloudHSM CLI](#) [Operationen zur Schlüsselverwaltung finden Sie unter dem Unterbefehl Schlüssel](#), und [kryptografische Operationen finden Sie unter dem Unterbefehl Crypto](#). Eine vollständige Liste der Befehle finden [Referenz für CloudHSM-CLI-Befehle](#) Sie unter.

Informationen zu den Vorteilen bei der Migration finden Sie unter [Vorteile von AWS CloudHSM Client SDK 5](#).

In den folgenden Themen finden Sie detaillierte Anweisungen zur Migration von Client SDK 3 zu Client SDK 5. Die neueste Version des AWS CloudHSM Client SDK ist 5.16.

- [Migrieren Sie Ihre AWS CloudHSM PKCS #11 -Bibliothek von Client SDK 3 zu Client SDK 5](#)
- [Migrieren Sie Ihre OpenSSL Dynamic Engine von AWS CloudHSM Client SDK 3 auf Client SDK 5](#)
- [Migrieren Sie Ihren Key Storage Provider \(KSP\) von AWS CloudHSM Client SDK 3 auf Client SDK 5](#)
- [Migrieren Sie Ihren JCE-Anbieter von AWS CloudHSM Client SDK 3 auf Client SDK 5](#)

Für Funktionen oder Anwendungsfälle, die von der CloudHSM-CLI nicht unterstützt werden, wenden Sie sich an. [AWS Support](#)

Migrieren Sie Ihre AWS CloudHSM PKCS #11 -Bibliothek von Client SDK 3 zu Client SDK 5

Verwenden Sie dieses Thema, um Ihre AWS CloudHSM [PKCS #11 -Bibliothek](#) von Client SDK 3 auf Client SDK 5 zu migrieren. Informationen zu den Vorteilen der Migration finden Sie unter. [Vorteile von AWS CloudHSM Client SDK 5](#)

In AWS CloudHSM führen Kundenanwendungen mithilfe des AWS CloudHSM Client Software Development Kit (SDK) kryptografische Operationen durch. Das Client SDK 5 ist das primäre SDK, das weiterhin um neue Funktionen und Plattformunterstützung erweitert wird.

Die Migrationsanweisungen für alle Anbieter finden Sie unter [Migration von AWS CloudHSM Client SDK 3 zu Client SDK 5](#).

Bereiten Sie sich vor, indem Sie wichtige Änderungen berücksichtigen

Überprüfen Sie diese grundlegenden Änderungen und aktualisieren Sie Ihre Anwendung in Ihrer Entwicklungsumgebung entsprechend.

Die Wickelmechanismen haben sich geändert

Mechanismus des Client-SDK 3	Entsprechender Client-SDK 5-Mechanismus
CKM_AES_KEY_WRAP	CKM_CLOUDHSM_AES_KEY_WRAP_P KCS5_PAD
CKM_AES_KEY_WRAP_PAD	CKM_CLOUDHSM_AES_KEY_WRAP_Z ERO_PAD
CKM_CLOUDHSM_AES_KEY_WRAP_P KCS5_PAD	CKM_CLOUDHSM_AES_KEY_WRAP_P KCS5_PAD
CKM_CLOUDHSM_AES_KEY_WRAP_NO_PAD	CKM_CLOUDHSM_AES_KEY_WRAP_NO_PAD
CKM_CLOUDHSM_AES_KEY_WRAP_Z ERO_PAD	CKM_CLOUDHSM_AES_KEY_WRAP_Z ERO_PAD

ECDH

In Client SDK 3 können Sie ECDH verwenden und ein KDF angeben. Diese Funktionalität ist derzeit in Client SDK 5 nicht verfügbar. Wenn Ihre Anwendung diese Funktionalität benötigt, wenden Sie sich bitte an den [Support](#).

Tastenkombinationen sind jetzt sitzungsspezifisch

Um Schlüsselnamen in Client-SDK 5 erfolgreich verwenden zu können, müssen Sie bei jeder Ausführung einer Anwendung die Schlüsselnamen abrufen. Wenn Sie bereits über Anwendungen verfügen, die in verschiedenen Sitzungen dieselben Schlüsselhandles verwenden, müssen Sie Ihren Code so ändern, dass das Schlüssel-Handle bei jeder Ausführung der Anwendung abgerufen wird. Informationen zum Abrufen von Schlüsselnamen finden Sie in [diesem AWS CloudHSM PKCS #11](#) - Beispiel. Diese Änderung entspricht der [PKCS #11 2.40-Spezifikation](#).

Migrieren Sie zum Client SDK 5

Folgen Sie den Anweisungen in diesem Abschnitt, um von Client SDK 3 auf Client SDK 5 zu migrieren.

Note

Amazon Linux, Ubuntu 16.04, Ubuntu 18.04, CentOS 6, CentOS 8 und RHEL 6 werden derzeit nicht mit Client SDK 5 unterstützt. Wenn Sie derzeit eine dieser Plattformen mit Client SDK 3 verwenden, müssen Sie bei der Migration zu Client SDK 5 eine andere Plattform wählen.

1. Deinstallieren Sie die PKCS #11 -Bibliothek für das Client SDK 3.

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. Beenden Sie den Client-Daemon für das Client-SDK 3.

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. Deinstallieren Sie den Client Daemon für Client SDK 3.

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

Benutzerdefinierte Konfigurationen müssen erneut aktiviert werden.

4. Installieren Sie die PKCS #11 -Bibliothek des Client SDK, indem Sie die Schritte unter befolgen. [Installieren Sie die PKCS #11 -Bibliothek für AWS CloudHSM Client SDK 5](#)
5. Das Client-SDK 5 führt ein neues Konfigurationsdateiformat und ein Befehlszeilen-Bootstrapping-Tool ein. Um Ihre PKCS #11 -Bibliothek für das Client SDK 5 zu booten, folgen Sie den Anweisungen im Benutzerhandbuch unter. [Bootstrap für das Client-SDK](#)
6. Testen Sie Ihre Anwendung in Ihrer Entwicklungsumgebung. Nehmen Sie vor der endgültigen Migration Aktualisierungen an Ihrem vorhandenen Code vor, um wichtige Änderungen zu beheben.

Verwandte Themen

- [Bewährte Methoden für AWS CloudHSM](#)

Migrieren Sie Ihre OpenSSL Dynamic Engine von AWS CloudHSM Client SDK 3 auf Client SDK 5

Verwenden Sie dieses Thema, um Ihre [OpenSSL Dynamic Engine](#) von AWS CloudHSM Client SDK 3 auf Client SDK 5 zu migrieren. Informationen zu den Vorteilen der Migration finden Sie unter [Vorteile von AWS CloudHSM Client SDK 5](#).

In AWS CloudHSM führen Kundenanwendungen mithilfe des AWS CloudHSM Client Software Development Kit (SDK) kryptografische Operationen durch. Das Client SDK 5 ist das primäre SDK, das weiterhin um neue Funktionen und Plattformunterstützung erweitert wird.

Note

Die Generierung von Zufallszahlen wird derzeit in Client SDK 5 mit OpenSSL Dynamic Engine nicht unterstützt.

Die Migrationsanweisungen für alle Anbieter finden Sie unter [Migration von AWS CloudHSM Client SDK 3 zu Client SDK 5](#).

Migrieren Sie zum Client SDK 5

Folgen Sie den Anweisungen in diesem Abschnitt, um von Client SDK 3 auf Client SDK 5 zu migrieren.

Note

Amazon Linux, Ubuntu 16.04, Ubuntu 18.04, CentOS 6, CentOS 8 und RHEL 6 werden derzeit nicht mit Client SDK 5 unterstützt. Wenn Sie derzeit eine dieser Plattformen mit Client SDK 3 verwenden, müssen Sie bei der Migration zu Client SDK 5 eine andere Plattform wählen.

1. Deinstallieren Sie die OpenSSL Dynamic Engine für Client SDK 3.

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. Stoppen Sie den Client-Daemon für das Client SDK 3.

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. Deinstallieren Sie den Client Daemon für Client SDK 3.

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

Benutzerdefinierte Konfigurationen müssen erneut aktiviert werden.

4. Installieren Sie das Client SDK OpenSSL Dynamic Engine, indem Sie die Schritte unter befolgen. [Installieren Sie die OpenSSL Dynamic Engine für AWS CloudHSM Client SDK 5](#)
5. Das Client SDK 5 führt ein neues Konfigurationsdateiformat und ein Befehlszeilen-Bootstrapping-Tool ein. Um Ihr Client SDK 5 OpenSSL Dynamic Engine zu booten, folgen Sie den Anweisungen im Benutzerhandbuch unter. [Bootstrap für das Client-SDK](#)
6. Testen Sie Ihre Anwendung in Ihrer Entwicklungsumgebung. Nehmen Sie vor der endgültigen Migration Aktualisierungen an Ihrem vorhandenen Code vor, um Ihre wichtigsten Änderungen zu beheben.

Verwandte Themen

- [Bewährte Methoden für AWS CloudHSM](#)

Migrieren Sie Ihren Key Storage Provider (KSP) von AWS CloudHSM Client SDK 3 auf Client SDK 5

In diesem Thema wird erklärt, wie Sie Ihren [Key Storage Provider \(KSP\)](#) von AWS CloudHSM Client SDK 3 auf Client SDK 5 migrieren. Die neueste Version des AWS CloudHSM Client SDK ist 5.16. Informationen zu den Vorteilen der Migration finden Sie unter [Vorteile von AWS CloudHSM Client SDK 5](#).

In AWS CloudHSM verwenden Sie das AWS CloudHSM Client Software Development Kit (SDK), um kryptografische Operationen durchzuführen. Das Client SDK 5 ist das primäre SDK, das neue Funktionen und Updates zur Plattformunterstützung erhält.

Anweisungen zur Migration für alle Anbieter finden Sie unter [Migration von AWS CloudHSM Client SDK 3 zu Client SDK 5](#).

Migrieren Sie zum Client SDK 5

1. Beenden Sie den Client-Daemon für das Client SDK 3.

```
PS C:\> Stop-Service "AWS CloudHSM Client"
```

2. Installieren Sie den Client SDK 5 Key Storage Provider (KSP) auf Ihrer Windows Server-Instanz. Detaillierte Anweisungen finden Sie unter [Installieren Sie den Key Storage Provider \(KSP\) für AWS CloudHSM Client SDK 5](#).
3. Konfigurieren Sie Ihren Client SDK 5 Key Storage Provider (KSP) mithilfe des neuen Konfigurationsdateiformats und des Befehlszeilen-Bootstrappingtools. Detaillierte Anweisungen finden Sie unter [Bootstrap für das Client-SDK](#).
4. Der Key Storage Provider (KSP) für AWS CloudHSM Client SDK 5 umfasst einen SDK3 Kompatibilitätsmodus zur Unterstützung der wichtigsten Referenzdateien, die in generiert wurden. SDK3 Weitere Informationen finden Sie unter [SDK3 Kompatibilitätsmodus für Key Storage Provider \(KSP\) für AWS CloudHSM](#).

 Note

Sie müssen den SDK3 Kompatibilitätsmodus aktivieren, wenn Sie vom Client SDK 3 generierte Schlüsselreferenzdateien mit Client SDK 5 verwenden.

Migrieren Sie zu neuen Windows Server-Instanzen

1. Führen Sie alle Schritte unter [Migration zu Client SDK 5](#) auf Ihren neuen Windows Server-Instanzen durch.
2. Suchen Sie nach vorhandenen wichtigen Referenzdateien

Suchen Sie auf Ihrer ursprünglichen Windows Server-Instanz unter nach wichtigen Referenzdateien `C:\Users\Default\AppData\Roaming\Microsoft\Crypto\CaviumKSP\GlobalPartition`.

- Wenn wichtige Referenzdateien vorhanden sind, kopieren Sie den gesamten Inhalt unter `C:\Users\Default\AppData\Roaming\Microsoft\Crypto\CaviumKSP` including `GlobalPartition` in denselben Verzeichnispfad auf Ihrer neuen Windows Server-Instanz. Erstellen Sie das Verzeichnis, falls es nicht existiert.
- Wenn wichtige Referenzdateien nicht existieren, verwenden Sie `cloudhsm-cli key generate-file --encoding ksp-key-reference` auf Ihrer neuen Windows Server-

Instanz, um sie zu erstellen. Detaillierte Anweisungen finden Sie unter [Generieren von KSP-Schlüsselreferenzen \(Windows\)](#).

3. Überprüfen Sie das Stammzertifikat

Überprüfen Sie Ihr Stammzertifikat bei den vertrauenswürdigen Stammzertifizierungsstellen:

```
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

Notieren Sie sich die Seriennummer des Zertifikats zur Verwendung im nächsten Schritt.

4. Exportieren Sie das Stammzertifikat

Exportieren Sie das Stammzertifikat in eine Datei:

```
certutil -store Root certificate-serial-number root-certificate-name.cer
```

5. Überprüfen Sie das HSM-Backend-Zertifikat

Überprüfen Sie Ihr HSM-Backend-Zertifikat im Personal Certificate Store:

```
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

Notieren Sie sich die Seriennummer des Zertifikats zur Verwendung im nächsten Schritt.

6. Exportieren Sie das HSM-Backend-Zertifikat

Exportieren Sie das HSM-Backend-Zertifikat in eine Datei:

```
certutil -store My certificate-serial-number signed-certificate-name.cer
```

7. Stammzertifikat importieren

Auf Ihrer neuen Windows-Instanz:

1. Kopieren Sie die Root-CA-Datei auf Ihre neue Windows-Instanz
2. Importieren Sie das Zertifikat:

```
certutil -addstore Root root-certificate-name.cer
```

8. Überprüfen Sie die Installation des Stammzertifikats

Stellen Sie sicher, dass das Stammzertifikat ordnungsgemäß installiert ist:

```

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-Backend-Zertifikat importieren

Auf Ihrer neuen Windows-Instanz:

1. Kopieren Sie das HSM-Backend-Zertifikat auf Ihre neue Windows-Instanz
2. Importieren Sie das Zertifikat:

```
certutil -addstore My signed-certificate-name.cer
```

10. Überprüfen Sie die Installation des HSM-Backend-Zertifikats

Stellen Sie sicher, dass das HSM-Backend-Zertifikat ordnungsgemäß installiert ist:

```

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

Notieren Sie sich die Seriennummer des Zertifikats zur Verwendung in den nachfolgenden Schritten.

11. Erstellen Sie eine Schlüsselreferenzdatei (optional)

Führen Sie diesen Schritt nur aus, wenn Sie eine neue Schlüsselreferenzdatei erstellen müssen. Andernfalls fahren Sie mit dem nächsten Schritt fort.

 Note

Diese Funktion ist nur in SDK-Version 5.16.0 und höher verfügbar.

1. Installiere [OpenSSL](#) und entpacke den Modulus:

```
openssl x509 -in signed-certificate-name.cer -modulus -noout
```

 Note

Der OpenSSL-Befehl gibt den Modulus im folgenden Format aus:.

Modulus=*modulus-value* Beachten Sie das *modulus-value* für die Verwendung im nächsten Befehl.

2. Erstellen Sie eine Schlüsselreferenzdatei mit der CloudHSM-CLI, siehe: [Generieren von KSP-Schlüsselreferenzen \(Windows\)](#)

```
& "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

Den Befehlsargumenten *modulus-value* in CloudHSM CLI muss ein Präfix vorangestellt werden, 0x um das Hexadezimalformat anzugeben.

Wichtige Referenzdateien werden in erstellt. C:\Users\Default\AppData
\Roaming\Microsoft\Crypto\CaviumKSP\GlobalPartition

12. Reparaturkonfiguration erstellen

Erstellen Sie eine Datei mit dem Namen `repair.txt` und dem folgenden Inhalt:

```
[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 Ersetzen Sie es durch den Schlüsselreferenzdateinamen von C:\Users\Default\AppData\Roaming\Microsoft\Crypto\CaviumKSP\GlobalPartition.

13. Reparieren Sie den Zertifikatsspeicher

Führen Sie den Reparaturbefehl aus:

```
certutil -repairstore My certificate-serial-number repair.txt
```

Note

Die Seriennummer des Zertifikats wurde anhand der vorherigen Schritte bei der Überprüfung der Installation des HSM-Backend-Zertifikats abgerufen.

14. Überprüfen Sie die Zertifikatszuweisung

Stellen Sie sicher, dass das Zertifikat ordnungsgemäß verknüpft ist:

```
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.
```

Stellen Sie sicher, dass die Ausgabe Folgendes anzeigt:

- Der richtige Name des Schlüsselcontainers
- Der Cavium-Schlüsselspeicheranbieter
- Das ERROR: Could not verify certificate public key against private key ist ein bekanntes Problem, siehe [Problem: Die Überprüfung eines Zertifikatsspeichers schlägt fehl](#)

15. Testen Sie Ihre Anwendung

Bevor Sie die Migration abschließen:

1. Testen Sie Ihre Anwendung in Ihrer Entwicklungsumgebung
2. Aktualisieren Sie Ihren Code, um alle wichtigen Änderungen zu beheben
3. Anwendungsspezifische Anleitungen finden Sie unter [Integration von Drittanbieteranwendungen mit AWS CloudHSM](#)

Überprüfen Sie die Migration

Stellen Sie nach Abschluss der Migrationsschritte sicher, dass:

- Ihre Zertifikate sind ordnungsgemäß in den richtigen Zertifikatsspeichern installiert
- Die wichtigsten Referenzdateien befinden sich am richtigen Speicherort
- Ihre Anwendung kann mithilfe der migrierten Zertifikate kryptografische Operationen ausführen

Fehlerbehebung

Wenn bei der Migration Probleme auftreten, überprüfen Sie Folgendes:

- Alle Zertifikate werden ordnungsgemäß aus dem Quellsystem exportiert
- Die Seriennummern der Zertifikate stimmen zwischen den Systemen überein
- Die Namen der Schlüsselcontainer in der Datei repair.txt stimmen mit Ihren wichtigsten Referenzdateien überein
- SDK3 Der Kompatibilitätsmodus ist aktiviert, wenn mit SDK3 -generierte Schlüsselreferenzdateien verwendet werden

Verwandte Themen

- [Bewährte Methoden für AWS CloudHSM](#)

Migrieren Sie Ihren JCE-Anbieter von AWS CloudHSM Client SDK 3 auf Client SDK 5

Verwenden Sie dieses Thema, um Ihren [JCE-Anbieter](#) von AWS CloudHSM Client SDK 3 auf Client SDK 5 zu migrieren. Informationen zu den Vorteilen der Migration finden Sie unter [Vorteile von AWS CloudHSM Client SDK 5](#)

In AWS CloudHSM führen Kundenanwendungen mithilfe des AWS CloudHSM Client Software Development Kit (SDK) kryptografische Operationen durch. Das Client SDK 5 ist das primäre SDK, das weiterhin um neue Funktionen und Plattformunterstützung erweitert wird.

Der Client SDK 3-JCE-Anbieter verwendet benutzerdefinierte Klassen APIs , die nicht Teil der Standard-JCE-Spezifikation sind. Das Client-SDK 5 für den JCE-Anbieter entspricht der JCE-Spezifikation und ist in bestimmten Bereichen abwärtsinkompatibel mit dem Client SDK 3. Kundenanwendungen müssen im Rahmen der Migration zum Client-SDK 5 möglicherweise geändert werden. In diesem Abschnitt werden die Änderungen beschrieben, die für eine erfolgreiche Migration erforderlich sind.

Die Migrationsanweisungen für alle Anbieter finden Sie unter [Migration von AWS CloudHSM Client SDK 3 zu Client SDK 5](#).

Themen

- [Bereiten Sie sich darauf vor, indem Sie wichtige Änderungen berücksichtigen](#)
- [Migrieren Sie zum Client SDK 5](#)
- [Verwandte Themen](#)

Bereiten Sie sich darauf vor, indem Sie wichtige Änderungen berücksichtigen

Überprüfen Sie diese grundlegenden Änderungen und aktualisieren Sie Ihre Anwendung in Ihrer Entwicklungsumgebung entsprechend.

Die Provider-Klasse und der Name haben sich geändert

Was hat sich geändert	Was es in Client SDK 3 war	Was ist es in Client SDK 5	Beispiel
Klasse und Name des Anbieters	Die JCE-Providerklasse im Client SDK 3 wird aufgerufen <code>CaviumProvider</code> und hat den Provider-Namen <code>Cavium</code> .	In Client SDK 5 wird die Provider-Klasse aufgerufen <code>CloudHsmProvider</code> und hat den Provider-Namen <code>CloudHSM</code> .	Ein Beispiel für die Initialisierung des <code>CloudHsmProvider</code> Objekts ist im AWS CloudHSM GitHub Beispiel-Repository verfügbar.

Die explizite Anmeldung hat sich geändert, die implizite nicht

Was hat sich geändert	Was es in Client SDK 3 war	Was ist es in Client SDK 5	Beispiel
Explizite Anmeldung	Das Client-SDK 3 verwendet die <code>LoginManager</code> Klasse für die explizite Anmeldung ¹ .	Im Client-SDK 5 implementiert der <code>CloudHSM</code> Anbieter <code>AuthProvider</code> die explizite Anmeldung. <code>AuthProvider</code> ist eine Standard-Java-Klasse und	Ein Beispiel für die Verwendung der expliziten Anmeldung mit Client SDK 5 finden Sie im <code>LoginRunner</code> Beispiel im AWS CloudHSM GitHub CloudHSM-Beispiel-Repository .

Was hat sich geändert	Was es in Client SDK 3 war	Was ist es in Client SDK 5	Beispiel
		folgt der idiomatischen Methode von Java, sich bei einem Provider anzumelden. Dank der verbesserten Verwaltung des Anmeldestatus im Client SDK 5 müssen Anwendungen die Anmeldung bei erneuten ² Verbindungen nicht mehr überwachen und durchführen.	
Implizite Anmeldung	Für die implizite Anmeldung sind keine Änderungen erforderlich. Dieselbe Eigenschaftendatei und alle Umgebungsvariablen funktionieren weiterhin für die implizite Anmeldung, wenn von Client SDK 3 auf Client SDK 5 migriert wird.		Ein Beispiel für die Verwendung der impliziten Anmeldung mit dem Client SDK 5 finden Sie im LoginRunner Beispiel im AWS CloudHSM GitHub Beispiel-Repository .

- [1] Codeausschnitt für das Client-SDK 3:

```

LoginManager lm = LoginManager.getInstance();

lm.login(partition, user, pass);

```

- [2] Codeausschnitt für das Client-SDK 5:

```

// 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);
```

Ein Beispiel für die Verwendung der expliziten Anmeldung mit dem Client SDK 5 finden Sie im [LoginRunner Beispiel im AWS CloudHSM GitHub Beispiel-Repository](#).

Die Schlüsselgenerierung hat sich geändert

Was hat sich geändert	Was es in Client SDK 3 war	Was ist es in Client SDK 5	Beispiel
Schlüsselgenerierung	Wird im Client-SDK 3 zur Angabe von Schlüsselgenerierungsparametern verwendet. <code>Cavium[Key-type]AlgorithmParameterSpec</code> . Einen Codeausschnitt finden Sie in der Fußnote. 1	Wird im Client SDK 5 zur Angabe von <code>KeyAttributes</code> verwendet. Einen Codeausschnitt finden Sie in der Fußnote. 2	Ein Beispiel <code>KeyAttributes</code> zur Generierung eines symmetrischen Schlüssels finden Sie im Beispiel im AWS CloudHSM SymmetricKeys GitHub CloudHSM-Beispiel-Repository .
Generierung von Schlüsselpaaren	Wird im Client SDK 3 verwendet, <code>Cavium[Key-type]AlgorithmParameterSpec</code> um Parameter für die Schlüsselpaargenerierung anzugeben. Einen Codeausschnitt finden Sie in der Fußnote. 3	Wird im Client SDK 5 verwendet, <code>KeyPairAttributes</code> um diese Parameter anzugeben. Einen Codeausschnitt finden Sie in der Fußnote. 4	Ein Beispiel <code>KeyPairAttributes</code> zur Generierung eines asymmetrischen Schlüssels finden Sie im AsymmetricKeys Beispiel im Beispiel-Repository . AWS CloudHSM GitHub

Was hat sich geändert	Was es in Client SDK 3 war	Was ist es in Client SDK 5	Beispiel
	hnitt finden Sie in der Fußnote. 3		

- [1] Codeausschnitt zur Schlüsselgenerierung des Client-SDK 3:

```
KeyGenerator keyGen = KeyGenerator.getInstance("AES", "Cavium");
CaviumAESKeyGenParameterSpec aesSpec = new CaviumAESKeyGenParameterSpec(
    keySizeInBits,
    keyLabel,
    isExtractable,
    isPersistent);
keyGen.init(aesSpec);
SecretKey aesKey = keyGen.generateKey();
```

- [2] Codeausschnitt zur Schlüsselgenerierung des Client-SDK 5:

```
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] Codeausschnitt zur Generierung von key pair für das Client-SDK 3:

```
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] Codeausschnitt zur Generierung von key pair für das Client-SDK 5:

```
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();
```

Das Suchen, Löschen und Referenzieren von Schlüsseln hat sich geändert

Um einen bereits generierten Schlüssel mit zu finden, AWS CloudHSM müssen Sie den KeyStore verwenden. Das Client-SDK 3 hat zwei KeyStore Typen: Cavium und CloudHSM. Das Client-SDK 5 hat nur einen KeyStore Typ: CloudHSM.

Der Wechsel von A Cavium KeyStore nach CloudHSM KeyStore erfordert eine Änderung des KeyStore Typs. Darüber hinaus verwendet das Client-SDK 3 Schlüsselnamen, um auf Schlüssel zu

verweisen, während das Client-SDK 5 Tastenbezeichnungen verwendet. Die daraus resultierenden Verhaltensänderungen sind unten aufgeführt.

Was hat sich geändert	Was es in Client SDK 3 war	Was ist es in Client SDK 5	Beispiel
Wichtige Referenzen	Mit Client SDK 3 verwenden Anwendungen entweder Tastenbezeichnungen oder Schlüsselhandles, um auf Schlüssel im HSM zu verweisen. Sie verwenden Labels mit KeyStore, um einen Schlüssel zu finden, oder sie verwenden Griffe und erstellen CaviumKey Objekte.	Im Client SDK 5 können Anwendung en die verwenden , AWS CloudHSM KeyStore Java-Klasse für Client SDK 5 um Schlüssel anhand von Labels zu finden. Verwenden Sie with, um Schlüssel anhand des Handles AWS CloudHSM KeyStoreWithAttributes zu finden AWS CloudHSM KeyReferenceSpec .	
Suche nach mehreren Einträgen	Bei der Suche nach einem Schlüssel mithilfe von getEntrygetKey, oder getCertificate in Szenarien , in denen mehrere Elemente mit denselben Kriterien existieren Cavium KeyStore, wird nur der erste gefundene	Bei Verwendung von AWS CloudHSM KeyStore und führt dasselbe Szenario dazuKeyStoreWithAttributes , dass eine Ausnahme ausgelöst wird. Um dieses Problem zu beheben, wird empfohlen, mithilfe des Legen Sie	

Was hat sich geändert	Was es in Client SDK 3 war	Was ist es in Client SDK 5	Beispiel
	Eintrag zurückgegeben.	die Attribute von Schlüsseln mit der CloudHSM-CLI fest Befehls in der CloudHSM-CLI eindeutige Bezeichnungen für Schlüssel festzulegen. Oder verwenden Sie diese KeyStoreWithAttributes#getKeys Option, um alle Schlüssel zurückzugeben, die den Kriterien entsprechen.	

Was hat sich geändert	Was es in Client SDK 3 war	Was ist es in Client SDK 5	Beispiel
Findet alle Schlüssel	Im Client SDK 3 ist es möglich, alle Schlüssel im HSM mithilfe von <code>Util.findAllKeys()</code> zu finden.	Das Client SDK 5 macht das Auffinden von Schlüsseln mithilfe der <code>KeyStoreWithAttributes</code> Klasse einfacher und effizienter. Wenn möglich, speichern Sie Ihre Schlüssel im Cache, um die Latenz zu minimieren. Weitere Informationen finden Sie unter Effektives Verwalten von Schlüsseln in Ihrer Anwendung . Wenn Sie alle Schlüssel aus dem HSM abrufen müssen, verwenden Sie <code>KeyStoreWithAttributes#getKeys</code> mit einem leeren <code>KeyAttributesMap</code> Schlüssel.	Ein Beispiel, das die <code>KeyStoreWithAttributes</code> Klasse verwendet, um einen Schlüssel zu finden, ist im AWS CloudHSM GitHub Beispiel-Repository verfügbar. Ein Codeausschnitt ist unter zu finden. 1

Was hat sich geändert	Was es in Client SDK 3 war	Was ist es in Client SDK 5	Beispiel
Löschen von Schlüsseln	Client SDK 3 verwendet <code>Util.deleteKey()</code> , um einen Schlüssel zu löschen.	Das Key Objekt in Client SDK 5 implementiert die <code>Destroyable</code> Schnittstelle, die das Löschen von Schlüsseln mithilfe der <code>destroy()</code> Methode dieser Schnittstelle ermöglicht.	Ein Beispielcode, der die Funktion zum Löschen von Schlüsseln zeigt, finden Sie im GitHub CloudHSM-Beispiel-Repository . Ein Beispielausschnitt für jedes SDK finden Sie unter. 2

- [1] Ein Ausschnitt ist unten dargestellt:

```
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] Löschen eines Schlüssels im Client SDK 3:

```
Util.deleteKey(key);
```

Löschen eines Schlüssels im Client-SDK 5:

```
((Destroyable) key).destroy();
```

Operationen zum Entpacken von Chiffren haben sich geändert, andere Verschlüsselungsoperationen nicht

 Note

Für Chiffrieroperationen sind keine Änderungen erforderlich. encrypt/decrypt/wrap

Bei Unwrap-Vorgängen muss die Client SDK CaviumUnwrapParameterSpec 3-Klasse durch eine der folgenden Klassen ersetzt werden, die für die aufgeführten kryptografischen Operationen spezifisch sind.

- GCMUnwrapKeySpec zum Auspacken AES/GCM/NoPadding
- IvUnwrapKeySpec für und AESWrap unwrap AES/CBC/NoPadding unwrap
- OAEPUnwrapKeySpec für RSA OAEP unwrap

Beispielausschnitt für: OAEPUnwrapKeySpec

```
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);
```

Die Signaturoperationen haben sich nicht geändert

Für Signaturoperationen sind keine Änderungen erforderlich.

Migrieren Sie zum Client SDK 5

Folgen Sie den Anweisungen in diesem Abschnitt, um von Client SDK 3 auf Client SDK 5 zu migrieren.

Note

Amazon Linux, Ubuntu 16.04, Ubuntu 18.04, CentOS 6, CentOS 8 und RHEL 6 werden derzeit nicht mit Client SDK 5 unterstützt. Wenn Sie derzeit eine dieser Plattformen mit Client SDK 3 verwenden, müssen Sie bei der Migration zu Client SDK 5 eine andere Plattform wählen.

1. Deinstallieren Sie den JCE-Anbieter für Client SDK 3.

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. Stoppen Sie den Client-Daemon für das Client SDK 3.

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. Deinstallieren Sie den Client Daemon für Client SDK 3.

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

Benutzerdefinierte Konfigurationen müssen erneut aktiviert werden.

4. Installieren Sie den Client SDK JCE Provider, indem Sie die Schritte unter befolgen. [Installieren Sie den JCE-Anbieter für AWS CloudHSM Client SDK 5](#)
5. Das Client SDK 5 führt ein neues Konfigurationsdateiformat und ein Befehlszeilen-Bootstrapping-Tool ein. Folgen Sie den Anweisungen im Benutzerhandbuch unter, um Ihren Client SDK 5 JCE-Anbieter zu booten. [Bootstrap für das Client-SDK](#)
6. Testen Sie Ihre Anwendung in Ihrer Entwicklungsumgebung. Nehmen Sie vor der endgültigen Migration Aktualisierungen an Ihrem vorhandenen Code vor, um Ihre wichtigsten Änderungen zu beheben.

Verwandte Themen

- [Bewährte Methoden für AWS CloudHSM](#)

Verwenden Sie das Client SDK 5 für die Arbeit mit AWS CloudHSM

AWS CloudHSM umfasst zwei Hauptversionen des Client-SDK:

- Client-SDK 5: Dies ist unser neuestes und standardmäßiges Client-SDK. Informationen zu den Vorteilen und Nutzen, die es bietet, finden Sie unter [Vorteile von AWS CloudHSM Client SDK 5](#).
- Client-SDK 3: Dies ist unser älteres Client-SDK. Es enthält einen vollständigen Satz von Komponenten für die Kompatibilität von plattform- und sprachbasierten Anwendungen und Verwaltungstools.

Anweisungen zur Migration von Client SDK 3 auf Client SDK 5 finden Sie unter [Migration von AWS CloudHSM Client SDK 3 zu Client SDK 5](#).

In diesem Thema wird das Client SDK 5 beschrieben. Informationen darüber, welche Version des Client-SDK Sie verwenden, finden Sie unter

Themen

- [Vorteile von AWS CloudHSM Client SDK 5](#)
- [AWS CloudHSM Client SDK 5 unterstützte Plattformen](#)
- [PKCS #11 -Bibliothek für AWS CloudHSM Client SDK 5](#)
- [OpenSSL Dynamic Engine für AWS CloudHSM Client SDK 5](#)
- [OpenSSL-Anbieter für AWS CloudHSM Client SDK 5](#)
- [Schlüsselspeicheranbieter \(KSP\) für AWS CloudHSM Client SDK 5](#)
- [JCE-Anbieter für AWS CloudHSM Client SDK 5](#)

Vorteile von AWS CloudHSM Client SDK 5

Im Vergleich zu AWS CloudHSM Client SDK 3 ist Client SDK 5 einfacher zu verwalten, bietet eine bessere Konfigurierbarkeit und eine höhere Zuverlässigkeit. Client-SDK 5 bietet außerdem einige zusätzliche wichtige Vorteile gegenüber Client-SDK 3.

Konzipiert für Serverless-Architekturen

Das Client-SDK 5 benötigt keinen Client-Daemon, sodass Sie keinen Hintergrunddienst mehr verwalten müssen. Dies hilft Benutzern in einigen wichtigen Punkten:

- Vereinfacht den Startvorgang der Anwendung. Um mit CloudHSM zu beginnen, müssen Sie lediglich das SDK konfigurieren, bevor Sie Ihre Anwendung ausführen.
- Sie benötigen keinen ständig laufenden Prozess, was die Integration mit Serverless-Komponenten wie Lambda und Elastic Container Service (ECS) erleichtert.

Bessere Integrationen von Drittanbietern und einfachere Portabilität

Das Client-SDK 5 folgt genau der JCE-Spezifikation und bietet eine einfachere Portabilität zwischen verschiedenen JCE-Anbietern sowie bessere Integrationen von Drittanbietern

Verbesserte Benutzererfahrung und Konfigurierbarkeit

Das Client-SDK 5 verbessert die Lesbarkeit von Protokollnachrichten und bietet klarere Ausnahmen und Mechanismen zur Fehlerbehandlung, was die Self-Service-Suche für Benutzer erheblich erleichtert. SDK 5 bietet auch eine Vielzahl von Konfigurationen, die auf der [Seite Configure Tool](#) aufgeführt sind.

Breitere Plattformunterstützung

Client-SDK 5 bietet mehr Unterstützung für moderne Betriebsplattformen. Dies beinhaltet Unterstützung für ARM-Technologien und eine bessere Unterstützung für [JCE](#), [PKCS #11](#) und [OpenSSL](#). Weitere Informationen finden Sie unter [Unterstützte Plattformen](#).

IPv6 Verbindungsunterstützung

Das Client-SDK 5.14+ unterstützt Verbindungen zu HSMs Dual-Stack unter Verwendung von IPv6

Zusätzliche Funktionen und Mechanismen

Client-SDK 5 enthält zusätzliche Funktionen und Mechanismen, die in Client-SDK 3 nicht verfügbar sind, und Client-SDK 5 wird in Zukunft weitere Mechanismen hinzufügen.

AWS CloudHSM Client SDK 5 unterstützte Plattformen

Die AWS CloudHSM Basisunterstützung ist für jede Version des Client-SDK unterschiedlich. Die Plattformunterstützung für Komponenten in einem SDK entspricht in der Regel der Basisunterstützung, jedoch nicht immer. Um die Plattformunterstützung für eine bestimmte Komponente zu ermitteln, stellen Sie zunächst sicher, dass die gewünschte Plattform im Basisbereich für das SDK angezeigt wird, und suchen Sie dann im Komponentenabschnitt nach Ausschlüssen oder anderen relevanten Informationen.

AWS CloudHSM unterstützt nur 64-Bit-Betriebssysteme.

Die Plattformunterstützung ändert sich im Laufe der Zeit. Frühere Versionen des CloudHSM-Client SDK unterstützen möglicherweise nicht alle hier aufgeführten Betriebssysteme. Ermitteln Sie anhand der Versionshinweise die Betriebssystemunterstützung für frühere Versionen des CloudHSM-Client SDK. Weitere Informationen finden Sie unter [Downloads für das AWS CloudHSM Client-SDK](#).

Informationen zu den unterstützten Plattformen für das vorherige Client-SDK finden Sie unter [AWS CloudHSM Unterstützte Plattformen vom Client SDK 3](#).

Für das Client-SDK 5 ist kein Client-Daemon erforderlich.

Themen

- [Linux-Unterstützung für AWS CloudHSM Client SDK 5](#)
- [Windows-Unterstützung für AWS CloudHSM Client SDK 5](#)
- [Serverlose Unterstützung für AWS CloudHSM Client SDK 5](#)
- [HSM-Kompatibilität für AWS CloudHSM Client SDK 5](#)

Linux-Unterstützung für AWS CloudHSM Client SDK 5

AWS CloudHSM Das Client SDK 5 unterstützt die folgenden Linux-Betriebssysteme und -Plattformen.

Unterstützte Plattformen	X86_64-Architektur	ARM-Architektur
Amazon Linux 2	Ja	Ja
Amazon Linux 2023	Ja	Ja

Unterstützte Plattformen	X86_64-Architektur	ARM-Architektur
Red Hat Enterprise Linux 8 (8.3+)	Ja	Ja
RedHat Enterprise Linux 9 (9,2 und höher)	Ja	Ja
RedHat Enterprise Linux 10 (10.0+)	Ja	Ja
Ubuntu 22.04 LTS	Ja	Ja
Ubuntu 24.04 LTS	Ja	Ja

- SDK 5.16 war die letzte Version, die Unterstützung für die Ubuntu 20.04 LTS-Plattform bot. [Weitere Informationen finden Sie auf der Ubuntu-Website.](#)
- SDK 5.12 war die letzte Version, die CentOS 7 (7.8+) Plattformunterstützung bot. Weitere Informationen finden Sie auf der [CentOS-Website](#).
- SDK 5.12 war die letzte Version, die Unterstützung für die Plattform Red Hat Enterprise Linux 7 (7.8+) bot. Weitere Informationen finden Sie auf der [Red Hat Website](#).
- SDK 5.4.2 war die letzte Version, die CentOS 8-Plattformunterstützung bot. Weitere Informationen finden Sie auf der [CentOS-Website](#).

Windows-Unterstützung für AWS CloudHSM Client SDK 5

AWS CloudHSM Das Client SDK 5 unterstützt die folgenden Versionen von Windows Server.

- Microsoft Windows Server 2016
- Microsoft Windows Server 2019
- Microsoft Windows Server 2022
- Microsoft Windows Server 2025

Serverlose Unterstützung für AWS CloudHSM Client SDK 5

AWS CloudHSM Das Client SDK 5 unterstützt die folgenden AWS serverlosen Dienste.

- AWS Lambda
- Docker/ECS

HSM-Kompatibilität für AWS CloudHSM Client SDK 5

In der folgenden Tabelle wird die Kompatibilität AWS CloudHSM des Client SDK 5 für HSMs beschrieben.

hsm1.medium	hsm2m. mittel
Kompatibel mit Client SDK Version 5.0.0 und höher.	Kompatibel mit der Client-SDK-Version 5.9.0 und höher.

PKCS #11 -Bibliothek für AWS CloudHSM Client SDK 5

PKCS #11 ist ein Standard für die Ausführung kryptografischer Operationen an Hardware-Sicherheitsmodulen (). HSMs AWS CloudHSM bietet Implementierungen der PKCS #11 -Bibliothek, die mit PKCS #11, Version 2.40, kompatibel sind.

Informationen zu Bootstrapping finden Sie unter [Verbinden mit dem Cluster](#). Informationen zur Problembehandlung finden Sie unter [Bekannte Probleme mit der PKCS #11 -Bibliothek für AWS CloudHSM](#)

Informationen zur Verwendung von Client-SDK 3 finden Sie unter [Verwenden der vorherigen SDK-Version zum Arbeiten mit AWS CloudHSM](#).

Themen

- [Installieren Sie die PKCS #11 -Bibliothek für AWS CloudHSM Client SDK 5](#)
- [Authentifizieren Sie sich bei der PKCS #11 -Bibliothek für AWS CloudHSM Client SDK 5](#)
- [Unterstützte Schlüsseltypen für die PKCS #11 -Bibliothek für das AWS CloudHSM Client-SDK 5](#)
- [Unterstützte Mechanismen für die PKCS #11 -Bibliothek für AWS CloudHSM das Client-SDK 5](#)
- [Unterstützte API-Operationen für die PKCS #11 -Bibliothek für das Client-SDK 5 AWS CloudHSM](#)
- [Schlüsselattribute in der PKCS #11 -Bibliothek für AWS CloudHSM das Client-SDK 5](#)
- [Codebeispiele für die PKCS #11 -Bibliothek für das AWS CloudHSM Client-SDK 5](#)
- [Erweiterte Konfigurationen für die PKCS #11 -Bibliothek für AWS CloudHSM](#)

- [Speicherung von Zertifikaten mit der PKCS #11 -Bibliothek](#)

Installieren Sie die PKCS #11 -Bibliothek für AWS CloudHSM Client SDK 5

Dieses Thema enthält Anweisungen zur Installation der neuesten Version der PKCS #11 -Bibliothek für die AWS CloudHSM Client SDK 5-Versionsserie. Weitere Informationen zum Client-SDK oder zur PKCS #11-Bibliothek finden Sie unter [Verwenden des Client-SDK](#) und der [PKCS #11-Bibliothek](#).

Mit dem Client-SDK 5 müssen Sie keinen Client-Daemon installieren oder ausführen.

Um einen einzelnen HSM-Cluster mit Client-SDK 5 auszuführen, müssen Sie zunächst die Einstellungen für die Haltbarkeit von Client-Schlüsseln verwalten, indem Sie die Einstellung `disable_key_availability_check` auf `True` festlegen. Weitere Informationen finden Sie unter [Schlüsselsynchronisierung](#) und [Client-SDK-5-Configure-Tool](#).

Weitere Informationen über die PKCS #11-Bibliothek in Client-SDK 5 finden Sie unter [PKCS #11-Bibliothek](#).

Note

Um einen einzelnen HSM-Cluster mit Client-SDK 5 auszuführen, müssen Sie zunächst die Einstellungen für die Haltbarkeit von Client-Schlüsseln verwalten, indem Sie die Einstellung `disable_key_availability_check` auf `True` festlegen. Weitere Informationen finden Sie unter [Schlüsselsynchronisierung](#) und [Client-SDK-5-Configure-Tool](#).

Installieren und Konfigurieren der PKCS #11-Bibliothek

1. Verwenden Sie die folgenden Befehle zum Herunterladen und Installieren der PKCS #11-Bibliothek.

Amazon Linux 2023

Installieren Sie die PKCS #11 -Bibliothek für Amazon Linux 2023 auf der X86_64-Architektur:

```
$ 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
```

Installieren Sie die PKCS #11 -Bibliothek für Amazon Linux 2023 auf der ARM64 Architektur:

```
$ 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

Installieren Sie die PKCS #11-Bibliothek für Amazon Linux 2 auf der X86_64-Architektur:

```
$ 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
```

Installieren Sie die PKCS #11 -Bibliothek für Amazon Linux 2 auf der ARM64 Architektur:

```
$ 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+)

Installieren Sie die PKCS #11 -Bibliothek für RHEL 10 auf der X86_64-Architektur:

```
$ 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
```

Installieren Sie die PKCS #11 -Bibliothek für RHEL 10 auf der folgenden Architektur: ARM64

```
$ 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+)

Installieren Sie die PKCS #11 -Bibliothek für RHEL 9 auf der X86_64-Architektur:

```
$ 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
```

Installieren Sie die PKCS #11 -Bibliothek für RHEL 9 auf folgender Architektur: ARM64

```
$ 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+)

Installieren Sie die PKCS #11 -Bibliothek für RHEL 8 auf der X86_64-Architektur:

```
$ 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
```

Installieren Sie die PKCS #11 -Bibliothek für RHEL 8 auf der folgenden Architektur: ARM64

```
$ 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 24.04 LTS

Installieren Sie die PKCS #11 -Bibliothek für Ubuntu 24.04 LTS auf der X86_64-Architektur:

```
$ 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
```

Installieren Sie die PKCS #11 -Bibliothek für Ubuntu 24.04 LTS auf der Architektur: ARM64

```
$ 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

Installieren Sie die PKCS #11-Bibliothek für Ubuntu 22.04 LTS auf der X86_64-Architektur:

```
$ 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
```

Installieren Sie die PKCS #11 -Bibliothek für Ubuntu 22.04 LTS auf der Architektur: ARM64

```
$ 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

Installieren Sie die PKCS #11 -Bibliothek für Windows Server auf der X86_64-Architektur:

1. Laden Sie die [PKCS #11-Bibliothek](#) für Client-SDK 5 herunter.

2. Führen Sie das PKCS #11 -Bibliotheksinstallationsprogramm (AWSCloudHSM PKCS11-latest.msi) mit Windows-Administratorrechten aus.
2. Verwenden Sie das Configure-Tool, um den Speicherort des ausstellenden Zertifikats anzugeben. Detaillierte Anweisungen finden Sie unter [Geben Sie den Speicherort des ausstellenden Zertifikats an](#).
3. Um eine Verbindung zu Ihrem Cluster herzustellen, siehe [Bootstrap für das Client-SDK](#).
4. Die PKCS #11-Bibliotheksdateien finden Sie an den folgenden Speicherorten:
 - Linux-Binärdateien, Konfigurationsskripten und Protokolldateien:

```
/opt/cloudhsm
```

Windows-Binärdateien:

```
C:\Program Files\Amazon\CloudHSM
```

Windows-Konfigurationsskripten und Protokolldateien:

```
C:\ProgramData\Amazon\CloudHSM
```

Authentifizieren Sie sich bei der PKCS #11 -Bibliothek für AWS CloudHSM Client SDK

5

Wenn Sie die PKCS #11 -Bibliothek verwenden, wird Ihre Anwendung unter einem bestimmten [Crypto User \(CU\)](#) in Ihrem Konto ausgeführt. HSMs AWS CloudHSM Ihre Anwendung kann nur die Schlüssel anzeigen und verwalten, die der CU besitzt und freigibt. Sie können eine vorhandene CU in Ihrer Anwendung verwenden HSMs oder eine neue CU für Ihre Anwendung erstellen. Informationen zur [Verwaltung CUs finden Sie unter HSM-Benutzer mit CloudHSM CLI verwalten und HSM-Benutzer mit CloudHSM Management Utility \(CMU\) verwalten](#)

Um den CU für die PKCS #11-Bibliothek anzugeben, verwenden Sie den Pin-Parameter der PKCS #11-Funktion [C_Login](#). Denn der AWS CloudHSM Pin-Parameter hat das folgende Format:

```
<CU_user_name>:<password>
```

Mit dem folgenden Befehl wird beispielsweise dem PKCS #11-Bibliothek-Pin der CU mit dem Benutzernamen `CryptoUser` und dem Passwort `CUPassword123!` zugewiesen.

```
CryptoUser:CUPassword123!
```

Unterstützte Schlüsseltypen für die PKCS #11 -Bibliothek für das AWS CloudHSM Client-SDK 5

Die PKCS #11 -Bibliothek für AWS CloudHSM Client SDK 5 unterstützt die folgenden Schlüsseltypen.

Schlüsseltyp	Description
AES	Generieren der 128-, 192- und 256-Bit-AES-Schlüssel.
Dreifaches DES (3DES,) DESede	Generieren von 192-Bit-Triple-DES-Schlüssel. In. Eine bevorstehende Änderung finden Sie im Hinweis 1 unten.
EC	Generieren Sie Schlüssel mit den Kurven <code>secp224r1</code> (P-224), <code>secp256r1</code> (P-256), <code>secp256k1</code> (Blockchain), <code>secp384r1</code> (P-384) und <code>secp521r1</code> (P-521).
GENERIC_SECRET	Generieren Sie generische Geheimnisse mit 1 bis 800 Byte.
RSA	Generieren Sie RSA-Schlüssel mit 2048 bis 4096 Bit, in Schritten von 256 Bit.

[1] Gemäß den NIST-Richtlinien ist dies für Cluster im FIPS-Modus nach 2023 nicht zulässig. Für Cluster im Nicht-FIPS-Modus ist dies auch nach 2023 zulässig. Details dazu finden Sie unter [FIPS-140-Konformität: Mechanismus 2024 nicht mehr unterstützt](#).

Unterstützte Mechanismen für die PKCS #11 -Bibliothek für AWS CloudHSM das Client-SDK 5

Die PKCS #11-Bibliothek entspricht Version 2.40 der PKCS #11-Spezifikation. Um eine kryptographische Funktion mit PKCS#11 aufzurufen, rufen Sie eine Funktion mit einem bestimmten Mechanismus auf. In den folgenden Abschnitten werden die vom AWS CloudHSM Client SDK 5 unterstützten Kombinationen von Funktionen und Mechanismen zusammengefasst.

Die PKCS #11-Bibliothek unterstützt die folgenden Algorithmen:

- Verschlüsselung und Entschlüsselung — AES-CBC, AES-CTR, AES-ECB, AES-GCM, -CBC, -ECB, RSA-OAEP und RSA-PKCS DES3 DES3
- Signieren und Verifizieren – RSA, HMAC und ECDSA; mit und ohne Hashing
- SHA1 SHA256 SHA384Hash/Digest SHA224 —,, und SHA512
- Key Wrap – AES Key Wrap¹, AES-GCM, RSA-AES und RSA-OAEP
- Schlüsselableitung — SP800 -108 Counter KDF und ECDH mit KDF (Unterstützte KDF-Algorithmen sind X9.63 mit,,,,) SHA1 SHA224 SHA256 SHA384 SHA512

Themen

- [Schlüssel- und Schlüsselpaarfunktionen generieren](#)
- [Funktionen zum Signieren und Überprüfen](#)
- [Funktionen zur Signierung, Wiederherstellung und Überprüfung der Wiederherstellung](#)
- [Digest-Funktionen](#)
- [Funktionen zum Verschlüsseln und Entschlüsseln](#)
- [Schlüsselfunktionen ableiten](#)
- [Funktionen „Packen“ und „Entpacken“](#)
- [Maximale Datengröße für jeden Mechanismus](#)
- [Anmerkungen zum Mechanismus](#)

Schlüssel- und Schlüsselpaarfunktionen generieren

AWS CloudHSM Mit der Softwarebibliothek für die PKCS #11 -Bibliothek können Sie die folgenden Mechanismen für die Funktionen Schlüssel generieren und Schlüsselpaar verwenden.

- CKM_RSA_PKCS_KEY_PAIR_GEN

- CKM_RSA_X9_31_KEY_PAIR_GEN – Dieser Mechanismus ist funktionell identisch mit dem CKM_RSA_PKCS_KEY_PAIR_GEN-Mechanismus, aber bietet stärkere Garantien für die Generierung von p und q.
- CKM_EC_KEY_PAIR_GEN
- CKM_GENERIC_SECRET_KEY_GEN
- CKM_AES_KEY_GEN
- CKM_DES3_KEY_GEN – bevorstehende Änderung in der Fußnote [5](#) aufgeführt.

Funktionen zum Signieren und Überprüfen

Mit der AWS CloudHSM Softwarebibliothek für PKCS #11 können Sie die folgenden Mechanismen für die Funktionen Signieren und Überprüfen verwenden. Mit Client-SDK 5 werden die Daten lokal in der Software gehasht. Das bedeutet, dass die Größe der Daten, die vom SDK gehasht werden können, unbegrenzt ist.

Mit Client-SDK 5 erfolgt das RSA- und ECDSA-Hashing lokal, sodass es kein Datenlimit gibt. Bei HMAC gibt es ein Datenlimit. Weitere Informationen finden Sie in der Fußnote [2](#).

RSA

- CKM_RSA_X_509
- CKM_RSA_PKCS – Nur Single-Part-Operationen.
- CKM_RSA_PKCS_PSS – Nur Single-Part-Operationen.
- 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 – Nur Single-Part-Operationen.
- CKM_ECDSA_SHA1
- CKM_ECDSA_SHA224
- CKM_ECDSA_SHA256
- CKM_ECDSA_SHA384
- CKM_ECDSA_SHA512

HMAC

- CKM_SHA_1_HMAC²
- CKM_SHA224_HMAC²
- CKM_SHA256_HMAC²
- CKM_SHA384_HMAC²
- CKM_SHA512_HMAC²

CMAC

- CKM_AES_CMAC

Funktionen zur Signierung, Wiederherstellung und Überprüfung der Wiederherstellung

Das Client-SDK 5 unterstützt die Funktionen Sign Recover und Verify Recover nicht.

Digest-Funktionen

Mit der AWS CloudHSM Softwarebibliothek für PKCS #11 können Sie die folgenden Mechanismen für Digest-Funktionen verwenden. Mit Client-SDK 5 werden die Daten lokal in der Software gehasht. Das bedeutet, dass die Größe der Daten, die vom SDK gehasht werden können, unbegrenzt ist.

- CKM_SHA_1

- CKM_SHA224
- CKM_SHA256
- CKM_SHA384
- CKM_SHA512

Funktionen zum Verschlüsseln und Entschlüsseln

Mit der AWS CloudHSM Softwarebibliothek für PKCS #11 können Sie die folgenden Mechanismen für Verschlüsselungs- und Entschlüsselungsfunktionen verwenden.

- CKM_RSA_X_509
- CKM_RSA_PKCS – Nur Single-Part-Operationen. Bevorstehende Änderung in der Fußnote [5](#) aufgeführt.
- CKM_RSA_PKCS_OAEP – Nur Single-Part-Operationen.
- CKM_AES_ECB
- CKM_AES_CTR
- CKM_AES_CBC
- CKM_AES_CBC_PAD
- CKM_DES3_CBC – bevorstehende Änderung in der Fußnote [5](#) aufgeführt.
- CKM_DES3_ECB – bevorstehende Änderung in der Fußnote [5](#) aufgeführt.
- CKM_DES3_CBC_PAD – bevorstehende Änderung in der Fußnote [5](#) aufgeführt.
- CKM_AES_GCM [1](#), [2](#)
- CKM_CLOUDHSM_AES_GCM^{[3](#)}

Schlüsselfunktionen ableiten

Die AWS CloudHSM Softwarebibliothek für die PKCS #11 -Bibliothek unterstützt die folgenden wichtigen Ableitungsmechanismen:

- CKM_SP800_108_COUNTER_KDF
- CKM_ECDH1_DERIVE- Unterstützt die ECDH-Schlüsselableitung mit den folgenden herstellerdefinierten KDF-Typen: [6](#)
 - CKD_CLOUDHSM_X963_SHA1_KDF- X9.63 KDF mit SHA1 [7](#)

- CKD_CLOUDHSM_X963_SHA224_KDF- X9.63 KDF mit SHA224 [7](#)
- CKD_CLOUDHSM_X963_SHA256_KDF- X9.63 KDF mit SHA256 [7](#)
- CKD_CLOUDHSM_X963_SHA384_KDF- X9.63 KDF mit SHA384 [7](#)
- CKD_CLOUDHSM_X963_SHA512_KDF- X9.63 KDF mit SHA512 [7](#)

Funktionen „Packen“ und „Entpacken“

Die AWS CloudHSM Softwarebibliothek für die PKCS #11 -Bibliothek ermöglicht es Ihnen, die folgenden Mechanismen für die Funktionen Wrap und Unwrap zu verwenden.

Weitere Informationen zum AES-Packen mit Schlüssel finden Sie unter [AES Key Wrapping](#).

- CKM_RSA_PKCS – Nur Single-Part-Operationen. Bevorstehende Änderung in der Fußnote [5](#) aufgeführt.
- 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](#)

Maximale Datengröße für jeden Mechanismus

Die folgende Tabelle listet die maximale Datengröße für jeden Mechanismus auf:.

Maximale Datensatzgröße

Mechanismus	Maximale Datengröße in Bytes
CKM_SHA_1_HMAC	16288
CKM_SHA224_HMAC	16256

Mechanismus	Maximale Datengröße in Bytes
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

Anmerkungen zum Mechanismus

- [1] Bei AES-GCM-Verschlüsselungen akzeptiert HSM keine Initialisierungsvektor (IV)-Daten von der Anwendung. Sie müssen einen erzeugten IV verwenden. Das vom HSM bereitgestellte 12 Byte IV wird in den referenzierten Speicherbereich geschrieben, auf den das pIV-Element der von Ihnen bereitgestellten CK_GCM_PARAMS-Parameterstruktur zeigt. Um Missverständnissen vorzubeugen: Das PKCS#11-SDK in Version 1.1.1 und höher erzwingt, dass pIV auf einen auf Null zurückgesetzten Puffer verweist, wenn die AES-GCM-Verschlüsselung initialisiert wird.
- [2] Wenn der Datenpuffer die maximale Datengröße überschreitet, führt die Operation bei der Bearbeitung von Daten mittels eines der folgenden Mechanismen zu einem Fehler. Für diese Mechanismen muss die gesamte Datenverarbeitung innerhalb des HSM erfolgen. Informationen zur maximalen Datengröße für jeden Mechanismus finden Sie unter [Maximale Datengröße für jeden Mechanismus](#).
- [3] Anbieterdefinierter Mechanismus. Um die von anbieterdefinierten CloudHSM-Mechanismen zu verwenden, müssen PKCS #11 -Anwendungen während der Kompilierung `/opt/cloudhsm/include/pkcs11/pkcs11t.h` enthalten.

CKM_CLOUDHSM_AES_GCM: Dieser proprietäre Mechanismus ist eine programmatisch sicherere Alternative zum Standard CKM_AES_GCM. Er stellt die vom HSM generierte IV dem Chiffretext voran, anstatt sie zurück in die CK_GCM_PARAMS- Struktur zu schreiben, die während der Chiffrierinitialisierung bereitgestellt wird. Sie können diesen Mechanismus mit C_Encrypt-, C_WrapKey-, C_Decrypt- und C_UnwrapKey-Funktionen verwenden. Bei Verwendung dieses

Mechanismus muss die pIV-Variable in der CK_GCM_PARAMS-Struktur auf NULL gesetzt werden. Wenn Sie diesen Mechanismus mit C_Decrypt und C_UnwrapKey verwenden, wird erwartet, dass der IV dem Verschlüsselungstext vorangestellt wird, der entpackt werden soll.

CKM_CLOUDHSM_AES_KEY_WRAP_PKCS5_PAD: AES-Verschlüsselung mit PKCS #5 Padding.

CKM_CLOUDHSM_AES_KEY_WRAP_ZERO_PAD: AES-Verschlüsselung mit Zero Padding.

- [4] Die folgenden CK_MECHANISM_TYPE und CK_RSA_PKCS_MGF_TYPE werden als CK_RSA_PKCS_OAEP_PARAMS für CKM_RSA_PKCS_OAEP unterstützt:
 - CKM_SHA_1 mit CKG_MGF1_SHA1
 - CKM_SHA224 mit CKG_MGF1_SHA224
 - CKM_SHA256 mit CKG_MGF1_SHA256
 - CKM_SHA384 mit CKM_MGF1_SHA384
 - CKM_SHA512 mit CKM_MGF1_SHA512
- [5] Gemäß den NIST-Leitlinien ist dies für Cluster im FIPS-Modus nach 2023 nicht zulässig. Für Cluster im Nicht-FIPS-Modus ist dies auch nach 2023 zulässig. Details dazu finden Sie unter [FIPS-140-Konformität: Mechanismus 2024 nicht mehr unterstützt](#).
- [6] Vom Hersteller definierte Typen. Um herstellerdefinierte CloudHSM-Typen verwenden zu können, müssen PKCS #11 -Anwendungen bei der Kompilierung Folgendes enthalten `cloudhsm_pkcs11_vendor_defs.h`. Dies ist `/opt/cloudhsm/include/pkcs11/cloudhsm_pkcs11_vendor_defs.h` für Linux-basierte Plattformen und `C:\Program Files\Amazon\CloudHSM\include\pkcs11\cloudhsm_pkcs11_vendor_defs.h` für Windows-basierte Plattformen zu finden
- [7] Die wichtigsten Ableitungsfunktionen (KDFs) sind in der [NIST-Sonderveröffentlichung 800-56A](#) Revision 3 spezifiziert.

Unterstützte API-Operationen für die PKCS #11 -Bibliothek für das Client-SDK 5 AWS CloudHSM

Die PKCS #11 -Bibliothek unterstützt die folgenden PKCS #11 -API-Operationen für das AWS CloudHSM Client-SDK 5.

- 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

Schlüsselattribute in der PKCS #11 -Bibliothek für AWS CloudHSM das Client-SDK 5

Ein AWS CloudHSM Schlüsselobjekt kann ein öffentlicher, privater oder geheimer Schlüssel sein. Für ein Schlüsselobjekt genehmigte Aktionen werden durch Attribute angegeben. Attribute werden bei der Erstellung des Schlüsselobjekts bestimmt. Wenn Sie die PKCS #11 -Bibliothek für verwenden AWS CloudHSM, weisen wir Standardwerte zu, die im PKCS #11 -Standard spezifiziert sind.

AWS CloudHSM unterstützt nicht alle in der PKCS #11 -Spezifikation aufgeführten Attribute. Wir erfüllen die Spezifikation für alle von uns unterstützten Attribute. Diese Attribute sind in den entsprechenden Tabellen aufgeführt.

Kryptografische Funktionen wie C_CreateObject, C_GenerateKey, C_GenerateKeyPair, C_UnwrapKey und C_DeriveKey zum Erstellen, Ändern oder Kopieren von Objekten benötigen eine Attributvorlage für einen ihrer Parameter. Weitere Informationen zum Übertragen von Attributvorlagen während der Erstellung von Objekten finden Sie in den Beispielen unter [Generieren von Schlüsseln über die PKCS #11-Bibliothek](#).

Die folgenden Themen enthalten weitere Informationen zu AWS CloudHSM Schlüsselattributen.

Themen

- [PKCS #11 -Bibliotheksattributtabelle für AWS CloudHSM Client SDK 5](#)
- [Ändern der PKCS #11 -Bibliotheksattribute für das AWS CloudHSM Client-SDK 5](#)
- [Interpretieren der Fehlercodes der PKCS #11 -Bibliothek für das Client-SDK 5 AWS CloudHSM](#)

PKCS #11 -Bibliotheksattributtabelle für AWS CloudHSM Client SDK 5

Die PKCS #11 -Bibliothekstabellen für AWS CloudHSM enthalten eine Liste von Attributen, die sich je nach Schlüsseltyp unterscheiden. Sie gibt an, ob ein bestimmtes Attribut für einen bestimmten Schlüsseltyp unterstützt wird, wenn eine bestimmte kryptografische Funktion mit verwendet wird. AWS CloudHSM

Legende:

- ✓ gibt an, dass CloudHSM das Attribut für den spezifischen Schlüsseltyp unterstützt.
- ✗ gibt an, dass CloudHSM das Attribut für den spezifischen Schlüsseltyp nicht unterstützt.
- R gibt an, dass der Attributwert für den spezifischen Schlüsseltyp als schreibgeschützt festgelegt ist.
- S zeigt an, dass das Attribut von `GetAttributeValue` nicht gelesen werden kann, da hierbei Groß- und Kleinschreibung beachtet werden muss.
- Eine leere Zelle in der Spalte „Standardwert“ gibt an, dass dem Attribut kein spezifischer Standardwert zugewiesen ist.

GenerateKeyPair

Attribut	Schlüsseltyp				Default Value (Standardwert)
	EC privat	EC öffentlich	RSA privat	RSA öffentlich	
CKA_CLASS	✓	✓	✓	✓	

Attribut	Schlüsseltyp				Default Value (Standard wert)
CKA_KEY_TYPE	✓	✓	✓	✓	
CKA_LABEL	✓	✓	✓	✓	
CKA_ID	✓	✓	✓	✓	
CKA_LOCAL	R	R	R	R	True
CKA_TOKEN	✓	✓	✓	✓	False
CKA_PRIVATE	✓ ¹	✓ ¹	✓ ¹	✓ ¹	True
CKA_ENCRYPT	✗	✓	✗	✓	False
CKA_DECRYPT	✓	✗	✓	✗	Falsch
CKA_DERIVE	✓	✓	✓	✓	False
CKA_MODIFIABLE	✓	✓	✓	✓	True
CKA_DESTRUYABLE	✓	✓	✓	✓	True
CKA_SIGN	✓	✗	✓	✗	False

Attribut	Schlüsseltyp				Default Value (Standard wert)
CKA_SIGN_RECOVER	×	×	×	×	
CKA_VERIFY	×	✓	×	✓	Falsch
CKA_VERIFY_RECOVER	×	×	×	×	
CKA_WRAP	×	✓	×	✓	Falsch
CKA_WRAP_TEMPLATE	×	✓	×	✓	
CKA_TRUSTED	×	✓	×	✓	Falsch
CKA_WRAP_WITH_TRUSTED	✓	×	✓	×	Falsch
CKA_UNWRAP	✓	×	✓	×	False
CKA_UNWRAP_TEMPLATE	✓	×	✓	×	
CKA_SENSITIVE	✓ ¹	×	✓ ¹	×	True

Attribut	Schlüsseltyp				Default Value (Standard wert)
CKA_ALWAYS_SENSITIVE	R	✗	R	✗	
CKA_EXTRACTABLE	✓	✗	✓	✗	True
CKA_NEVER_EXTRACTABLE	R	✗	R	✗	
CKA_MODULUS	✗	✗	✗	✗	
CKA_MODULUS_BITS	✗	✗	✗	✓ ²	
CKA_PRIME_1	✗	✗	✗	✗	
CKA_PRIME_2	✗	✗	✗	✗	
CKA_COEFFICIENT	✗	✗	✗	✗	
CKA_EXPONENT_1	✗	✗	✗	✗	
CKA_EXPONENT_2	✗	✗	✗	✗	

Attribut	Schlüsseltyp				Default Value (Standard wert)
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

Attribut	Schlüsseltyp			Default Value (Standard wert)
	AES	DES3	Allgemeiner geheimer Schlüssel	

Attribut	Schlüsseltyp			Default Value (Standard wert)
CKA_CLASS		✓	✓	✓
CKA_KEY_T YPE		✓	✓	✓
CKA_LABEL		✓	✓	✓
CKA_ID		✓	✓	✓
CKA_LOCAL		R	R	R
CKA_TOKEN		✓	✓	✓
CKA_PRIVA TE		✓ ¹	✓ ¹	✓ ¹
CKA_ENCRY PT		✓	✓	✗
CKA_DECRY PT		✓	✓	✗
CKA_DERIV E		✓	✓	✓
CKA_MODIF IABLE		✓	✓	✓
CKA_DESTR OYABLE		✓	✓	✓
CKA_SIGN		✓	✓	✓
CKA_SIGN_ RECOVER		✗	✗	✗

Attribut	Schlüsseltyp				Default Value (Standard wert)
CKA_VERIFY		✓	✓	✓	True
CKA_VERIFY_RECOVER		✗	✗	✗	
CKA_WRAP		✓	✓	✗	False
CKA_WRAP_TEMPLATE		✓	✓	✗	
CKA_TRUSTED		✓	✓	✗	Falsch
CKA_WRAP_WITH_TRUSTED		✓	✓	✓	Falsch
CKA_UNWRAP		✓	✓	✗	False
CKA_UNWRAP_TEMPLATE		✓	✓	✗	
CKA_SENSITIVE		✓	✓	✓	True
CKA_ALWAYS_SENSITIVE		✗	✗	✗	
CKA_EXTRACTABLE		✓	✓	✓	True

Attribut	Schlüsseltyp			Default Value (Standard wert)
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	×	×	×	
CKA_EC_PARAMS	×	×	×	

Attribut	Schlüsseltyp			Default Value (Standard wert)
CKA_EC_POINT		✗	✗	✗
CKA_VALUE		✗	✗	✗
CKA_VALUE_LEN		✓ ²	✗	✓ ²
CKA_CHECK_VALUE		R	R	R

CreateObject

Attribut	Schlüsseltyp							Default Value (Standard wert)
	EC privat	EC öffentlich	RSA privat	RSA öffentlich	AES	DES3	Allgemeiner geheimer Schlüssel	
CKA_CLASS	✓ ²	✓ ²	✓ ²	✓ ²	✓ ²	✓ ²	✓ ²	
CKA_KEY_TYPE	✓ ²	✓ ²	✓ ²	✓ ²	✓ ²	✓ ²	✓ ²	
CKA_LABEL	✓	✓	✓	✓	✓	✓	✓	
CKA_ID	✓	✓	✓	✓	✓	✓	✓	

Attribut	Schlüsseltyp							Default Value (Standardwert)
CKA_LOCAL	R	R	R	R	R	R	R	Falsch
CKA_TOKEN	✓	✓	✓	✓	✓	✓	✓	False
CKA_PRIVATE	✓ ¹	✓ ¹	✓ ¹	✓ ¹	✓ ¹	✓ ¹	✓ ¹	True
CKA_ENCRYPT	✗	✗	✗	✓	✓	✓	✗	False
CKA_DECRYPT	✗	✗	✓	✗	✓	✓	✗	Falsch
CKA_DERIVE	✓	✓	✓	✓	✓	✓	✓	False
CKA_MODIFIABLE	✓	✓	✓	✓	✓	✓	✓	True
CKA_DESTROYABLE	✓	✓	✓	✓	✓	✓	✓	True
CKA_SIGN	✓	✗	✓	✗	✓	✓	✓	False
CKA_SIGN_RECOVER	✗	✗	✗	✗	✗	✗	✗	Falsch
CKA_VERIFY	✗	✓	✗	✓	✓	✓	✓	Falsch

Attribut	Schlüsseltyp							Default Value (Standardwert)
CKA_VERIFY_RECOVER	×	×	×	×	×	×	×	
CKA_WRAP	×	×	×	✓	✓	✓	×	Falsch
CKA_WRAP_TEMPLATE	×	✓	×	✓	✓	✓	×	
CKA_TRUSTED	×	✓	×	✓	✓	✓	×	Falsch
CKA_WRAP_WITH_TRUSTED	✓	×	✓	×	✓	✓	✓	Falsch
CKA_UNWRAP	×	×	✓	×	✓	✓	×	False
CKA_UNWRAP_TEMPLATE	✓	×	✓	×	✓	✓	×	
CKA_SENSITIVE	✓	×	✓	×	✓	✓	✓	True
CKA_ALWAYS_SENSITIVE	R	×	R	×	R	R	R	
CKA_EXTRACTABLE	✓	×	✓	×	✓	✓	✓	True

Attribut	Schlüsseltyp							Default Value (Standardwert)
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	×	×	×	

Attribut	Schlüsseltyp							Default Value (Standard wert)
CKA_EC_PA RAMS	✓ ²	✓ ²	✗	✗	✗	✗	✗	
CKA_EC_PO INT	✗	✓ ²	✗	✗	✗	✗	✗	
CKA_VALUE	✓ ²	✗	✗	✗	✓ ²	✓ ²	✓ ²	
CKA_VALUE _LEN	✗	✗	✗	✗	✗	✗	✗	
CKA_CHECK _VALUE	R	R	R	R	R	R	R	

UnwrapKey

Attribut	Schlüsseltyp					Default Value (Standard wert)
	EC privat	RSA privat	AES	DES3	Allgemeiner geheimer Schlüssel	
CKA_CLASS	✓ ²	✓ ²	✓ ²	✓ ²	✓ ²	
CKA_KEY_T YPE	✓ ²	✓ ²	✓ ²	✓ ²	✓ ²	

Attribut	Schlüsseltyp						Default Value (Standard wert)
CKA_LABEL		✓	✓	✓	✓	✓	
CKA_ID		✓	✓	✓	✓	✓	
CKA_LOCAL		R	R	R	R	R	Falsch
CKA_TOKEN		✓	✓	✓	✓	✓	False
CKA_PRIVATE		✓ ¹	✓ ¹	✓ ¹	✓ ¹	✓ ¹	True
CKA_ENCRYPT		✗	✗	✓	✓	✗	False
CKA_DECRYPT		✗	✓	✓	✓	✗	Falsch
CKA_DERIVE		✓	✓	✓	✓	✓	False
CKA_MODIFIABLE		✓	✓	✓	✓	✓	True
CKA_DESTROYABLE		✓	✓	✓	✓	✓	True
CKA_SIGN		✓	✓	✓	✓	✓	False
CKA_SIGN_RECOVER		✗	✗	✗	✗	✗	Falsch

Attribut	Schlüsseltyp						Default Value (Standard wert)
CKA_VERIFY		✗	✗	✓	✓	✓	Falsch
CKA_VERIFY_RECOVER		✗	✗	✗	✗	✗	
CKA_WRAP		✗	✗	✓	✓	✗	Falsch
CKA_UNWRAP		✗	✓	✓	✓	✗	False
CKA_SENSITIVE		✓	✓	✓	✓	✓	True
CKA_EXTRACTABLE		✓	✓	✓	✓	✓	True
CKA_NEVER_EXTRACTABLE		R	R	R	R	R	
CKA_ALWAYS_SENSITIVE		R	R	R	R	R	
CKA_MODULUS		✗	✗	✗	✗	✗	
CKA_MODULUS_BITS		✗	✗	✗	✗	✗	
CKA_PRIME_1		✗	✗	✗	✗	✗	

Attribut	Schlüsseltyp						Default Value (Standard wert)
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		×	×	×	×	×	
CKA_VALUE_LEN		×	×	×	×	×	
CKA_CHECK_VALUE		R	R	R	R	R	

DeriveKey

Attribut	Schlüsseltyp			Default Value (Standard wert)
	AES	DES3	Allgemein er geheimer Schlüssel	
CKA_CLASS	✓ ²	✓ ²	✓ ²	
CKA_KEY_T YPE	✓ ²	✓ ²	✓ ²	
CKA_LABEL	✓	✓	✓	
CKA_ID	✓	✓	✓	
CKA_LOCAL	R	R	R	True
CKA_TOKEN	✓	✓	✓	False
CKA_PRIVATE	✓ ¹	✓ ¹	✓ ¹	True
CKA_ENCRYPT	✓	✓	✗	False
CKA_DECRYPT	✓	✓	✗	Falsch
CKA_DERIVE	✓	✓	✓	False
CKA_MODIFI ABLE	✓	✓	✓	True
CKA_DESTROY ABLE	✓	✓	✓	True

Attribut	Schlüsseltyp				Default Value (Standard wert)
CKA_SIGN		✓	✓	✓	False
CKA_SIGN_ RECOVER		✗	✗	✗	
CKA_VERIF Y		✓	✓	✓	Falsch
CKA_VERIF Y_RECOVER		✗	✗	✗	
CKA_WRAP		✓	✓	✗	Falsch
CKA_UNWRA P		✓	✓	✗	Falsch
CKA_SENSI TIVE		R	R	R	Wahr
CKA_EXTRA CTABLE		✓	✓	✓	True
CKA_NEVER _EXTRACTA BLE		R	R	R	
CKA_ALWAY S_SENSITI VE		R	R	R	
CKA_MODUL US		✗	✗	✗	

Attribut	Schlüsseltyp			Default Value (Standard wert)
CKA_MODUL US_BITS	×	×	×	
CKA_PRIME _1	×	×	×	
CKA_PRIME _2	×	×	×	
CKA_COEFF ICIENT	×	×	×	
CKA_EXPON ENT_1	×	×	×	
CKA_EXPON ENT_2	×	×	×	
CKA_PRIVA TE_EXPONE NT	×	×	×	
CKA_PUBLI C_EXPONEN T	×	×	×	
CKA_EC_PA RAMS	×	×	×	
CKA_EC_PO INT	×	×	×	
CKA_VALUE	×	×	×	

Attribut	Schlüsseltyp			Default Value (Standard wert)
CKA_VALUE_LEN		✓ ²	✗	✓ ²
CKA_CHECK_VALUE		R	R	R

GetAttributeValue

Attribut	Schlüsseltyp						
	EC privat	EC öffentlic h	RSA privat	RSA öffentlic h	AES	DES3	Allgemein er geheimer Schlüssel
CKA_CLASS	✓	✓	✓	✓	✓	✓	✓
CKA_KEY_T YPE	✓	✓	✓	✓	✓	✓	✓
CKA_LABEL	✓	✓	✓	✓	✓	✓	✓
CKA_ID	✓	✓	✓	✓	✓	✓	✓
CKA_LOCAL	✓	✓	✓	✓	✓	✓	✓
CKA_TOKEN	✓	✓	✓	✓	✓	✓	✓
CKA_PRIVA TE	✓ ¹	✓ ¹	✓ ¹	✓ ¹	✓ ¹	✓ ¹	✓ ¹

Attribut	Schlüsseltyp						
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	✗	✓	✗	✓	✓	✓	✓
CKA_WRAP_WITH_TRUSTED	✓	✗	✓	✗	✓	✓	✓

Attribut	Schlüsseltyp						
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	×	×	×	×
CKA_COEFFICIENT	×	×	S	×	×	×	×

Attribut	Schlüsseltyp						
CKA_EXPONENT_1	×	×	S	×	×	×	×
CKA_EXPONENT_2	×	×	S	×	×	×	×
CKA_PRIVATE_EXPONENT	×	×	S	×	×	×	×
CKA_PUBLIC_EXPONENT	×	×	✓	✓	×	×	×
CKA_EC_PARAMS	✓	✓	×	×	×	×	×
CKA_EC_POINT	×	✓	×	×	×	×	×
CKA_VALUE	S	×	×	×	✓	✓	✓
CKA_VALUE_LEN	×	×	×	×	✓	×	✓
CKA_CHECK_VALUE	✓	✓	✓	✓	✓	✓	×

Anmerkungen zu Attributen

- [1] Dieses Attribut wird teilweise von der Firmware unterstützt und muss explizit nur auf den Standardwert festgelegt werden.
- [2] Obligatorisches Attribut.

Ändern der PKCS #11 -Bibliotheksattribute für das AWS CloudHSM Client-SDK 5

Einige PKCS #11 -Bibliotheksattribute für ein AWS CloudHSM Objekt können nach der Erstellung des Objekts geändert werden, andere nicht. Verwenden Sie den Befehl [key set-attribute in der CloudHSM-CLI, um Attribute](#) zu ändern. Sie können auch eine Liste von Attributen ableiten, indem Sie den Befehl [key list](#) in der CloudHSM-CLI verwenden.

Die folgende Liste zeigt Attribute, die nach der Erstellung eines Objekts geändert werden dürfen.

- CKA_LABEL
- CKA_TOKEN

Note

Änderungen sind nur zum Umwandeln eines Sitzungsschlüssels in einen Token-Schlüssel zulässig. Verwenden Sie den Befehl [key set-attribute](#) in der CloudHSM-CLI, um den Attributwert zu ändern.

- CKA_ENCRYPT
- CKA_DECRYPT
- CKA_SIGN
- CKA_VERIFY
- CKA_WRAP
- CKA_UNWRAP
- CKA_LABEL
- CKA_SENSITIVE
- CKA_DERIVE

Note

Dieses Attribut unterstützt die Schlüsselableitung. Es muss für alle öffentlichen Schlüssel `False` lauten und kann nicht auf `True` festgelegt werden. Für geheime und private EC-Schlüssel kann es auf `True` oder `False` eingestellt werden.

- CKA_TRUSTED

 Note

Dieses Attribut kann nur vom Verschlüsselungsverantwortlichen (CO) auf `True` oder `False` festgelegt werden.

- `CKA_WRAP_WITH_TRUSTED`

 Note

Wenden Sie dieses Attribut auf einen exportierbaren Datenschlüssel an, um anzugeben, dass Sie diesen Schlüssel nur mit Schlüsseln packen können, die als `CKA_TRUSTED` markiert sind. Sobald Sie `CKA_WRAP_WITH_TRUSTED` auf „true“ setzen, wird das Attribut schreibgeschützt und Sie können das Attribut nicht mehr ändern oder entfernen.

Interpretieren der Fehlercodes der PKCS #11 -Bibliothek für das Client-SDK 5 AWS CloudHSM

Die Angabe eines PKCS #11 -Bibliotheksattributs in der Vorlage, das von einem bestimmten Schlüssel nicht unterstützt wird, führt zu einem Fehler. Die folgende Tabelle enthält die Fehlercodes, die generiert werden, wenn Sie gegen die Spezifikationen verstoßen:

Fehlercode	Beschreibung
<code>CKR_TEMPLATE_INCONSISTENT</code>	Diese Fehlermeldung erhalten Sie, wenn Sie in der Attributvorlage ein Attribut angeben, das zwar der PKCS #11-Spezifikation entspricht, jedoch nicht von CloudHSM unterstützt wird.
<code>CKR_ATTRIBUTE_TYPE_INVALID</code>	Diese Fehlermeldung erhalten Sie, wenn Sie den Wert für ein Attribut abrufen, das zwar der PKCS #11-Spezifikation entspricht, jedoch nicht von CloudHSM unterstützt wird.
<code>CKR_ATTRIBUTE_INCOMPLETE</code>	Diese Fehlermeldung erhalten Sie, wenn Sie in der Attributvorlage das obligatorische Attribut nicht angeben.

Fehlercode	Beschreibung
CKR_ATTRIBUTE_READ_ONLY	Diese Fehlermeldung erhalten Sie, wenn Sie in der Attributvorlage ein schreibgeschütztes Attribut angeben.

Codebeispiele für die PKCS #11 -Bibliothek für das AWS CloudHSM Client-SDK 5

Die Codebeispiele auf dieser Seite GitHub zeigen Ihnen, wie Sie grundlegende Aufgaben mithilfe der PKCS #11 -Bibliothek für das AWS CloudHSM Client-SDK 5 ausführen.

Voraussetzungen

Bevor Sie die Beispiele ausführen, führen Sie die folgenden Schritte aus, um Ihre Umgebung einzurichten:

- Installieren und konfigurieren Sie die [PKCS #11-Bibliothek](#) für das Client-SDK 5.
- Richten Sie einen [Crypto-Benutzer \(CU\)](#) ein. Ihre Anwendung verwendet dieses HSM-Konto, um die Codebeispiele auf dem HSM auszuführen.

Codebeispiele

Codebeispiele für die AWS CloudHSM Softwarebibliothek für PKCS #11 finden Sie unter. [GitHub](#)
Dieses Repository enthält Beispiele für allgemeine Vorgänge mit PKCS #11, einschließlich Verschlüsselung, Entschlüsselung, Signieren und Verifizieren.

- [Schlüssel generieren \(AES, RSA, EC\)](#)
- [Schlüsselattribute auflisten](#)
- [Verschlüsseln und Entschlüsseln von Daten mit AES GCM](#)
- [Verschlüsseln und Entschlüsseln von Daten mit AES_CTR](#)
- [Verschlüsseln und Entschlüsseln von Daten mit 3DES](#)
- [Signieren und Verifizieren von Daten mit RSA](#)
- [Ableiten von Schlüsseln mit HMAC KDF](#)
- [Verpacken und Entpacken von Schlüsseln mit AES und PKCS #5 Padding](#)
- [Verpacken und Entpacken von Schlüsseln mit AES ohne Padding](#)

- [Verpacken und Entpacken von Schlüsseln mit AES und Zero Padding](#)
- [Packen und Entpacken von Schlüsseln mit AES-GCM](#)
- [Schlüssel mit RSA packen und entpacken](#)

Erweiterte Konfigurationen für die PKCS #11 -Bibliothek für AWS CloudHSM

Der AWS CloudHSM PKCS #11 -Anbieter umfasst die folgende erweiterte Konfiguration, die nicht Teil der allgemeinen Konfigurationen ist, die die meisten Kunden verwenden. Diese Konfigurationen bieten zusätzliche Funktionen.

- [Verbindung zu mehreren Steckplätzen mit PKCS #11](#)
- [Versuchen Sie erneut, die Konfiguration für PKCS #11 zu konfigurieren](#)

Konfiguration mit mehreren Steckplätzen mit PKCS #11 -Bibliothek für AWS CloudHSM

Ein einzelner Steckplatz in der PKCS #11-Bibliothek des Client-SDK 5 steht für eine einzelne Verbindung zu einem Cluster in AWS CloudHSM. Mit Client SDK 5 können Sie Ihre PKCS11 Bibliothek so konfigurieren, dass mehrere Slots Benutzer über eine einzige PKCS #11 -Anwendung mit mehreren CloudHSM-Clustern verbinden können.

Verwenden Sie die Anweisungen in diesem Thema, um dafür zu sorgen, dass Ihre Anwendung Multi-Slot-Funktionen verwendet, um eine Verbindung mit mehreren Clustern herzustellen.

Themen

- [Voraussetzungen für mehrere Steckplätze für die PKCS #11 -Bibliothek für AWS CloudHSM](#)
- [Konfigurieren Sie die PKCS #11 -Bibliothek für Multi-Slot-Funktionalität für AWS CloudHSM](#)
- [Fügen Sie einen Cluster mit Multi-Slot-Funktionalität hinzu für AWS CloudHSM](#)
- [Entfernen Sie einen Cluster mit Multi-Slot-Funktionalität für AWS CloudHSM](#)

Voraussetzungen für mehrere Steckplätze für die PKCS #11 -Bibliothek für AWS CloudHSM

Bevor Sie mehrere Steckplätze für die PKCS #11 -Bibliothek für konfigurieren AWS CloudHSM, müssen Sie die folgenden Voraussetzungen erfüllen.

- Zwei oder mehr AWS CloudHSM Cluster, zu denen Sie eine Verbindung herstellen möchten, zusammen mit ihren Cluster-Zertifikaten.

- Eine EC2-Instance mit Sicherheitsgruppen, die korrekt konfiguriert sind, um eine Verbindung zu allen oben genannten Clustern herzustellen. Weitere Informationen zum Einrichten eines Clusters und der Client-Instanz finden Sie unter [Erste Schritte mit AWS CloudHSM](#).
- Um die Multi-Steckplatz-Funktionalität einzurichten, müssen Sie die PKCS #11-Bibliothek bereits heruntergeladen und installiert haben. Wenn Sie dies noch nicht getan haben, lesen Sie die Anweisungen unter [???](#).

Konfigurieren Sie die PKCS #11 -Bibliothek für Multi-Slot-Funktionalität für AWS CloudHSM

Gehen Sie wie folgt vor, um Ihre PKCS #11 -Bibliothek für die Multi-Slot-Funktionalität zu konfigurieren: AWS CloudHSM

1. Identifizieren Sie die Cluster, zu denen Sie mithilfe der Multi-Steckplatz-Funktionalität eine Verbindung herstellen möchten.
2. Fügen Sie diese Cluster zu Ihrer PKCS #11-Konfiguration hinzu, indem Sie den Anweisungen unter [???](#) folgen.
3. Wenn Ihre PKCS #11-Anwendung das nächste Mal ausgeführt wird, wird sie über Multi-Steckplatz-Funktionen verfügen.

Fügen Sie einen Cluster mit Multi-Slot-Funktionalität hinzu für AWS CloudHSM

Wenn Sie [mit PKCS #11 for eine Verbindung zu mehreren Steckplätzen](#) herstellen AWS CloudHSM, verwenden Sie den `configure-pkcs11 add-cluster` Befehl, um Ihrer Konfiguration einen Cluster hinzuzufügen.

Syntax

```
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]
```


Beispiele

Fügen Sie mithilfe des **cluster-id**-Parameters einen Cluster hinzu

Example

Verwenden Sie den `configure-pkcs11 add-cluster` zusammen mit dem `cluster-id`-Parameter, um Ihrer Konfiguration einen Cluster (mit der ID von `cluster-1234567`) hinzuzufügen.

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

Wenn die Verwendung von `configure-pkcs11 add-cluster` mit dem `cluster-id`-Parameter nicht dazu führt, dass der Cluster hinzugefügt wird, finden Sie im folgenden Beispiel eine längere Version dieses Befehls, die auch die Parameter `--region` und `--endpoint` zur Identifizierung des hinzugefügten Clusters erfordert. Wenn zum Beispiel die Region des Clusters eine andere ist als die, die als Standard für Ihre AWS CLI konfiguriert ist, sollten Sie den `--region`-Parameter verwenden, um die richtige Region zu verwenden. Darüber hinaus haben Sie die Möglichkeit, den AWS CloudHSM API-Endpunkt anzugeben, der für den Anruf verwendet werden soll. Dies kann für verschiedene Netzwerkkonfigurationen erforderlich sein, z. B. für die Verwendung von VPC-Schnittstellenendpunkten, für die nicht den Standard-DNS-Hostnamen verwendet wird. AWS CloudHSM

Fügen Sie einen Cluster mit den Parametern **cluster-id**, **endpoint**, und **region** hinzu

Example

Verwenden Sie `configure-pkcs11 add-cluster` zusammen mit den Parametern `cluster-id`, `endpoint` und `region`, um Ihrer Konfiguration einen Cluster (mit der ID von `cluster-1234567`) hinzuzufügen.

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>
```

Weitere Hinweise zu den Parametern `--cluster-id`, `--region` und `--endpoint` finden Sie unter [the section called “Parameters”](#).

Parameters

`--cluster-id` **<Cluster ID>**

Führt einen `DescribeClusters`-Aufruf aus, um alle IP-Adressen der HSM-Elastic-Network-Schnittstelle (ENI) im Cluster mit der Cluster-ID zu finden. Das System fügt die ENI-IP-Adressen zu den AWS CloudHSM Konfigurationsdateien hinzu.

Note

Wenn Sie den `--cluster-id` Parameter von einer EC2-Instance innerhalb einer VPC verwenden, die keinen Zugriff auf das öffentliche Internet hat, müssen Sie einen VPC-Schnittstellen-Endpunkt erstellen, mit dem Sie eine Verbindung herstellen können. AWS CloudHSM Weitere Informationen über VPC-Endpunkte finden Sie unter [???](#).

Erforderlich: Ja

`--endpoint` **<Endpoint>**

Geben Sie den AWS CloudHSM API-Endpunkt an, der für den `DescribeClusters` Aufruf verwendet wird. Sie müssen diese Option in Kombination mit `--cluster-id` festlegen.

Erforderlich: Nein

`--hsm-ca-cert` *<HsmCA Certificate Filepath>*

Gibt den Dateipfad zum HSM-CA-Zertifikat an.

Erforderlich: Nein

`--region` *<Region>*

Geben Sie die Region Ihres Clusters an. Sie müssen diese Option in Kombination mit `--cluster-id` festlegen.

Wenn Sie den `--region`-Parameter nicht angeben, wählt das System die Region aus, indem es versucht, die Umgebungsvariablen `AWS_DEFAULT_REGION` oder `AWS_REGION` zu lesen. Wenn diese Variablen nicht festgelegt sind, überprüft das System die Region, die Ihrem Profil in Ihrer AWS-Config-Datei zugeordnet ist (normalerweise `~/.aws/config`), sofern Sie in der `AWS_CONFIG_FILE`-Umgebungsvariable keine andere Datei angegeben haben. Wenn keine der oben genannten Optionen festgelegt ist, verwendet das System standardmäßig die `us-east-1`-Region.

Erforderlich: Nein

`--client-cert-hsm-tls` *<client certificate hsm tls path>*

Pfad zum Client-Zertifikat, das für die gegenseitige Authentifizierung zwischen TLS Client und HSM verwendet wird.

Verwenden Sie diese Option nur, wenn Sie mindestens einen Vertrauensanker auf HSM mit CloudHSM CLI registriert haben. Sie müssen diese Option in Kombination mit `--client-key-hsm-tls-file` festlegen.

Erforderlich: Nein

`--client-key-hsm-tls` *<client key hsm tls path>*

Pfad zum Client-Schlüssel, der für die gegenseitige Authentifizierung zwischen TLS Client und HSM verwendet wird.

Verwenden Sie diese Option nur, wenn Sie mindestens einen Vertrauensanker auf HSM mit CloudHSM CLI registriert haben. Sie müssen diese Option in Kombination mit `--client-cert-hsm-tls-file` festlegen.

Erforderlich: Nein

Entfernen Sie einen Cluster mit Multi-Slot-Funktionalität für AWS CloudHSM

Wenn Sie [mit PKCS #11 eine Verbindung zu mehreren Steckplätzen herstellen](#), verwenden Sie den `configure-pkcs11 remove-cluster`-Befehl, um einen Cluster aus den verfügbaren PKCS #11-Steckplätzen zu entfernen.

Syntax

```
configure-pkcs11 remove-cluster [OPTIONS]  
    --cluster-id <CLUSTER ID>  
    [-h, --help]
```

Beispiele

Entfernen Sie mithilfe des **cluster-id**-Parameters einen Cluster

Example

Verwenden Sie den `configure-pkcs11 remove-cluster` zusammen mit dem `cluster-id`-Parameter, um aus Ihrer Konfiguration einen Cluster (mit der ID von `cluster-1234567`) zu entfernen.

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>
```

Weitere Informationen zum Parameter `--cluster-id` erhalten Sie unter [the section called "Parameters"](#).

Parameter

--cluster-id **<Cluster ID>**

Die ID des Clusters, der aus der Konfiguration entfernt werden soll

Erforderlich: Ja

Wiederholen Sie die Befehle für die PKCS #11 -Bibliothek für AWS CloudHSM

AWS CloudHSM Das Client-SDK 5.8.0 und höher verfügen über eine integrierte automatische Wiederholungsstrategie, mit der HSM-gedrosselte Operationen von der Clientseite aus wiederholt werden. Wenn ein HSM Operationen drosselt, weil es mit der Ausführung früherer Operationen zu beschäftigt ist und keine weiteren Anfragen annehmen kann, versucht der Client, gedrosselte Operationen bis zu dreimal zu wiederholen, während er SDKs sich exponentiell zurückzieht. Diese automatische Wiederholungsstrategie kann auf einen von zwei Modi eingestellt werden: aus und Standard.

- aus: Das Client-SDK führt bei gedrosselten Vorgängen durch das HSM keine Wiederholungsstrategie durch.
- Standard: Dies ist der Standardmodus für das Client-SDK 5.8.0 und höher. In diesem Modus wiederholt der Client automatisch gedrosselte Operationen, indem er SDKs sich exponentiell zurückzieht.

Weitere Informationen finden Sie unter [HSM-Drosselung](#).

Stellen Sie den Modus für Wiederholungsbefehle auf Aus

Linux

So setzen Sie Wiederholungsbefehle auf off für Client-SDK 5 unter Linux

- Sie können den folgenden Befehl verwenden, um die Konfiguration der Wiederholungen auf den off-Modus zu setzen:

```
$ sudo /opt/cloudhsm/bin/configure-pkcs11 --default-retry-mode off
```

Windows

So setzen Sie Wiederholungsbefehle auf off für Client-SDK 5 unter Windows

- Sie können den folgenden Befehl verwenden, um die Konfiguration der Wiederholungen auf den off-Modus zu setzen:

```
PS C:\> & "C:\Program Files\Amazon\CloudHSM\bin\configure-pkcs11.exe" --default-retry-mode off
```

Speicherung von Zertifikaten mit der PKCS #11 -Bibliothek

Die AWS CloudHSM PKCS #11 -Bibliothek unterstützt das Speichern von Zertifikaten mit öffentlichen Schlüsseln als „öffentliche Objekte“ (wie in PKCS #11 2.40 definiert) auf hsm2m.medium-Clustern. Diese Funktion ermöglicht sowohl öffentlichen als auch privaten PKCS #11 -Sitzungen das Erstellen, Abrufen, Ändern und Löschen von Zertifikaten mit öffentlichem Schlüssel.

Um den Zertifikatsspeicher mit der PKCS #11 -Bibliothek zu verwenden, müssen Sie ihn in Ihrer Client-Konfiguration aktivieren. Nach der Aktivierung können Sie Zertifikatsobjekte von Ihren PKCS #11 -Anwendungen aus verwalten. Bei Vorgängen, die sowohl für Zertifikate als auch für Schlüsselobjekte gelten, z. B. [C_FindObjects](#), werden Ergebnisse sowohl aus dem Schlüssel- als auch aus dem Zertifikatsspeicher zurückgegeben.

Themen

- [Zertifikatsspeicher aktivieren](#)
- [API-Operationen zur Speicherung von Zertifikaten](#)
- [Speicherattribute für Zertifikate](#)
- [Auditprotokolle für die Speicherung von Zertifikaten](#)

Zertifikatsspeicher aktivieren

Sie können die Zertifikatsspeicherung auf hsm2m.medium-Clustern mithilfe des Bibliothekskonfigurationstools PKCS #11 aktivieren. Diese Funktion ist in den SDK-Versionen 5.13 und höher verfügbar. Eine Liste der Operationen, die den Zertifikatsobjekttyp unterstützen, finden Sie unter [API-Operationen zur Speicherung von Zertifikaten](#).

Gehen Sie für Ihr Betriebssystem wie folgt vor, um die Zertifikatsspeicherung zu aktivieren:

Linux

- Aktivieren Sie den Zertifikatsspeicher

Führen Sie den folgenden Befehl aus:

```
$ sudo /opt/cloudhsm/bin/configure-pkcs11 --enable-certificate-storage
```

Windows

- Aktivieren Sie den Zertifikatsspeicher

Öffnen Sie eine Eingabeaufforderung und führen Sie den folgenden Befehl aus:

```
PS C:\> & "C:\Program Files\Amazon\CloudHSM\bin\configure-pkcs11.exe" --enable-certificate-storage
```

API-Operationen zur Speicherung von Zertifikaten

Die folgenden PKCS #11 -Operationen unterstützen den Zertifikatsobjekttyp (CKO_CERTIFICATE):

Allgemeine Zertifikatsoperationen

C_CreateObject

Erstellt ein neues Zertifikatsobjekt.

C_DestroyObject

Löscht ein vorhandenes Zertifikatsobjekt.

C_GetAttributeValue

Ruft den Wert eines oder mehrerer Attribute eines Zertifikatsobjekts ab.

C_SetAttributeValue

Aktualisiert den Wert eines oder mehrerer Attribute eines Zertifikatsobjekts.

Suchvorgänge für Zertifikatsobjekte

C_FindObjectsInit

Startet eine Suche nach Zertifikatsobjekten.

C_FindObjects

Setzt die Suche nach Zertifikatsobjekten fort.

C_FindObjectsFinal

Beendet die Suche nach Zertifikatsobjekten.

Speicherattribute für Zertifikate

In der folgenden Tabelle sind die unterstützten Zertifikatsobjektattribute und ihre Werte aufgeführt:

Attribut	Standardwert	Description
CKA_CLASS	Erforderlich	Der Wert muss CKO_CERTIFICATE sein.
CKA_TOKEN	Wahr	Der Wert muss True sein.
CKA_MODIFIABLE	Wahr	Der Wert muss True sein.
CKA_PRIVATE	Falsch	Der Wert muss False sein.
CKA_LABEL	Leer	Beschränkt auf 127 Zeichen.
CKA_COPYABLE	Falsch	Der Wert muss False sein.
CKA_DESTROYABLE	Wahr	Der Wert muss True sein.
CKA_CERTIFICATE_TYPE	Erforderlich	Der Wert muss CKC_X_509 sein.
CKA_TRUSTED	Falsch	Der Wert muss False sein.
CKA_CERTIFICATE_CATEGORY	CK_CERTIFICATE_CATEGORY_UNSPECIFIED	Der Wert muss CK_CERTIFICATE_CATEGORY_UNSPECIFIED sein.

Attribut	Standardwert	Description
	EGORY_UNSPECIFIED	
CKA_CHECK_VALUE	Abgeleitet von CKA_VALUE	Automatisch festgelegt basierend auf CKA_VALUE .
CKA_START_DATE	Leer	Das Datum „nicht vor“ des Zertifikats.
CKA_END_DATE	Leer	Das Datum „nicht danach“ des Zertifikats.
CKA_PUBLIC_KEY_INFO	Leer	Die maximale Größe beträgt 16 Kilobyte.
CKA_SUBJECT	Erforderlich	Der Betreff des Zertifikats.
CKA_ID	Leer	Die maximale Größe beträgt 128 Byte. Eindeutigkeit wird nicht erzwungen.
CKA_ISSUER	Leer	Der Zertifikatsaussteller.
CKA_SERIAL_NUMBER	Leer	Die Seriennummer des Zertifikats.
CKA_VALUE	Erforderlich	Die maximale Größe beträgt 32 Kilobyte.

Auditprotokolle für die Speicherung von Zertifikaten

AWS CloudHSM schreibt Prüfprotokolle für Zertifikatsspeichervorgänge, die Daten ändern, in einen separaten Amazon CloudWatch Events-Protokollstream innerhalb der CloudWatch Protokollgruppe Ihres Clusters. Dieser Protokollstream ist nach dem Cluster benannt, nicht nach einem bestimmten HSM innerhalb des Clusters.

Hinweise zum Zugriff auf Audit-Logs in finden Sie CloudWatch unter [Arbeiten mit Amazon CloudWatch Logs und AWS CloudHSM Audit Logs](#).

Felder für Protokolleinträge

object_handle

Die eindeutige Kennung des Zertifikatsobjekts.

op_code

Der Vorgang wurde ausgeführt oder versucht. Mögliche Werte:

- CreateObject
- DestroyObject
- SetAttributeValues

response

OK wenn der Vorgang erfolgreich war oder einer der folgenden Fehlertypen vorliegt:

- DuplicateAttribute
- InvalidAttributeValue
- ObjectNotFound
- MaxObjectsReached
- InternalFailure

attributes

Die Attribute wurden, falls vorhanden, geändert.

timestamp

Die Zeit, zu der die Operation stattfand, in Millisekunden seit der Unix-Epoche.

Prüfungsprotokollbeispiele

CreateObject Beispiel

```
{
  "object_handle": 463180677312929947,
  "op_code": "CreateObject",
  "response": "OK",
  "attributes": null,
  "timestamp": 1725482483671
}
```

DestroyObject Beispiel

```
{
  "object_handle": 463180677312929947,
  "op_code": "DestroyObject",
  "response": "OK",
  "attributes": null,
  "timestamp": 1725482484559
}
```

SetAttributeValues Beispiel

```
{
  "object_handle": 463180678453346687,
  "op_code": "SetAttributeValues",
  "response": "OK",
  "attributes": [
    "Label"
  ],
  "timestamp": 1725482488004
}
```

Erfolgleses CreateObject Beispiel

```
{
  "object_handle": null,
  "op_code": "CreateObject",
  "response": "MaxObjectsReached",
  "attributes": null,
  "timestamp": 1726084937125
}
```

OpenSSL Dynamic Engine für AWS CloudHSM Client SDK 5

Mit der AWS CloudHSM OpenSSL Dynamic Engine können Sie kryptografische Operationen über die OpenSSL-API auf Ihren CloudHSM-Cluster auslagern.

AWS CloudHSM stellt eine OpenSSL Dynamic Engine bereit, über die Sie in [AWS CloudHSM SSL/TLS-Offload unter Linux mit Tomcat mit JSSE](#) oder [AWS CloudHSM SSL/TLS-Offload unter Linux mit NGINX oder Apache mit OpenSSL](#) nachlesen können. Ein Beispiel zur Verwendung

AWS CloudHSM mit OpenSSL finden Sie in [diesem AWS-Sicherheitsblog](#). Informationen zur Plattformunterstützung für finden Sie SDKs unter [the section called “Unterstützte Plattformen”](#). Informationen zur Problembehandlung finden Sie unter [Bekannte Probleme mit der OpenSSL Dynamic Engine für AWS CloudHSM](#).

Verwenden Sie die folgenden Abschnitte, um die AWS CloudHSM dynamische Engine für OpenSSL mithilfe des Client SDK 5 zu installieren und zu konfigurieren.

Informationen zur Verwendung von Client-SDK 3 finden Sie unter [Verwenden der vorherigen SDK-Version zum Arbeiten mit AWS CloudHSM](#).

Themen

- [Installieren Sie die OpenSSL Dynamic Engine für AWS CloudHSM Client SDK 5](#)
- [Unterstützte Schlüsseltypen für OpenSSL Dynamic Engine for AWS CloudHSM Client SDK 5](#)
- [Unterstützte Mechanismen für OpenSSL Dynamic Engine für AWS CloudHSM Client SDK 5](#)
- [Erweiterte Konfigurationen für OpenSSL für AWS CloudHSM](#)

Installieren Sie die OpenSSL Dynamic Engine für AWS CloudHSM Client SDK 5

Verwenden Sie die folgenden Abschnitte, um die OpenSSL Dynamic Engine for AWS CloudHSM Client SDK 5 zu installieren.

Note

Um einen einzelnen HSM-Cluster mit Client-SDK 5 auszuführen, müssen Sie zunächst die Einstellungen für die Haltbarkeit von Client-Schlüsseln verwalten, indem Sie die Einstellung `disable_key_availability_check` auf `True` festlegen. Weitere Informationen finden Sie unter [Schlüsselsynchronisierung](#) und [Client-SDK-5-Configure-Tool](#).

Zum Installieren und Konfigurieren von OpenSSL Dynamic Engine

1. Verwenden Sie die folgenden Befehle zum Herunterladen und Installieren der OpenSSL-Engine.

Amazon Linux 2023

Installieren Sie die OpenSSL Dynamic Engine für Amazon Linux 2023 auf einer x86_64-Architektur:

```
$ 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
```

Installieren Sie die OpenSSL Dynamic Engine für Amazon Linux 2023 auf der ARM64 Architektur:

```
$ 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

Installieren Sie die OpenSSL Dynamic Engine für Amazon Linux 2 auf einer x86_64-Architektur:

```
$ 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
```

Installieren Sie die OpenSSL Dynamic Engine für Amazon Linux 2 auf der ARM64 Architektur:

```
$ 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+)

Installieren Sie die OpenSSL Dynamic Engine für RHEL 9 auf einer x86_64-Architektur:

```
$ 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
```

Installieren Sie die OpenSSL Dynamic Engine für RHEL 9 auf der ARM64 Architektur:

```
$ 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+)

Installieren Sie die OpenSSL Dynamic Engine für RHEL 8 auf einer x86_64-Architektur:

```
$ 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
```

Installieren Sie die OpenSSL Dynamic Engine für RHEL 8 auf der ARM64 Architektur:

```
$ 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 24.04 LTS

Installieren Sie die OpenSSL Dynamic Engine für Ubuntu 24.04 LTS auf der x86_64-Architektur:

```
$ 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
```

Installieren Sie die OpenSSL Dynamic Engine für Ubuntu 24.04 LTS auf der Architektur: ARM64

```
$ 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

Installieren Sie die OpenSSL Dynamic Engine für Ubuntu 22.04 LTS auf der x86_64-Architektur:

```
$ 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
```

Installieren Sie die OpenSSL Dynamic Engine für Ubuntu 22.04 LTS auf der Architektur: ARM64

```
$ 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
```

Sie haben die gemeinsam genutzte Bibliothek für die Dynamic Engine unter `/opt/cloudhsm/lib/libcloudhsm_openssl_engine.so` installiert.

2. Bootstrap für Client-SDK 5 Weitere Informationen zu Bootstrapping finden Sie unter [Bootstrap für das Client-SDK](#).
3. Legen Sie eine Umgebungsvariable mit den Anmeldeinformationen eines Crypto-Benutzers (CU) fest. Hinweise zum Erstellen CUs finden Sie unter. [the section called “Erstellen”](#)

```
$ export CLOUDHSM_PIN=<HSM user name>:<password>
```

 Note

Das Client-SDK 5 führt die CLOUDHSM_PIN-Umgebungsvariable zum Speichern der Anmeldeinformationen der CU ein. Im Client-SDK 3 speichern Sie die CU-Anmeldeinformationen in der n3fips_password-Umgebungsvariable. Client-SDK 5 unterstützt beide Umgebungsvariablen, wir empfehlen jedoch, CLOUDHSM_PIN zu verwenden.

Beim Setzen von CLOUDHSM_PIN Umgebungsvariablen müssen Sie alle Sonderzeichen maskieren, die von Ihrer Shell interpretiert werden könnten.

4. Verbinden Sie Ihre Installation von OpenSSL Dynamic Engine mit dem Cluster. Weitere Informationen finden Sie unter [Verbinden mit dem Cluster](#).
5. Bootstrap für das Client-SDK 5. Weitere Informationen finden Sie unter [the section called “Bootstrap für das Client-SDK”](#).

Überprüfen Sie die OpenSSL Dynamic Engine für Client-SDK 5

Verwenden Sie den folgenden Befehl, um Ihre Installation der OpenSSL Dynamic Engine zu überprüfen.

```
$ openssl engine -t cloudhsm
```

Die folgende Ausgabe verifiziert Ihre Konfiguration:

```
(cloudhsm) CloudHSM OpenSSL Engine  
[ available ]
```

Unterstützte Schlüsseltypen für OpenSSL Dynamic Engine for AWS CloudHSM Client SDK 5

Die AWS CloudHSM OpenSSL Dynamic Engine unterstützt die folgenden Schlüsseltypen mit Client SDK 5.

Schlüsseltyp	Description
EC	ECDSA sign/verify für die Schlüsseltypen P-256, P-384 und secp256k1. Informationen zum Generieren von EC-Schlüsseln, die mit der OpenSSL-Engine kompatibel sind, finden Sie unter Exportieren Sie einen asymmetrischen Schlüssel mit CloudHSM CLI .
RSA	Generierung von RSA-Schlüsseln für 2048-, 3072- und 4096-Bit-Schlüssel. RSA-Signieren/Verifizieren. Die Überprüfung wird auf die OpenSSL-Software übertragen.

Unterstützte Mechanismen für OpenSSL Dynamic Engine für AWS CloudHSM Client SDK 5

Die AWS CloudHSM OpenSSL Dynamic Engine unterstützt die folgenden Mechanismen für Sign- und Verify-Funktionen mit Client SDK 5.

Funktionen zum Signieren und Überprüfen

Mit Client-SDK 5 werden die Daten lokal in der Software gehasht. Das bedeutet, dass die Größe der Daten, die gehasht werden können, unbegrenzt ist.

RSA-Signaturtypen

- SHA1withRSA
- SHA224withRSA
- SHA256withRSA
- SHA384withRSA
- SHA512withRSA

ECDSA-Signaturtypen

- SHA1withECDSA

- SHA224withECDSA
- SHA256withECDSA
- SHA384withECDSA
- SHA512withECDSA

Erweiterte Konfigurationen für OpenSSL für AWS CloudHSM

Der AWS CloudHSM OpenSSL-Anbieter beinhaltet die folgende erweiterte Konfiguration, die nicht Teil der allgemeinen Konfigurationen ist, die die meisten Kunden verwenden. Diese Konfigurationen bieten zusätzliche Funktionen.

- [Wiederholungsbefehle für OpenSSL](#)

Befehle für OpenSSL wiederholen für AWS CloudHSM

AWS CloudHSM Das Client-SDK 5.8.0 und höher verfügen über eine integrierte automatische Wiederholungsstrategie, mit der HSM-gedrosselte Operationen von der Clientseite aus wiederholt werden. Wenn ein HSM Operationen drosselt, weil es mit der Ausführung früherer Operationen zu beschäftigt ist und keine weiteren Anfragen annehmen kann, versucht der Client, gedrosselte Operationen bis zu dreimal zu wiederholen, während er SDKs sich exponentiell zurückzieht. Diese automatische Wiederholungsstrategie kann auf einen von zwei Modi eingestellt werden: aus und Standard.

- aus: Das Client-SDK führt bei gedrosselten Vorgängen durch das HSM keine Wiederholungsstrategie durch.
- Standard: Dies ist der Standardmodus für das Client-SDK 5.8.0 und höher. In diesem Modus wiederholt der Client automatisch gedrosselte Operationen, indem er SDKs sich exponentiell zurückzieht.

Weitere Informationen finden Sie unter [HSM-Drosselung](#).

Stellen Sie den Modus für Wiederholungsbefehle auf Aus

Sie können den folgenden Befehl verwenden, um Wiederholungsbefehle in den off-Modus zu setzen:

```
$ sudo /opt/cloudhsm/bin/configure-dyn --default-retry-mode off
```

OpenSSL-Anbieter für AWS CloudHSM Client SDK 5

Mit dem AWS CloudHSM OpenSSL Provider können Sie kryptografische TLS-Operationen über die OpenSSL Provider-API auf Ihren CloudHSM-Cluster auslagern. Die Provider-Schnittstelle ist der empfohlene Ansatz für neue Bereitstellungen mit OpenSSL 3.2 und höher.

Verwenden Sie die folgenden Abschnitte, um den AWS CloudHSM OpenSSL Provider mithilfe des Client SDK 5 zu installieren und zu konfigurieren.

Warning

Die Integration mit OpenSSL CLI wird derzeit vom AWS CloudHSM OpenSSL Provider nicht unterstützt. Informationen zu unterstützten [the section called “Offload auf Linux mit OpenSSL Provider”](#) Integrationen finden Sie unter.

Unterstützte Plattformen

Der OpenSSL Provider benötigt OpenSSL 3.2 oder höher, verfügbar auf EL9 +, Ubuntu 24.04+ und Amazon Linux 2023+.

Überprüfen Sie die Kompatibilität: `openssl version`

Topics

- [Installieren Sie den OpenSSL Provider für AWS CloudHSM Client SDK 5](#)
- [Unterstützte Schlüsseltypen für OpenSSL Provider for AWS CloudHSM Client SDK 5](#)
- [Von OpenSSL Providern unterstützte Mechanismen](#)
- [Erweiterte Konfiguration des OpenSSL-Anbieters](#)

Installieren Sie den OpenSSL Provider für AWS CloudHSM Client SDK 5

Verwenden Sie die folgenden Abschnitte, um den OpenSSL Provider for AWS CloudHSM Client SDK 5 zu installieren.

Note

Um einen einzelnen HSM-Cluster mit Client-SDK 5 auszuführen, müssen Sie zunächst die Einstellungen für die Haltbarkeit von Client-Schlüsseln verwalten, indem Sie die Einstellung

`disable_key_availability_check` auf `True` festlegen. Weitere Informationen finden Sie unter [Schlüsselsynchronisierung](#) und [Client-SDK-5-Configure-Tool](#).

Voraussetzungen

Der OpenSSL Provider benötigt die Clustertypen `hsm2m.medium` und mindestens CloudHSM Client SDK Version 5.17.0 oder höher.

Installieren Sie den OpenSSL Provider

Um den OpenSSL Provider zu installieren

1. Verwenden Sie die folgenden Befehle, um den OpenSSL Provider herunterzuladen und zu installieren.

Amazon Linux 2023

Installieren Sie den OpenSSL Provider für Amazon Linux 2023 auf der `x86_64`-Architektur:

```
$ 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
```

Installieren Sie den OpenSSL Provider für Amazon Linux 2023 auf der `ARM64` Architektur:

```
$ 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+)

Installieren Sie den OpenSSL Provider für RHEL 9 auf einer `x86_64`-Architektur:

```
$ 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
```

Installieren Sie den OpenSSL Provider für RHEL 9 auf der ARM64 Architektur:

```
$ 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+)

Installieren Sie den OpenSSL Provider für RHEL 10 auf einer x86_64-Architektur:

```
$ 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
```

Installieren Sie den OpenSSL Provider für RHEL 10 auf der ARM64 Architektur:

```
$ 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 24.04

Installieren Sie den OpenSSL Provider für Ubuntu 24.04 auf der x86_64-Architektur:

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/Noble/cloudhsm-openssl-provider_latest_u24.04_amd64.deb
```

```
$ sudo dpkg -i ./cloudhsm-openssl-provider_latest_u24.04_amd64.deb
```

Installieren Sie den OpenSSL Provider für Ubuntu 24.04 auf der Architektur: ARM64

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/Noble/  
cloudhsm-openssl-provider_latest_u24.04_arm64.deb
```

```
$ sudo dpkg -i ./cloudhsm-openssl-provider_latest_u24.04_arm64.deb
```

Sie haben die Shared Library für den OpenSSL Provider unter `/opt/cloudhsm/lib/licloudhsm_openssl_provider.so` installiert.

2. Bootstrap für Client-SDK 5 Weitere Informationen zu Bootstrapping finden Sie unter [Bootstrap für das Client-SDK](#).
3. Stellen Sie die `CLOUDHSM_PIN` Umgebungsvariable mit Ihren Crypto-Benutzeranmeldedaten (CU) ein:

```
$ export CLOUDHSM_PIN=<username>:<password>
```

4. Connect Sie Ihre Installation von OpenSSL Provider mit dem Cluster. Weitere Informationen finden Sie unter [Verbinden mit dem Cluster](#).

Überprüfen der Installation

Stellen Sie sicher, dass der OpenSSL Provider korrekt installiert ist:

```
$ CLOUDHSM_PIN=<username>:<password> openssl list -providers -provider cloudhsm
```

Sie sollten eine Ausgabe sehen, die der folgenden ähnelt:

```
Providers:  
cloudhsm  
  name: AWS CloudHSM OpenSSL Provider  
  version: 5.17.0  
  status: active  
default  
  name: OpenSSL Default Provider  
  version: 3.2.2  
  status: active
```

Unterstützte Schlüsseltypen für OpenSSL Provider for AWS CloudHSM Client SDK 5

Der AWS CloudHSM OpenSSL Provider unterstützt die folgenden Schlüsseltypen mit Client SDK 5.

Schlüsseltyp	Description
RSA	RSA sign/verify - und asymmetrische Verschlüsselungsoperationen. Die Überprüfung wird auf die OpenSSL-Software übertragen. Informationen zum Generieren von RSA-Schlüsseln, die mit dem OpenSSL Provider kompatibel sind, finden Sie unter. Exportieren Sie einen asymmetrischen Schlüssel mit CloudHSM CLI
EC	ECDSA sign/verify für P-256-, P-384- und P-521-Kurven. Die Überprüfung wird auf die OpenSSL-Software übertragen. Informationen zum Generieren von EC-Schlüsseln, die mit dem OpenSSL Provider kompatibel sind, finden Sie unter. Exportieren Sie einen asymmetrischen Schlüssel mit CloudHSM CLI

Von OpenSSL Providern unterstützte Mechanismen

Das AWS CloudHSM OpenSSL Provider SDK unterstützt einen umfassenden Satz kryptografischer Mechanismen für verschiedene Operationen, darunter digitale Signaturen, asymmetrische Verschlüsselung, symmetrische Verschlüsselung, Schlüsselaustausch und mehr.

RSA-Signaturtypen

Der OpenSSL Provider unterstützt digitale RSA-Signaturen mit mehreren Hash-Algorithmen und Padding-Schemata:

SHA1withRSA

RSA-Signaturen mit dem SHA-1-Hash-Algorithmus

- PKCS #1 v1.5-Polsterung
- PSS-Polsterung (Probabilistisches Signaturschema)

SHA224withRSA

RSA-Signaturen mit dem SHA-224-Hash-Algorithmus

- PKCS #1 v1.5-Polsterung
- PSS-Polsterung

SHA256withRSA

RSA-Signaturen mit dem SHA-256-Hash-Algorithmus

- PKCS #1 v1.5-Polsterung
- PSS-Polsterung

SHA384withRSA

RSA-Signaturen mit dem SHA-384-Hash-Algorithmus

- PKCS #1 v1.5-Polsterung
- PSS-Polsterung

SHA512withRSA

RSA-Signaturen mit dem SHA-512-Hash-Algorithmus

- PKCS #1 v1.5-Polsterung
- PSS-Polsterung

ECDSA-Signaturtypen

Der OpenSSL Provider unterstützt digitale ECDSA-Signaturen mit mehreren Hash-Algorithmen:

SHA1withECDSA

ECDSA-Signaturen mit dem SHA-1-Hash-Algorithmus

SHA224withECDSA

ECDSA-Signaturen mit dem SHA-224-Hash-Algorithmus

SHA256withECDSA

ECDSA-Signaturen mit dem SHA-256-Hash-Algorithmus

SHA384withECDSA

ECDSA-Signaturen mit dem SHA-384-Hash-Algorithmus

SHA512withECDSA

ECDSA-Signaturen mit dem SHA-512-Hash-Algorithmus

Erweiterte Konfiguration des OpenSSL-Anbieters

Das AWS CloudHSM OpenSSL Provider SDK bietet erweiterte Konfigurationsoptionen, um sein Verhalten an verschiedene Anwendungsfälle und Umgebungen anzupassen. Verwenden Sie das Konfigurationstool, um diese Optionen festzulegen.

- [Konfiguration für OpenSSL Provider erneut versuchen](#)

Wiederholen Sie die Befehle für OpenSSL Provider für AWS CloudHSM

AWS CloudHSM Das Client-SDK 5.8.0 und höher verfügen über eine integrierte automatische Wiederholungsstrategie, mit der HSM-gedrosselte Operationen von der Clientseite aus wiederholt werden. Wenn ein HSM Operationen drosselt, weil es mit der Ausführung früherer Operationen zu beschäftigt ist und keine weiteren Anfragen annehmen kann, versucht der Client, gedrosselte Operationen bis zu dreimal zu wiederholen, während er SDKs sich exponentiell zurückzieht. Diese automatische Wiederholungsstrategie kann auf einen von zwei Modi eingestellt werden: aus und Standard.

- aus: Das Client-SDK führt bei gedrosselten Vorgängen durch das HSM keine Wiederholungsstrategie durch.
- Standard: Dies ist der Standardmodus für das Client-SDK 5.8.0 und höher. In diesem Modus wiederholt der Client automatisch gedrosselte Operationen, indem er SDKs sich exponentiell zurückzieht.

Weitere Informationen finden Sie unter [HSM-Drosselung](#).

Stellen Sie den Modus für Wiederholungsbefehle auf Aus

Linux

So setzen Sie Wiederholungsbefehle auf off für Client-SDK 5 unter Linux

- Sie können den folgenden Befehl verwenden, um die Konfiguration der Wiederholungen auf den off-Modus zu setzen:

```
$ sudo /opt/cloudhsm/bin/configure-openssl-provider --default-retry-mode off
```

Schlüsselspeicheranbieter (KSP) für AWS CloudHSM Client SDK 5

Key Storage Provider (KSP) ist eine kryptografische API, die für das Microsoft Windows-Betriebssystem spezifisch ist. Mit Key Storage Provider (KSP) können Entwickler kryptografische Techniken verwenden, um Windows-basierte Anwendungen zu sichern.

Informationen zu Bootstrapping finden Sie unter [Verbinden mit dem Cluster](#).

Informationen zur Verwendung von Client-SDK 3 finden Sie unter [Verwenden der vorherigen SDK-Version zum Arbeiten mit AWS CloudHSM](#).

Topics

- [Installieren Sie den Key Storage Provider \(KSP\) für AWS CloudHSM Client SDK 5](#)
- [Authentifizieren Sie sich beim Key Storage Provider \(KSP\) für AWS CloudHSM Client SDK 5](#)
- [Unterstützte Schlüsseltypen für Key Storage Provider \(KSP\) für AWS CloudHSM Client SDK 5](#)
- [Unterstützte API-Operationen Key Storage Provider \(KSP\) für AWS CloudHSM Client SDK 5](#)
- [Erweiterte Konfigurationen für KSP für AWS CloudHSM](#)

Installieren Sie den Key Storage Provider (KSP) für AWS CloudHSM Client SDK 5

Verwenden Sie die folgenden Abschnitte, um den Key Storage Provider (KSP) für das AWS CloudHSM Client SDK 5 zu installieren.

Note

Um einen einzelnen HSM-Cluster mit Client-SDK 5 auszuführen, müssen Sie zunächst die Einstellungen für die Haltbarkeit von Client-Schlüsseln verwalten, indem Sie die Einstellung `disable_key_availability_check` auf `True` festlegen. Weitere Informationen finden Sie unter [Schlüsselsynchronisierung](#) und [Client-SDK-5-Configure-Tool](#).

So installieren und konfigurieren Sie den Key Storage Provider (KSP)

1. Installieren Sie den Key Storage Provider (KSP) für Windows Server auf einer x86_64-Architektur, öffnen Sie ihn PowerShell als Administrator und führen Sie den folgenden Befehl aus:

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

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

2. Verwenden Sie das Configure-Tool, um den Speicherort des ausstellenden Zertifikats anzugeben. Detaillierte Anweisungen finden Sie unter [Geben Sie den Speicherort des ausstellenden Zertifikats an](#).
3. Um eine Verbindung zu Ihrem Cluster herzustellen, siehe [Bootstrap für das Client-SDK](#).
4. Sie finden die KSP-Dateien (Key Storage Provider) an den folgenden Speicherorten:
 - Windows-Binärdateien:

```
C:\Program Files\Amazon\CloudHSM
```

Windows-Konfigurationsskripten und Protokolldateien:

```
C:\ProgramData\Amazon\CloudHSM
```

Authentifizieren Sie sich beim Key Storage Provider (KSP) für AWS CloudHSM Client SDK 5

Bevor Sie den Key Storage Provider (KSP) für AWS CloudHSM Client SDK 5 verwenden, müssen Sie die Anmeldeinformationen für das HSM auf Ihrem System einrichten. Sie haben zwei Optionen:

- Windows Credentials Manager (für mehr Sicherheit empfohlen)
- Systemumgebungsvariablen (einfachere Einrichtung)

Windows-Anmeldeinformationsverwaltung

Sie können Anmeldeinformationen entweder mit dem `set_cloudhsm_credentials` Hilfsprogramm oder mit der Windows Credentials Manager-Oberfläche einrichten.

- Verwenden des Dienstprogramms **set_cloudhsm_credentials**:

Das Windows-Installationsprogramm enthält das `set_cloudhsm_credentials` Hilfsprogramm. Sie können dieses Dienstprogramm verwenden, um HSM-Anmeldeinformationen bequem an die Windows-Anmeldeinformationsverwaltung zu übergeben. Wenn Sie dieses Hilfsprogramm aus dem Quellcode kompilieren möchten, können Sie den im Installer enthaltenen Python-Code verwenden.

1. Navigieren Sie zu `C:\Program Files\Amazon\CloudHSM\tools\`.
2. Führen Sie den folgenden Befehl aus:

```
set_cloudhsm_credentials.exe --username <CU USER> --password <CU PASSWORD>
```

- Verwenden der Schnittstelle der Anmeldeinformationsverwaltung:

1. Öffnen Sie den Credential Manager:
 - Geben Sie `credential manager` in das Suchfeld der Taskleiste Folgendes ein
 - Wählen Sie Credential Manager
2. Wählen Sie Windows-Anmeldeinformationen aus, um Windows-Anmeldeinformationen zu verwalten.
3. Wählen Sie Generische Anmeldeinformationen hinzufügen
4. Geben Sie folgende Details ein:
 - Internet- oder Netzwerkadresse: `CLOUDHSM_PIN`.
 - Nutzernamen: `<CU USER>`.
 - Passwort: `<CU PASSWORD>`.
5. Klicken Sie auf OK

Systemumgebungsvariablen

Sie können Systemumgebungsvariablen festlegen, um Ihr HSM und Ihren [Crypto-Benutzer](#) (CU) zu identifizieren.

 Warning

Wenn Sie Anmeldeinformationen über Systemumgebungsvariablen festlegen, wird Ihr Passwort im Klartext auf Ihrem System gespeichert. Verwenden Sie aus Sicherheitsgründen stattdessen den Windows Credential Manager.

Sie können Umgebungsvariablen wie folgt festlegen:

- Das Tool [setx](#).
- Die Windows-Systemeigenschaften-Systemsteuerung (Registerkarte „Erweitert“).
- legt permanente Systemumgebungsvariablen fest. [Programmatische](#) Methoden.

So legen Sie die Systemumgebungsvariable fest:

CLOUDHSM_PIN=<CU USERNAME>:<CU PASSWORD>

Identifiziert einen [Crypto-Benutzer](#) (CU, Crypto User) im HSM und stellt alle erforderlichen Anmeldeinformationen bereit. Ihre Anwendung führt die Authentifizierung durch und wird als dieser CU ausgeführt. Die Anwendung verfügt über die Berechtigungen dieses CU und kann nur die Schlüssel anzeigen und verwalten, die sich im Besitz dieses CU befinden und von ihm freigegeben werden. Verwenden Sie den Befehl [user create in der CloudHSM-CLI, um eine neue CU zu erstellen](#). Verwenden Sie den Befehl [user list](#) in der CloudHSM-CLI CUs, um nach vorhandenen zu suchen.

Beispiel:

```
setx /m CLOUDHSM_PIN test_user:password123
```

 Note

Beim Setzen von CLOUDHSM_PIN-Umgebungsvariablen müssen Sie alle Sonderzeichen maskieren, die von Ihrer Shell interpretiert werden könnten.

Unterstützte Schlüsseltypen für Key Storage Provider (KSP) für AWS CloudHSM Client SDK 5

Der AWS CloudHSM Key Storage Provider (KSP) unterstützt die folgenden Schlüsseltypen mit dem Client SDK 5.

Schlüsseltyp	Description
EC	Generieren Sie Schlüssel mit den Kurven secp256r1 (P-256), secp384r1 (P-384) und secp521r1 (P-521).
RSA	Generieren Sie 2048-, 3072- und 4096-Bit-RSA-Schlüssel.

Unterstützte API-Operationen Key Storage Provider (KSP) für AWS CloudHSM Client SDK 5

Die Parameter im KSP werden von Microsoft KSP definiert. Weitere Informationen finden Sie in der [Microsoft-Dokumentation](#).

Der Key Storage Provider (KSP) unterstützt die folgenden KSP-API-Operationen für das AWS CloudHSM Client-SDK 5.

- [NCryptOpenStorageProvider](#)
- [NCryptOpenKey](#)
- [NCryptCreatePersistedKey](#)
- [NCryptGetProperty](#)
- [NCryptSetProperty](#)
- [NCryptFinalizeKey](#)
- [NCryptDeleteKey](#)
- [NCryptFreeObject](#)
- [NCryptFreeBuffer](#)
- [NCryptIsAlgSupported](#)
- [NCryptEnumAlgorithms](#)

- [NCryptEnumKeys](#)
- [NCryptExportKey](#)
- [NCryptSignHash](#)
- [NCryptVerifySignature](#)

NCryptOpenStorageProvider Funktion mit Key Storage Provider (KSP)

Die NCryptOpenStorageProvider Funktion lädt und initialisiert den Key Storage Provider (KSP).

Parameters

phProvider[aus]

Ein Zeiger auf eine NCRYPT_PROV_HANDLE Variable, die das Anbieter-Handle speichert.

pszProviderName[in]

Ein Zeiger auf eine mit Null terminierte Unicode-Zeichenfolge, die den Schlüsselspeicheranbieter identifiziert. AWS CloudHSM Key Storage Provider (KSP) unterstützt die folgenden Werte:

Wert	Bedeutung
ICloudHSM Key Storage Provider	Identifiziert den Namen des Client SDK 5-Anbieters. Wir empfehlen, diesen Namen standardmäßig zu verwenden.
L „Cavium Key Storage Provider“	Identifiziert den Namen des Client SDK 3-Anbieters. Unterstützt aus Gründen der Abwärtskompatibilität.

Note

Bei den Werten handelt es sich um Zeichenkettenlitterale mit breiten Zeichen, wie durch L vor dem Literal angegeben.

dwFlags[in]

Flags, die das Verhalten der Funktion ändern. Für diese Funktion sind keine Flags definiert.

Rückgabewert

Die Funktion gibt einen Statuscode zurück, der auf Erfolg oder Misserfolg hinweist.

Zu den gängigen Rückgabecodes gehören:

Rückgabecode	Description
ERROR_SUCCESS	Der Vorgang wurde erfolgreich abgeschlossen.
NTE_INVALID_PARAMETER	Ein oder mehrere Parameter sind nicht gültig.
NET_FAIL	Der Vorgang konnte nicht abgeschlossen werden.

NCryptOpenKey mit Key Storage Provider (KSP)

Die NCryptOpenKey Funktion öffnet einen Schlüssel, der im Key Storage Provider (KSP) vorhanden ist.

Parameters

hProvider[in]

Das KSP-Handle, das den Schlüssel enthält. Verwenden Sie [NCryptOpenStorageProvider](#), um das Handle zu erhalten.

phKey[raus]

Ein Zeiger auf eine NCrypt_KEY_HANDLE Variable, die das Tastenkürzel speichert.

pszKeyName[in]

Ein Zeiger auf eine mit Null terminierte Unicode-Zeichenfolge, die den Schlüsselnamen enthält.

dwLegacyKeySpec[in, unbenutzt]

AWS CloudHSM Key Storage Provider (KSP) verwendet diesen Parameter nicht.

dwFlags[in]

Flags, die das Verhalten der Funktion ändern. Für diese Funktion sind keine Flags definiert.

Rückgabewert

Die Funktion gibt einen Statuscode zurück, der auf Erfolg oder Misserfolg hinweist.

Zu den gängigen Rückgabecodes gehören:

Rückgabecode	Description
ERROR_SUCCESS	Der Vorgang wurde erfolgreich abgeschlossen.
NTE_INVALID_PARAMETER	Ein oder mehrere Parameter sind nicht gültig.
NET_FAIL	Der Vorgang konnte nicht abgeschlossen werden.
NTE_INVALID_HANDLE	Das Handle in ist nicht gültig. hProvider
NET_BAD_KEYSET	Der angegebene Schlüsselname lieferte kein eindeutiges Ergebnis.

NCryptCreatePersistedKey mit Key Storage Provider (KSP)

Die NCryptCreatePersistedKey Funktion erstellt einen neuen Schlüssel und speichert ihn im Key Storage Provider (KSP). Sie können die [NCryptSetProperty](#) Funktion verwenden, um ihre Eigenschaften nach der Erstellung festzulegen. Sie müssen anrufen, [NCryptFinalizeKey](#) bevor Sie den Schlüssel verwenden können.

Parameters

hProvider[rein]

Das Handle des Schlüsselspeicheranbieters, unter dem Sie den Schlüssel erstellen werden. Verwenden Sie [NCryptOpenStorageProvider](#), um dieses Handle zu erhalten.

phKey[raus]

Die Adresse einer NCRYPT_KEY_HANDLE Variablen, die das Schlüssel-Handle speichert.

pszAlgId[in]

Ein Zeiger auf eine mit Null terminierte Unicode-Zeichenfolge, die den Bezeichner des kryptografischen Algorithmus für die Erstellung des Schlüssels angibt.

AWS CloudHSM Der Key Storage Provider (KSP) unterstützt die folgenden Algorithmen:

Konstante/Wert	Description
BCRYPT_RSA_ALGORITHM „RSA“	Der RSA-Algorithmus für öffentliche Schlüssel.
BCRYPT_ECDSA_P256_ALGORITHM „ECDSA_P256“	Der 256-Bit-Algorithmus für digitale Signaturen mit elliptischen Primzahlkurven (FIPS 186-2).
BCRYPT_ECDSA_P384_ALGORITHM „ECDSA_P384“	Der 384-Bit-Algorithmus für digitale Signaturen mit elliptischen Primzahlkurven (FIPS 186-2).
BCRYPT_ECDSA_P521_ALGORITHM „ECDSA_P521“	Der 521-Bit-Algorithmus für digitale Signaturen mit elliptischer Primzahlkurve (FIPS 186-2).

pszKeyName[rein, optional]

Ein Zeiger auf eine Unicode-Zeichenfolge mit Nullterminierung, die den Namen des Schlüssels enthält. Wenn dieser Parameter NULL ist, erstellt diese Funktion einen kurzlebigen Schlüssel, der nicht persistent ist.

dwLegacyKeySpec[in, unbenutzt]

AWS CloudHSM Key Storage Provider (KSP) verwendet diesen Parameter nicht.

dwFlags[in]

Flags, um das Verhalten der Funktion zu ändern. Verwenden Sie null oder mehr der folgenden Werte:

Wert	Bedeutung
NCRYPT_MACHINE_KEY_FLAG	Dieses Flag hat keine Wirkung.
NCRYPT_SILENT_FLAG	Dieses Flag hat keine Wirkung.

Wert	Bedeutung
NCRYPT_OVERWRITE_KEY_FLAG	Wenn Sie dieses Flag angeben, werden alle vorhandenen Schlüssel mit demselben Namen im HSM überschrieben. Ohne dieses Flag kehrt die Funktion zurück. NTE_EXISTS

Rückgabewert

Die Funktion gibt einen Statuscode zurück, der auf Erfolg oder Misserfolg hinweist.

Zu den gängigen Rückgabecodes gehören:

Rückgabecode	Description
ERROR_SUCCESS	Die Funktion wurde erfolgreich abgeschlossen.
NTE_INVALID_PARAMETER	Ein oder mehrere Parameter sind nicht gültig.
NET_FAIL	Der Vorgang konnte nicht abgeschlossen werden.
NET_BAD_FLAGS	Der dwFlags Parameter enthält einen ungültigen Wert.
NTE_NOT_SUPPORTED	Der pszAlgId Parameter enthält einen Wert, der nicht unterstützt wird.
NTE_EXISTS	Ein Schlüssel mit dem angegebenen Namen ist bereits vorhanden und der Vorgang wurde nicht verwendet. NCRYPT_OVERWRITE_KEY_FLAG

NCryptGetProperty mit Key Storage Provider (KSP)

Die NCryptGetProperty Funktion ruft Eigenschaftswerte für ein Schlüsselspeicherobjekt ab.

Parameters

hObject[in]

Das Handle des Objekts, dessen Eigenschaft Sie abrufen möchten. Sie können Folgendes verwenden:

- Ein Anbieter-Handle (NCRYPT_PROV_HANDLE)
- Ein Schlüsselgriff (NCRYPT_KEY_HANDLE)

pszProperty [in]

Ein Zeiger auf eine Unicode-Zeichenfolge mit Nullterminierung, die den abzurufenden Eigenschaftsnamen enthält.

Bei der Verwendung NCRYPT_PROV_HANDLE unterstützt AWS CloudHSM Key Storage Provider (KSP) die folgenden KSP-Bezeichner:

Bezeichner/Wert	Description
NCRYPT_IMPL_TYPE_PROPERTY L „Impl Typ“	Ein DWORD, das Flags enthält, die Details der Provider-Implementierung definieren
NCRYPT_MAX_NAME_LENGTH_PROPERTY L „Max. Länge des Namens“	Ein DWORD, das die maximale Länge (in Zeichen) für einen persistenten Schlüsselnamen enthält.
NCRYPT_NAME_PROPERTY L „Name“	Ein Zeiger auf eine mit Null terminierte Unicode-Zeichenfolge, die den KSP-Namen enthält.
NCRYPT_VERSION_PROPERTY L „Version“	Ein DWORD, das die Provider-Version enthält (hohes Wort: Hauptversion, niedriges Wort: Nebenversion).
NCRYPT_USE_CONTEXT_PROPERTY Ich verwende Kontext	Ein Zeiger auf eine mit Null terminierte Unicode-Zeichenfolge, die den Operationsskontext beschreibt.

Bezeichner/Wert	Description
NCRYPT_SECURITY_DESCR_SUPPORT_PROPERTY L „Security Descr Support“	Gibt an, ob der Anbieter Sicherheitsbeschreibungen für Schlüssel unterstützt.

Bei der Verwendung NCRYPT_KEY_HANDLE unterstützt der AWS CloudHSM Key Storage Provider (KSP) die folgenden KSP-Identifikatoren:

Bezeichner/Wert	Description
NCRYPT_ALGORITHM_PROPERTY L „Name des Algorithmus“	Eine mit Null terminierte Unicode-Zeichenfolge, die den Algorithmusnamen des Schlüssels enthält.
NCRYPT_BLOCK_LENGTH_PROPERTY L „Blocklänge“	Ein DWORD, das die Länge des Verschlüsselungsblocks in Byte enthält.
NCRYPT_EXPORT_POLICY_PROPERTY L „Richtlinie exportieren“	Ein DWORD mit Flags, die die Exportrichtlinie des persistenten Schlüssels angeben.
NCRYPT_KEY_USAGE_PROPERTY L „Verwendung des Schlüssels“	Ein DWORD, das Flags enthält, die Details zur Schlüsselverwendung definieren.
NCRYPT_KEY_TYPE_PROPERTY L „Schlüsseltyp“	Ein DWORD, das Flags enthält, die den Schlüsseltyp definieren.
NCRYPT_LENGTH_PROPERTY L „Länge“	Ein DWORD, das die Schlüssellänge in Bits enthält.
NCRYPT_LENGTHS_PROPERTY L „Längen“	Ein Zeiger auf eine NCRYPT_SUPPORTED_LENGTHS-Struktur, die unterstützte Schlüsselgrößen enthält.

Bezeichner/Wert	Description
NCRYPT_NAME_PROPERTY L „Name“	Ein Zeiger auf eine mit Null terminierte Unicode-Zeichenfolge, die den Schlüsselnamen enthält.
NCRYPT_SECURITY_DESCR_PROPERTY L „Sicherheitsbescheinigung“	Ein Zeiger auf eine SECURITY_DESCRIPTOR-Struktur, die wichtige Informationen zur Zugriffskontrolle enthält.
NCRYPT_ALGORITHM_GROUP_PROPERTY L „Algorithmusgruppe“	Eine mit Null terminierte Unicode-Zeichenfolge, die den Namen der Algorithmusgruppe des Objekts enthält.
NCRYPT_UNIQUE_NAME_PROPERTY L „Eindeutiger Name“	Ein Zeiger auf eine mit Null terminierte Unicode-Zeichenfolge, die den eindeutigen Namen des Schlüssels enthält.

 Note

Bei Werten handelt es sich um Zeichenkettenliteral mit breiten Zeichen, wie durch L vor dem Literal angegeben.

pbOutput[aus]

Die Adresse eines Puffers zum Speichern des Eigenschaftswerts. Geben Sie die Puffergröße mit `cbOutput`.

Um die erforderliche Puffergröße zu ermitteln, setzen Sie diesen Parameter auf `NULL`. Die Funktion speichert die erforderliche Größe (in Byte) an dem Ort, auf den von `zeigtpcbResult`.

cbOutput[in]

Die Größe des `pbOutput` Puffers in Byte.

pcbResult[aus]

Ein Zeiger auf eine `DWORD`-Variable, die die Anzahl der in den `pbOutput` Puffer kopierten Byte speichert.

Wenn der Wert NULL `pbOutput` ist, wird die erforderliche Größe (in Byte) gespeichert.
`dwFlags[in]`

Flags, um das Verhalten der Funktion zu ändern. Sie können Null verwenden oder:

Wert	Bedeutung
<code>NCRYPT_SILENT_FLAG</code>	Diese Flagge hat keine Wirkung.

Wenn `pszProperty` den Wert `NCRYPT_SECURITY_DESCR_PROPERTY` hat, verwenden Sie eine oder eine Kombination aus:

Wert	Bedeutung
<code>OWNER_SECURITY_INFORMATION</code>	Diese Flagge hat keine Wirkung.
<code>GROUP_SECURITY_INFORMATION</code>	Dieses Flag hat keine Wirkung.
<code>DACL_SECURITY_INFORMATION</code>	Diese Flagge hat keine Wirkung.
<code>LABEL_SECURITY_INFORMATION</code>	Dieses Flag hat keine Wirkung.
<code>SACL_SECURITY_INFORMATION</code>	Dieses Flag hat keine Wirkung.

Rückgabewert

Die Funktion gibt einen Statuscode zurück, der auf Erfolg oder Misserfolg hinweist.

Zu den gängigen Rückgabecodes gehören:

Rückgabecode	Description
<code>ERROR_SUCCESS</code>	Der Vorgang wurde erfolgreich abgeschlossen.
<code>NTE_INVALID_PARAMETER</code>	Ein oder mehrere Parameter sind nicht gültig.
<code>NET_FAIL</code>	Der Vorgang konnte nicht abgeschlossen werden.

Rückgabecode	Description
NET_BAD_FLAGS	Der dwFlags Parameter enthält einen ungültigen Wert.
NTE_NOT_SUPPORTED	Der pszAlgId Parameter enthält einen Wert, der nicht unterstützt wird.
NTE_INVALID_HANDLE	Das Handle in ist nicht gültig. hObject
NET_BUFFER_TOO_SMALL	Der cbOutput Parameter ist zu klein für Rückgabewerte.

NCryptSetProperty mit Key Storage Provider (KSP)

Die NCryptSetProperty Funktion legt Eigenschaftswerte für ein Schlüsselspeicherobjekt fest.

Parameters

hObject[in]

Das Handle des Objekts, dessen Eigenschaft Sie festlegen möchten. Sie können Folgendes verwenden:

- Ein Anbieter-Handle (NCRYPT_PROV_HANDLE)
- Ein Schlüsselgriff (NCRYPT_KEY_HANDLE)

pszProperty [in]

Ein Zeiger auf eine Unicode-Zeichenfolge mit Nullterminierung, die den abzurufenden Eigenschaftsnamen enthält.

Bei der Verwendung NCRYPT_PROV_HANDLE unterstützt AWS CloudHSM Key Storage Provider (KSP) die folgenden KSP-Bezeichner:

Bezeichner/Wert	Description
NCRYPT_USE_CONTEXT_PROPERTY	Ein Zeiger auf eine mit Null terminierte Unicode-Zeichenfolge, die den Operation
Ich verwende Kontext	skontext beschreibt.

Bei der Verwendung `NCRYPT_KEY_HANDLE` unterstützt AWS CloudHSM Key Storage Provider (KSP) die folgenden KSP-Bezeichner:

Bezeichner/Wert	Description
<code>NCRYPT_KEY_USAGE_PROPERTY</code> L „Verwendung des Schlüssels“	Ein <code>DWORD</code>, das eine Reihe von Flags enthält, die Details zur Schlüsselnutzung definieren. Diese Eigenschaft gilt nur für Schlüssel. Dies kann Null oder eine Kombination aus einem oder mehreren der folgenden Werte enthalten. <code>NCRYPT_ALLOW_DECRYPT_FLAG</code> (0x00000001) <code>NCRYPT_ALLOW_SIGNING_FLAG</code> (0x00000002)
<code>NCRYPT_LENGTH_PROPERTY</code> L „Länge“	Ein <code>DWORD</code>, das die Schlüssellänge in Bits enthält.
<code>NCRYPT_EXPORT_POLICY_PROPERTY</code> L „Richtlinie exportieren“	Ein <code>DWORD</code> mit Flags, die die Exportrichtlinie des persistenten Schlüssels angeben. Dies kann Null oder eine Kombination aus einem oder mehreren der folgenden Werte enthalten. <code>NCRYPT_ALLOW_EXPORT_FLAG</code> (0x00000001)

 Note

Bei den Werten handelt es sich um Zeichenkettenliteral mit breiten Zeichen, wie durch L vor dem Literal angegeben.

pbInput[in]

Die Adresse eines Puffers, der den neuen Eigenschaftswert enthält. cbInput enthält die Größe des Puffers.

cbInput[in]

Die Größe des pbInput Puffers in Byte.

dwFlags[in]

Flags, die das Verhalten der Funktion ändern. Für diese Funktion sind keine Flags definiert.

Rückgabewert

Die Funktion gibt einen Statuscode zurück, der auf Erfolg oder Misserfolg hinweist.

Zu den gängigen Rückgabecodes gehören:

Rückgabecode	Description
ERROR_SUCCESS	Der Vorgang wurde erfolgreich abgeschlossen.
NTE_INVALID_PARAMETER	Ein oder mehrere Parameter sind nicht gültig.
NET_FAIL	Der Vorgang konnte nicht abgeschlossen werden.
NET_BAD_FLAGS	Der dwFlags Parameter enthält einen ungültigen Wert.
NTE_NOT_SUPPORTED	Der pszProperty Parameter enthält einen Wert, der nicht unterstützt wird.
NTE_INVALID_HANDLE	Das Handle in ist nicht gültig. hObject
NET_BAD_DATA	Die Daten, auf die verwiesen wird pbInput und die nicht gültig sind. cbInput

NCryptFinalizeKey mit Key Storage Provider (KSP)

Die NCryptFinalizeKey Funktion vervollständigt einen KSP-Schlüssel. Sie müssen diese Funktion aufrufen, bevor Sie die Taste verwenden können.

Parameters

hKey[in]

Der Griff des Schlüssels, der abgeschlossen werden soll. Rufen Sie dieses Handle ab, indem Sie die [NCryptCreatePersistedKey](#) Funktion aufrufen.

dwFlags[rein]

Flags, um das Verhalten der Funktion zu ändern. Sie können Null oder diese Werte verwenden:

Wert	Bedeutung
NCRYPT_SILENT_FLAG	Diese Flagge hat keine Wirkung.
NCRYPT_NO_KEY_VALIDATION	Dieses Flag hat keine Wirkung.

Rückgabewert

Die Funktion gibt einen Statuscode zurück, der auf Erfolg oder Misserfolg hinweist.

Zu den gängigen Rückgabecodes gehören:

Rückgabecode	Description
ERROR_SUCCESS	Der Vorgang wurde erfolgreich abgeschlossen.
NET_FAIL	Der Vorgang konnte nicht abgeschlossen werden.
NTE_INVALID_HANDLE	Das Handle in ist nicht gültig. hKey
NET_NOT_SUPPORTED	Der dwFlags Parameter enthält einen Wert, der nicht unterstützt wird.

Rückgabecode	Description
NET_BAD_FLAGS	Der dwFlags Parameter enthält einen ungültigen Wert.

NCryptDeleteKey mit Key Storage Provider (KSP)

Die NCryptDeleteKey Funktion löscht einen KSP-Schlüssel vom Key Storage Provider (KSP).

Parameters

hKey[in]

Der Griff des Schlüssels, der gelöscht werden soll.

dwFlags[in]

Flags, um das Verhalten der Funktion zu ändern. Sie können null oder mehr der folgenden Werte verwenden:

Wert	Bedeutung
NCRYPT_SILENT_FLAG	Dieses Flag hat keine Wirkung.

Rückgabewert

Die Funktion gibt einen Statuscode zurück, der auf Erfolg oder Misserfolg hinweist.

Zu den gängigen Rückgabecodes gehören:

Rückgabecode	Description
ERROR_SUCCESS	Die Funktion war erfolgreich.
NTE_INVALID_PARAMETER	Ein oder mehrere Parameter sind nicht gültig.
NET_BAD_FLAGS	Der dwFlags Parameter enthält einen ungültigen Wert.

Rückgabecode	Description
NET_FAIL	Der Vorgang konnte nicht abgeschlossen werden.
NTE_INVALID_HANDLE	Das Handle in ist nicht gültig. hKey
NET_INTERNAL_ERROR	Beim Löschen des Schlüssels ist ein interner Fehler aufgetreten.

NCryptFreeObject mit Key Storage Provider (KSP)

Die NCryptFreeObject Funktion gibt den Provider oder das Key-Handle vom Key Storage Provider (KSP) frei.

Parameters

hObject[in]

Der Handle des Objekts, das freigegeben werden soll. Sie können Folgendes verwenden:

- Ein Provider-Handle (NCRYPT_PROV_HANDLE)
- Ein Schlüsselgriff (NCRYPT_KEY_HANDLE)

Rückgabewert

Die Funktion gibt einen Statuscode zurück, der auf Erfolg oder Misserfolg hinweist.

Zu den gängigen Rückgabecodes gehören:

Rückgabecode	Description
ERROR_SUCCESS	Der Vorgang wurde erfolgreich abgeschlossen.
NTE_INVALID_HANDLE	Das Handle in ist nicht gültig. hObject

NCryptFreeBuffer mit Key Storage Provider (KSP)

Die NCryptFreeBuffer Funktion gibt einen Speicherblock frei, der vom Key Storage Provider (KSP) zugewiesen wurde.

Parameters

pvInput[in]

Die Adresse des Speichers, der freigegeben werden soll.

Rückgabewert

Die Funktion gibt einen Statuscode zurück, der auf Erfolg oder Misserfolg hinweist.

Zu den gängigen Rückgabecodes gehören:

Rückgabecode	Description
ERROR_SUCCESS	Der Vorgang wurde erfolgreich abgeschlossen.
NET_FAIL	Der Vorgang konnte nicht abgeschlossen werden.

NCryptIsAlgSupported mit Key Storage Provider (KSP)

NCryptIsAlgSupported Die Funktion bestimmt, ob der Key Storage Provider (KSP) einen bestimmten kryptografischen Algorithmus unterstützt.

Parameters

hProvider[in]

Das Handle des Schlüsselspeicheranbieters. Verwenden Sie [NCryptOpenStorageProvider](#), um das Handle zu bekommen.

pszAlgId[in]

Ein Zeiger auf eine mit Null terminierte Unicode-Zeichenfolge, die den Bezeichner des kryptografischen Algorithmus zur Erstellung des Schlüssels enthält. AWS CloudHSM Key Storage Provider (KSP) unterstützt die folgenden Algorithmen:

Konstante/Wert	Description
BCRYPT_RSA_ALGORITHM „RSA“	Der RSA-Algorithmus für öffentliche Schlüssel .
BCRYPT_ECDSA_P256_ALGORITHM „ECDSA_P256“	Der 256-Bit-Algorithmus für digitale Signaturen mit elliptischer Primzahlkurve (FIPS 186-2).
BCRYPT_ECDSA_P384_ALGORITHM „ECDSA_P384“	Der 384-Bit-Algorithmus für digitale Signaturen mit elliptischer Primzahlkurve (FIPS 186-2).
BCRYPT_ECDSA_P521_ALGORITHM „ECDSA_P521“	Der 521-Bit-Algorithmus für digitale Signaturen mit elliptischer Primzahlkurve (FIPS 186-2).

dwFlags[in]

Flags, die das Verhalten von Funktionen ändern. Dies kann Null oder der folgende Wert sein:

Wert	Bedeutung
NCRYPT_SILENT_FLAG	Diese Flagge hat keine Wirkung.

Rückgabewert

Die Funktion gibt einen Statuscode zurück, der auf Erfolg oder Misserfolg hinweist.

Zu den gängigen Rückgabecodes gehören:

Rückgabecode	Description
ERROR_SUCCESS	Der Vorgang wurde erfolgreich abgeschlossen.
NTE_INVALID_PARAMETER	Ein oder mehrere Parameter sind nicht gültig.

Rückgabecode	Description
NET_BAD_FLAGS	Der dwFlags Parameter enthält einen ungültigen Wert.
NTE_NOT_SUPPORTED	Der pszAlgId Parameter enthält einen Wert, der nicht unterstützt wird.
NTE_INVALID_HANDLE	Das Handle in ist nicht gültig. hProvider

NCryptEnumAlgorithms mit Key Storage Provider (KSP)

Die NCryptEnumAlgorithms Funktion ruft die Namen der Algorithmen ab, die der Key Storage Provider (KSP) unterstützt.

Parameters

hProvider[in]

Das Handle des Schlüsselspeicheranbieters, für den die Algorithmen aufgezählt werden sollen. Verwenden Sie die [NCryptOpenStorageProvider](#) Funktion, um dieses Handle abzurufen.

dwAlgOperations[in]

Eine Reihe von Werten, die angeben, welche Algorithmusklassen aufgezählt werden sollen. Sie können Null verwenden, um alle Algorithmen aufzuzählen, oder Sie können einen oder mehrere dieser Werte kombinieren:

Wert	Bedeutung
NCRYPT_ASYMMETRIC_ENCRYPTION_OPERATION 0x00000004	Listet die asymmetrischen Verschlüsselungsalgorithmen auf.
NCRYPT_SIGNATURE_OPERATION 0x00000010	Listet die Algorithmen für digitale Signaturen auf.

pdwAlgCount[aus]

Die Adresse eines DWORD-Objekts, das die Anzahl der Elemente im ppAlgList Array speichert.

ppAlgList[aus]

Die Adresse eines NCryptAlgorithmName Strukturzeigers, der eine Reihe registrierter Algorithmusnamen speichert. Der pdwAlgCount Parameter gibt die Anzahl der Elemente in diesem Array an.

dwFlags[in]

Flags, um das Verhalten der Funktion zu ändern. Verwenden Sie Null oder den folgenden Wert:

Wert	Bedeutung
NCRYPT_SILENT_FLAG	Dieses Flag hat keine Wirkung.

Rückgabewert

Die Funktion gibt einen Statuscode zurück, der auf Erfolg oder Misserfolg hinweist.

Zu den gängigen Rückgabecodes gehören:

Rückgabecode	Description
ERROR_SUCCESS	Der Vorgang wurde erfolgreich abgeschlossen.
NTE_INVALID_PARAMETER	Ein oder mehrere Parameter sind nicht gültig.
NET_FAIL	Der Vorgang konnte nicht abgeschlossen werden.
NET_BAD_FLAGS	Der dwFlags Parameter enthält einen ungültigen Wert.
NTE_NOT_SUPPORTED	Der dwAlgOperations Parameter enthält einen Wert, der nicht unterstützt wird.

NCryptEnumKeys mit Key Storage Provider (KSP)

NCryptEnumKeys Die Funktion listet die im Key Storage Provider (KSP) gespeicherten Schlüssel auf.

Parameters

hProvider[in]

Das Handle des wichtigsten Speicheranbieters. Verwenden Sie [NCryptOpenStorageProvider](#), um dieses Handle zu erhalten.

pszScope[rein, unbenutzt]

Setzen Sie diesen Parameter auf NULL.

ppKeyName[aus]

Eine Zeigeradresse auf eine NCryptKeyName Struktur, die den Schlüsselnamen speichert. Rufen Sie auf, um diesen Speicher nach dem Gebrauch wieder freizugeben [NCryptFreeBuffer](#).

ppEnumState[rein, raus]

Eine VOID-Zeigeradresse, die den Fortschritt der Aufzählung verfolgt. Der Schlüsselspeicheranbieter verwendet diese Informationen intern, um die Aufzählungssequenz zu verwalten. Um eine neue Aufzählung von Anfang an zu beginnen, setzen Sie diesen Zeiger auf NULL.

Um diesen Speicher nach Abschluss der Aufzählung freizugeben, übergeben Sie diesen Zeiger an den. [NCryptFreeBuffer](#)

dwFlags[in]

Flags, um das Verhalten der Funktion zu ändern. Diese Funktion hat keine Flags.

Rückgabewert

Die Funktion gibt einen Statuscode zurück, der auf Erfolg oder Misserfolg hinweist.

Zu den gängigen Rückgabecodes gehören:

Rückgabecode	Description
ERROR_SUCCESS	Der Vorgang wurde erfolgreich abgeschlossen.

Rückgabecode	Description
NTE_INVALID_PARAMETER	Ein oder mehrere Parameter sind nicht gültig.
NET_FAIL	Der Vorgang konnte nicht abgeschlossen werden.
NTE_INVALID_HANDLE	Das Handle in ist nicht gültig. hProvider
NET_NO_MORE_ITEMS	Die Aufzählung hat alle verfügbaren Schlüssel aufgelistet.

NCryptExportKey mit Key Storage Provider (KSP)

Die NCryptExportKey Funktion exportiert einen KSP-Schlüssel in einen Speicher. BLOB Diese Funktion unterstützt nur den Export von öffentlichen Schlüsseln.

Parameters

hKey[in]

Das Handle des zu exportierenden Schlüssels.

hExportKey[drin, unbenutzt]

AWS CloudHSM Key Storage Provider (KSP) verwendet diesen Parameter nicht.

pszBlobType[in]

Eine Unicode-Zeichenfolge mit Nullabschluss, die den zu exportierenden BLOB Typ angibt. AWS CloudHSM Key Storage Provider (KSP) unterstützt die folgenden Werte:

Wert	Bedeutung
BCRYPT_RSAPUBLIC_BLOB	Exportiert einen öffentlichen RSA-Schlüssel. Der pbOutput Puffer enthält eine BCRYPT_RSAKEY_BLOB Struktur, gefolgt von den Schlüsseldaten.
BCRYPT_ECCPUBLIC_BLOB	Exportiert einen öffentlichen Schlüssel. ECC Der pbOutput Puffer enthält eine

Wert	Bedeutung
	BCRYPT_ECCKEY_BLOB Struktur, gefolgt von den Schlüsseldaten.

pParameterList[drin, unbenutzt]

AWS CloudHSM Key Storage Provider (KSP) verwendet diesen Parameter nicht.

pbOutput[aus, optional]

Eine Pufferadresse zum Speichern des Schlüssels BLOB. Geben Sie die Puffergröße mit cbOutput. Wenn der Wert auf NULL gesetzt ist, speichert die Funktion die erforderliche Größe (in Byte) in dem DWORD, auf das verwiesen wird pcbResult.

cbOutput[in]

Die Größe des pbOutput Puffers in Byte.

pcbResult[aus]

Eine DWORD-Variablenadresse, die die Anzahl der in den pbOutput Puffer kopierten Byte speichert. Wenn NULL pbOutput ist, speichert die Funktion die erforderliche Puffergröße in Byte.

dwFlags[in]

Flags, die die Funktionsweise der Funktion ändern. Sie können Null oder Folgendes verwenden:

Wert	Bedeutung
NCRYPT_SILENT_FLAG	Diese Flagge hat keine Wirkung.

Rückgabewert

Die Funktion gibt einen Statuscode zurück, der auf Erfolg oder Misserfolg hinweist.

Zu den gängigen Rückgabecodes gehören:

Rückgabecode	Description
ERROR_SUCCESS	Der Vorgang wurde erfolgreich abgeschlossen.

Rückgabecode	Description
NTE_INVALID_PARAMETER	Ein oder mehrere Parameter sind nicht gültig.
NET_FAIL	Der Vorgang konnte nicht abgeschlossen werden.
NTE_INVALID_HANDLE	Das Handle in ist nicht gültig. hProvider
NET_BAD_FLAGS	Der dwFlags Parameter enthält einen ungültigen Wert.
NTE_BAD_KEY_STATE	Der Schlüsselstatus ist nicht gültig.
NET_NOT_SUPPORTED	Der dwFlags Parameter pszBlobType oder enthält einen nicht unterstützten Wert.
STATUS_INTERNAL_ERROR	Während des Vorgangs ist ein interner Fehler aufgetreten.

NCryptSignHash mit Key Storage Provider (KSP)

Die NCryptSignHash Funktion erstellt eine Signatur eines Hashwerts.

Parameters

hKey[in]

Das Handle des Schlüssels, mit dem der Hash signiert werden soll.

pPaddingInfo[ein, optional]

Ein Zeiger auf eine Struktur, die Füllinformationen enthält. Der Strukturtyp hängt vom dwFlags Wert ab. Verwenden Sie diesen Parameter nur mit asymmetrischen Schlüsseln. Setzen Sie ihn für andere Schlüsseltypen auf NULL.

pbHashValue[in]

Ein Zeiger auf einen Puffer, der den zu signierenden Hashwert enthält. Geben Sie die Puffergröße mit cbHashValue.

cbHashValue[in]

Die Größe des zu signierenden pbHashValue Puffers in Byte.

pbSignature[aus]

Die Adresse eines Puffers zum Speichern der Signatur. Geben Sie die Puffergröße mit cbSignature.

Um die erforderliche Puffergröße zu ermitteln, setzen Sie diesen Parameter auf NULL. Die Funktion speichert die erforderliche Größe (in Byte) an dem Ort, auf den von zeigt pcbResult.

cbSignature[in]

Die Größe des pbSignature Puffers in Byte. Die Funktion ignoriert diesen Parameter, wenn er NULL pbSignature ist.

pcbResult[aus]

Ein Zeiger auf eine DWORD-Variable, die die Anzahl der in den pbSignature Puffer kopierten Byte speichert.

Wenn pbSignature der Wert NULL ist, wird die erforderliche Puffergröße in Byte gespeichert.

dwFlags[in]

Flags, um das Verhalten der Funktion zu ändern. Die erlaubten Flags hängen von Ihrem Schlüsseltyp ab. Verwenden Sie einen der folgenden Werte:

Wert	Bedeutung
BCRYPT_PAD_PKCS1	Verwendet das Padding-Schema. PKCS1 Legt fest pPaddingInfo , dass es auf eine BCRYPT_PKCS1_PADDING_INFO Struktur zeigt.
BCRYPT_PAD_PSS	Verwendet das Auffüllschema Probabilistic Signature Scheme (PSS). Setzt den pPaddingInfo Parameter so, dass er auf eine Struktur zeigt. BCRYPT_PSS_PADDING_INFO
NCRYPT_SILENT_FLAG	Diese Flagge hat keine Wirkung.

Rückgabewert

Die Funktion gibt einen Statuscode zurück, der auf Erfolg oder Misserfolg hinweist.

Zu den gängigen Rückgabecodes gehören:

Rückgabecode	Description
ERROR_SUCCESS	Der Vorgang wurde erfolgreich abgeschlossen.
NTE_INVALID_PARAMETER	Ein oder mehrere Parameter sind nicht gültig.
NET_FAIL	Der Vorgang konnte nicht abgeschlossen werden.
NTE_INVALID_HANDLE	Das Handle in ist nicht gültig. hKey
NET_BAD_FLAGS	Der dwFlags Parameter enthält einen ungültigen Wert.
NTE_BUFFER_TOO_SMALL	Der pcbOutput Parameter ist zu klein für Rückgabewerte.
NET_BAD_KEY_STATE	Der Schlüsselstatus ist nicht gültig.
NTE_INTERNAL_ERROR	Beim Signieren des Hashs ist ein interner Fehler aufgetreten.

NCryptVerifySignature mit Key Storage Provider (KSP)

Die NCryptVerifySignature Funktion bestätigt, ob eine Signatur mit einem angegebenen Hash übereinstimmt.

Parameters

hKey[in]

Das Handle des Schlüssels, der zum Entschlüsseln der Signatur verwendet werden soll. Sie müssen den öffentlichen Schlüsselteil des key pair verwenden, mit dem die Daten signiert wurden [NCryptSignHash](#).

pPaddingInfo[in, optional]

Ein Zeiger auf eine Struktur, die Füllinformationen enthält. Der Strukturtyp hängt vom dwFlags Wert ab. Verwenden Sie diesen Parameter nur mit asymmetrischen Schlüsseln. Setzen Sie ihn für andere Schlüsseltypen auf NULL.

pbHashValue[in]

Ein Zeiger auf einen Puffer, der den zu signierenden Hashwert enthält. Geben Sie die Puffergröße mit cbHashValue.

cbHashValue[in]

Die Größe des pbHashValue Puffers in Byte.

pbSignature[aus]

Die Adresse eines Puffers, der den signierten Hash der Daten enthält. Wird verwendet [NCryptSignHash](#), um diese Signatur zu erstellen. Geben Sie die Puffergröße mit ancSignature.

cbSignature[in]

Die Größe des pbSignature Puffers in Byte. Wird verwendet [NCryptSignHash](#), um die Signatur zu erstellen.

dwFlags[in]

Flags, um das Verhalten der Funktion zu ändern. Die erlaubten Flags hängen von Ihrem Schlüsseltyp ab. Verwenden Sie einen der folgenden Werte:

Wert	Bedeutung
NCRYPT_PAD_FLAG_PKCS1	Zeigt die verwendete Auffüllung der Signatur an. PKCS1 Legt fest pPaddingInfo , dass es auf eine BCrypt_PKCS1_PADDING_INFO Struktur zeigt.
NCRYPT_PAD_PSS_FLAG	Zeigt an, dass für die Signatur Padding des Probabilistic Signature Scheme (PSS) verwendet wurde. Legt fest, pPaddingInfo dass es auf eine Struktur zeigt. BCrypt_PSS_PADDING_INFO

Wert	Bedeutung
NCRYPT_SILENT_FLAG	Diese Flagge hat keine Wirkung.

Rückgabewert

Die Funktion gibt einen Statuscode zurück, der auf Erfolg oder Misserfolg hinweist.

Zu den gängigen Rückgabecodes gehören:

Rückgabecode	Description
ERROR_SUCCESS	Der Vorgang wurde erfolgreich abgeschlossen.
NTE_INVALID_PARAMETER	Ein oder mehrere Parameter sind nicht gültig.
NET_FAIL	Der Vorgang konnte nicht abgeschlossen werden.
NTE_INVALID_HANDLE	Das Handle in ist nicht gültig. hKey
NET_BAD_FLAGS	Der dwFlags Parameter enthält einen ungültigen Wert.
NTE_BAD_SIGNATURE	Die Signatur wurde nicht verifiziert.
NTE_BAD_KEY_STATE	Der Schlüsselstatus ist nicht gültig.
NTE_INTERNAL_ERROR	Bei der Überprüfung der Signatur ist ein interner Fehler aufgetreten.

Erweiterte Konfigurationen für KSP für AWS CloudHSM

Der AWS CloudHSM Key Storage Provider (KSP) umfasst die folgende erweiterte Konfiguration, die nicht Teil der allgemeinen Konfigurationen ist, die die meisten Kunden verwenden. Diese Konfigurationen bieten zusätzliche Funktionen.

- [SDK3 Kompatibilitätsmodus für KSP](#)

SDK3 Kompatibilitätsmodus für Key Storage Provider (KSP) für AWS CloudHSM

Key Storage Provider (KSP) implementiert verschiedene Ansätze für die HSM-Schlüsselinteraktion:

- Client SDK 5: Ermöglicht die direkte Kommunikation mit Schlüsseln, die im HSM gespeichert sind, sodass keine lokalen Referenzdateien erforderlich sind
- Client-SDK 3: Verwaltet lokale Dateien auf dem Windows-Server, die als Verweise auf im HSM gespeicherte Schlüssel dienen, und verwendet diese Dateien, um wichtige Operationen zu erleichtern

Für Kunden, die von Client SDK 3 auf Client SDK 5 migrieren, unterstützt die Aktivierung der Option für den SDK3 Kompatibilitätsmodus Operationen, die vorhandene Schlüsselreferenzdateien verwenden, während die zugrunde liegende HSM-Schlüsselspeicherarchitektur beibehalten wird.

Aktivieren Sie SDK3 den Kompatibilitätsmodus

Windows

Um den SDK3 Kompatibilitätsmodus für Key Storage Provider (KSP) für Client SDK 5 in Windows zu aktivieren

- Sie können den folgenden Befehl verwenden, um den SDK3 Kompatibilitätsmodus zu aktivieren:

```
PS C:\> & "C:\Program Files\Amazon\CloudHSM\bin\configure-ksp.exe" --enable-  
sdk3-compatibility-mode
```

Deaktivieren Sie den SDK3 Kompatibilitätsmodus

Windows

So deaktivieren Sie den SDK3 Kompatibilitätsmodus für Key Storage Provider (KSP) für Client SDK 5 in Windows

- Sie können den folgenden Befehl verwenden, um den SDK3 Kompatibilitätsmodus zu deaktivieren:

```
PS C:\> & "C:\Program Files\Amazon\CloudHSM\bin\configure-ksp.exe" --disable-  
sdk3-compatibility-mode
```

JCE-Anbieter für AWS CloudHSM Client SDK 5

Der AWS CloudHSM JCE-Anbieter ist eine Provider-Implementierung, die auf dem Java Cryptographic Extension (JCE) Provider-Framework basiert. JCE ermöglicht es Ihnen, kryptografische Operationen mit dem Java Development Kits (JDK) durchzuführen. In diesem Handbuch wird der AWS CloudHSM JCE-Anbieter manchmal als JCE-Anbieter bezeichnet. Verwenden Sie den JCE-Anbieter und das JDK, um kryptografische Operationen auf das HSM auszulagern. Informationen zur Problembehandlung finden Sie unter [Bekannte Probleme mit dem JCE SDK für AWS CloudHSM](#)

Informationen zur Verwendung von Client-SDK 3 finden Sie unter [Verwenden der vorherigen SDK-Version zum Arbeiten mit AWS CloudHSM](#).

Topics

- [Installieren Sie den JCE-Anbieter für AWS CloudHSM Client SDK 5](#)
- [Unterstützte Schlüsseltypen für den JCE-Anbieter für AWS CloudHSM Client SDK 5](#)
- [Grundlagen der Schlüsselverwaltung im JCE-Anbieter für AWS CloudHSM Client SDK 5](#)
- [Unterstützte Mechanismen für den JCE-Anbieter für AWS CloudHSM Client SDK 5](#)
- [Unterstützte Java-Schlüsselattribute für AWS CloudHSM Client SDK 5](#)
- [Codebeispiele für die AWS CloudHSM Softwarebibliothek für Java für Client SDK 5](#)
- [AWS CloudHSM JCE-Anbieter Javadocs](#)
- [AWS CloudHSM KeyStore Java-Klasse für Client SDK 5](#)
- [Erweiterte Konfigurationen für AWS CloudHSM JCE für Client SDK 5](#)

Installieren Sie den JCE-Anbieter für AWS CloudHSM Client SDK 5

Der JCE-Anbieter für AWS CloudHSM Client SDK 5 ist mit OpenJDK 8, OpenJDK 11, OpenJDK 17, OpenJDK 21 und OpenJDK 25 kompatibel. Sie können beide von der [OpenJDK-Website](#) herunterladen.

Verwenden Sie die folgenden Abschnitte, um den Anbieter zu installieren und Anmeldeinformationen für ihn bereitzustellen.

Note

Um einen einzelnen HSM-Cluster mit Client-SDK 5 auszuführen, müssen Sie zunächst die Einstellungen für die Haltbarkeit von Client-Schlüsseln verwalten, indem Sie die Einstellung

`disable_key_availability_check` auf `True` festlegen. Weitere Informationen finden Sie unter [Schlüsselsynchronisierung](#) und [Client-SDK-5-Configure-Tool](#).

Themen

- [Schritt 1: Installieren Sie den JCE-Anbieter](#)
- [Schritt 2: Geben Sie Anmeldeinformationen für den JCE-Anbieter ein](#)

Schritt 1: Installieren Sie den JCE-Anbieter

1. Verwenden Sie die folgenden Befehle, um den JCE-Anbieter herunterzuladen und zu installieren.

Amazon Linux 2023

Installieren Sie den JCE-Anbieter für Amazon Linux 2023 auf der x86_64-Architektur:

```
$ 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
```

Installieren Sie den JCE-Anbieter für Amazon Linux 2023 auf der ARM64 Architektur:

```
$ 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

Installieren Sie den JCE-Anbieter für Amazon Linux 2 auf der x86_64-Architektur:

```
$ 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
```

Installieren Sie den JCE-Anbieter für Amazon Linux 2 auf der ARM64 Architektur:

```
$ 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+)

Installieren Sie den JCE-Anbieter für RHEL 10 auf einer x86_64-Architektur:

```
$ 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
```

Installieren Sie den JCE-Anbieter für RHEL 10 auf der Architektur: ARM64

```
$ 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+)

Installieren Sie den JCE-Anbieter für RHEL 9 (9.2+) auf der x86_64-Architektur:

```
$ 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
```

Installieren Sie den JCE-Anbieter für RHEL 9 (9.2+) auf der folgenden Architektur: ARM64

```
$ 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+)

Installieren Sie den JCE-Anbieter für RHEL 8 auf einer x86_64-Architektur:

```
$ 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
```

Installieren Sie den JCE-Anbieter für RHEL 8 auf der Architektur: ARM64

```
$ 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 24.04 LTS

Installieren Sie den JCE-Anbieter für Ubuntu 24.04 LTS auf der x86_64-Architektur:

```
$ 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
```

Installieren Sie den JCE-Anbieter für Ubuntu 24.04 LTS auf der Architektur: ARM64

```
$ 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

Installieren Sie den JCE-Provider für Ubuntu 22.04 LTS auf der x86_64-Architektur:

```
$ 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
```

Installieren Sie den JCE-Provider für Ubuntu 22.04 LTS auf der Architektur: ARM64

```
$ 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

Installieren Sie den JCE-Anbieter für Windows Server auf der x86_64-Architektur, öffnen Sie ihn PowerShell als Administrator und führen Sie den folgenden Befehl aus:

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

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

2. Bootstrap für Client-SDK 5 Weitere Informationen zu Bootstrapping finden Sie unter [Bootstrap für das Client-SDK](#).
3. Suchen Sie die folgenden JCE-Anbieter-Dateien:

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

Schritt 2: Geben Sie Anmeldeinformationen für den JCE-Anbieter ein

Bevor Ihre Java-Anwendung ein HSM verwenden kann, muss das HSM die Anwendung zuerst authentifizieren. HSMs authentifizieren Sie sich entweder mit einer expliziten oder einer impliziten Anmeldemethode.

Explizite Anmeldemethode – Mit dieser Methode können Sie AWS CloudHSM -Anmeldeinformationen direkt in der Anwendung bereitstellen. Es verwendet die Methode aus dem [AuthProvider](#), bei der Sie einen CU-Benutzernamen und ein Passwort im Pin-Muster übergeben. Weitere Informationen finden Sie unter Codebeispiel für [Anmeldung bei einem HSM](#).

Implizite Anmeldemethode – Mit dieser Methode können Sie AWS CloudHSM -Anmeldeinformationen entweder in einer neuen Property-Datei, über Systemeigenschaften oder als Umgebungsvariablen festlegen.

- **Systemeigenschaften** – Anmeldeinformationen über Systemeigenschaften fest, wenn Sie Ihre Anwendung ausführen. Die folgenden Beispiele zeigen zwei verschiedene Möglichkeiten:

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>");
```


- Umgebungsvariablen – Anmeldeinformationen als Umgebungsvariablen festlegen.

 Note

Beim Setzen von Umgebungsvariablen müssen Sie alle Sonderzeichen maskieren, die von Ihrer Shell interpretiert werden könnten.

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>"
```

Anmeldeinformationen sind möglicherweise nicht verfügbar, wenn die Anwendung sie nicht bereitstellt oder wenn Sie eine Operation ausführen, bevor der HSM die Sitzung authentifiziert. In diesen Fällen sucht die CloudHSM-Softwarebibliothek für Java in der folgenden Reihenfolge nach den Anmeldeinformationen:

1. Systemeigenschaften
2. Umgebungsvariablen

Unterstützte Schlüsseltypen für den JCE-Anbieter für AWS CloudHSM Client SDK 5

Mit der AWS CloudHSM Softwarebibliothek für Java können Sie die folgenden Schlüsseltypen generieren.

Schlüsseltyp	Description
AES	Generieren der 128-, 192- und 256-Bit-AES-Schlüssel.

Schlüsseltyp	Description
Dreifaches DES (3DES, DESede)	Generieren Sie einen 192-Bit-Triple-DES-Schlüssel. *
EC	Generieren Sie EC-Schlüsselpaare – die NIST-Kurven secp224r1 (P-224), secp256r1 (P-256), secp256k1 (Blockchain), secp384r1 (P-384) und secp521r1 (P-521).
GENERIC_SECRET	Generieren Sie generische Geheimnisse mit 1 bis 800 Byte.
HMAC	Hash-Unterstützung für SHA1,, SHA224, SHA256, SHA384. SHA512
RSA	Generieren Sie RSA-Schlüssel mit 2048 bis 4096 Bit, in Schritten von 256 Bit.

* Gemäß den NIST-Richtlinien ist dies für Cluster im FIPS-Modus nach 2023 nicht zulässig. Für Cluster im Nicht-FIPS-Modus ist dies auch nach 2023 zulässig. Details dazu finden Sie unter [FIPS-140-Konformität: Mechanismus 2024 nicht mehr unterstützt](#).

Grundlagen der Schlüsselverwaltung im JCE-Anbieter für AWS CloudHSM Client SDK 5

Die Grundlagen der Schlüsselverwaltung im JCE-Anbieter umfassen den Import von Schlüsseln, den Export von Schlüsseln, das Laden von Schlüsseln per Handle oder das Löschen von Schlüsseln. Weitere Informationen zur Schlüsselverwaltung finden Sie im Codebeispiel [Schlüssel verwalten](#).

Weitere Codebeispiele für JCE-Anbieter finden Sie auch in [Codebeispiele](#).

Unterstützte Mechanismen für den JCE-Anbieter für AWS CloudHSM Client SDK 5

Dieses Thema enthält Informationen zu den unterstützten Mechanismen für den JCE-Anbieter mit dem AWS CloudHSM Client-SDK 5. Informationen zu den Schnittstellen und Engine-Klassen der Java Cryptography Architecture (JCA), die von unterstützt werden AWS CloudHSM, finden Sie in den folgenden Themen.

Themen

- [Schlüssel- und Schlüsselpaarfunktionen generieren](#)
- [Cipher-Funktionen](#)
- [Funktionen zum Signieren und Überprüfen](#)
- [Digest-Funktionen](#)
- [Funktionen des Hash-basierten Nachrichtenauthentifizierungscodes \(HMAC\)](#)
- [Funktionen des verschlüsselten Nachrichtenauthentifizierungscodes \(CMAC\)](#)
- [Die wichtigsten Funktionen der Vereinbarung](#)
- [Mithilfe von Schlüsselfabriken können Schlüssel in Schlüsselspezifikationen umgewandelt werden](#)
- [Anmerkungen zum Mechanismus](#)

Schlüssel- und Schlüsselpaarfunktionen generieren

Die AWS CloudHSM Softwarebibliothek für Java ermöglicht es Ihnen, die folgenden Operationen zum Generieren von Schlüssel- und Schlüsselpaarfunktionen zu verwenden.

- RSA
- EC
- AES
- DESede (Triple DES) ^{siehe Hinweis [1](#)}
- GenericSecret

Cipher-Funktionen

Die AWS CloudHSM Softwarebibliothek für Java unterstützt die folgenden Kombinationen aus Algorithmus, Modus und Auffüllung.

Algorithmus	Mode	Padding	Hinweise
AES	CBC	AES/CBC/N oPadding	Implementiert Cipher.EN CRYPT_MODE
		AES/CBC/P KCS5Padding	und Cipher.DE CRYPT_MODE .

Algorithmus	Mode	Padding	Hinweise
			Implementiert Cipher.UN WRAP_MODE for AES/CBC NoPadding
AES	ECB	AES/ECB/P KCS5Padding AES/ECB/N oPadding	Implementiert Cipher.EN CRYPT_MODE und Cipher.DE CRYPT_MODE .
AES	CTR	AES/CTR/N oPadding	Implementiert Cipher.EN CRYPT_MODE und Cipher.DE CRYPT_MODE .

Algorithmus	Mode	Padding	Hinweise
AES	GCM	AES/GCM/NoPadding	Implementiert <code>Cipher.WRAP_MODE</code>, <code>Cipher.ENCRYPT_MODE</code> und <code>Cipher.DECRYPT_MODE</code>. HSM ignoriert den Initialisierungsvektor (IV) der Anforderung während der AES-GCM-Datenverschlüsselung und verwendet stattdessen einen selbst generierten IV. Nach Abschluss der Operation müssen Sie <code>Cipher.getIV()</code> abrufen, um den IV zu erhalten.
AESWrap	ECB	AESWrap/ECB/NoPadding AESWrap/ECB/PKCS5Padding AESWrap/ECB/ZeroPadding	Implementiert <code>Cipher.WRAP_MODE</code> und <code>Cipher.UNWRAP_MODE</code>.

Algorithmus	Mode	Padding	Hinweise
DESede (Dreifaches DES)	CBC	DESede/CBC/ PKCS5Padding DESede/CBC/ NoPadding	Implementiert Cipher.EN CRYPT_MODE und Cipher.DE CRYPT_MODE . Eine bevorstehende Änderung finden Sie im Hinweis 1 unten.
DESede (Dreifaches DES)	ECB	DESede/ECB/ NoPadding DESede/ECB/ PKCS5Padding	Implementiert Cipher.EN CRYPT_MODE und Cipher.DE CRYPT_MODE . Eine bevorstehende Änderung finden Sie im Hinweis 1 unten.

Algorithmus	Mode	Padding	Hinweise
RSA	ECB	RSA/ECB/P KCS1Padding Hinweis 1	Implementiert Cipher.WR AP_MODE , Cipher.UN WRAP_MODE , Cipher.EN CRYPT_MODE und Cipher.DE CRYPT_MODE .
		RSA/ECB/0 AEPPadding	
		RSA/ECB/0 AEPWithSH A-1ANDMGF 1Padding	
		RSA/ECB/0 AEPWithSH A-224ANDM GF1Padding	
		RSA/ECB/0 AEPWithSH A-256ANDM GF1Padding	
		RSA/ECB/0 AEPWithSH A-384ANDM GF1Padding	
		RSA/ECB/0 AEPWithSH A-512ANDM GF1Padding	

Algorithmus	Mode	Padding	Hinweise
RSA	ECB	RSA/ECB/NoPadding	Implementiert Cipher.ENCRYPT_MODE und Cipher.DECRYPT_MODE .
RSAAESWrap	ECB	RSAAESWrap/ECB/OAEP Padding RSAAESWrap/ECB/OAEPWithSHA-1ANDMGF1Padding RSAAESWrap/ECB/OAEPWithSHA-224ANDMGF1Padding RSAAESWrap/ECB/OAEPWithSHA-256ANDMGF1Padding RSAAESWrap/ECB/OAEPWithSHA-384ANDMGF1Padding RSAAESWrap/ECB/OAEPWithSHA-512ANDMGF1Padding	Implementiert Cipher.WRAP_MODE und Cipher.UNWRAP_MODE .

Funktionen zum Signieren und Überprüfen

Die AWS CloudHSM Softwarebibliothek für Java unterstützt die folgenden Arten der Signatur und Überprüfung. Mit Client-SDK 5 und Signaturalgorithmen mit Hashing werden die Daten lokal in der Software gehasht, bevor sie zur Signatur/Überprüfung an das HSM gesendet werden. Das bedeutet, dass die Größe der Daten, die vom SDK gehasht werden können, unbegrenzt ist.

RSA-Signatortypen

- NONEwithRSA
- RSASSA-PSS
- SHA1withRSA
- SHA1withRSA/PSS
- SHA1withRSAandMGF1
- SHA224withRSA
- SHA224withRSAandMGF1
- SHA224withRSA/PSS
- SHA256withRSA
- SHA256withRSAandMGF1
- SHA256withRSA/PSS
- SHA384withRSA
- SHA384withRSAandMGF1
- SHA384withRSA/PSS
- SHA512withRSA
- SHA512withRSAandMGF1
- SHA512withRSA/PSS

ECDSA-Signatortypen

- NONEwithECDSA
- SHA1withECDSA
- SHA224withECDSA
- SHA256withECDSA

- SHA384withECDSA
- SHA512withECDSA

Digest-Funktionen

Die AWS CloudHSM Softwarebibliothek für Java unterstützt die folgenden Message Digests. Mit Client-SDK 5 werden die Daten lokal in der Software gehasht. Das bedeutet, dass die Größe der Daten, die vom SDK gehasht werden können, unbegrenzt ist.

- SHA-1
- SHA-224
- SHA-256
- SHA-384
- SHA-512

Funktionen des Hash-basierten Nachrichtenauthentifizierungscodes (HMAC)

Die AWS CloudHSM Softwarebibliothek für Java unterstützt die folgenden HMAC-Algorithmen.

- HmacSHA1(Maximale Datengröße in Byte: 16288)
- HmacSHA224(Maximale Datengröße in Byte: 16256)
- HmacSHA256(Maximale Datengröße in Byte: 16288)
- HmacSHA384(Maximale Datengröße in Byte: 16224)
- HmacSHA512(Maximale Datengröße in Byte: 16224)

Funktionen des verschlüsselten Nachrichtenauthentifizierungscodes (CMAC)

CMACs (Verschlüsselungsbasierte Nachrichtenauthentifizierungscodes) Erstellen Sie Nachrichtenauthentifizierungscodes (MACs) mithilfe einer Blockchiffre und eines geheimen Schlüssels. Sie unterscheiden HMACs sich dadurch, dass sie eine blocksymmetrische Schlüsselmethode anstelle einer Hashing-Methode verwenden. MACs

Die AWS CloudHSM Softwarebibliothek für Java unterstützt die folgenden CMAC-Algorithmen.

- AESCMAC

Die wichtigsten Funktionen der Vereinbarung

Die AWS CloudHSM Softwarebibliothek für Java unterstützt ECDH mit Key Derivation Functions (KDF). Die folgenden KDF-Typen werden unterstützt:

- `ECDHwithX963SHA1KDF` Unterstützt den X9.63 KDF-Algorithmus SHA1 [2](#)
- `ECDHwithX963SHA224KDF` Unterstützt den X9.63 KDF-Algorithmus SHA224 [2](#)
- `ECDHwithX963SHA256KDF` Unterstützt den X9.63 KDF-Algorithmus SHA256 [2](#)
- `ECDHwithX963SHA384KDF` Unterstützt den X9.63 KDF-Algorithmus SHA384 [2](#)
- `ECDHwithX963SHA512KDF` Unterstützt den X9.63 KDF-Algorithmus SHA512 [2](#)

Mithilfe von Schlüsselfabriken können Schlüssel in Schlüsselspezifikationen umgewandelt werden

Sie können Key Factorys verwenden, um Schlüssel in Schlüsselspezifikationen umzuwandeln. AWS CloudHSM hat zwei Arten von Schlüsselfabriken für JCE:

SecretKeyFactory: Wird verwendet, um symmetrische Schlüssel zu importieren oder abzuleiten. Mithilfe SecretKeyFactory können Sie einen unterstützten Schlüssel oder einen unterstützten Schlüssel übergeben, in KeySpec den symmetrische Schlüssel importiert oder abgeleitet werden sollen. AWS CloudHSM Im Folgenden finden Sie die unterstützten Spezifikationen für: KeyFactory

- Für die `generateSecret` Methode SecretKeyFactory von For werden die folgenden [KeySpec](#) Klassen unterstützt:
 - `KeyAttributesMap` kann verwendet werden, um Schlüsselbytes mit zusätzlichen Attributen als CloudHSM-Schlüssel zu importieren. Ein Beispiel finden Sie [hier](#).
 - [SecretKeySpec](#) kann verwendet werden, um eine symmetrische Schlüsselspezifikation als CloudHSM-Schlüssel zu importieren.
 - `AesCmacKdfParameterSpec` kann verwendet werden, um symmetrische Schlüssel mit einem anderen CloudHSM AES-Schlüssel abzuleiten.

Note

SecretKeyFactory [Die `translateKey` Methode verwendet jeden Schlüssel, der die Schlüsselschnittstelle implementiert.](#)

KeyFactory: Wird für den Import asymmetrischer Schlüssel verwendet. Mithilfe KeyFactory können Sie einen unterstützten Schlüssel oder einen unterstützten Schlüssel übergeben, in KeySpec den ein asymmetrischer Schlüssel importiert werden soll. AWS CloudHSM Weitere Informationen finden Sie in folgenden verwandten Ressourcen:

- Für die `generatePublic` Methode KeyFactory von For werden die folgenden [KeySpec](#)-Klassen unterstützt:
 - CloudHSM KeyAttributesMap für RSA und EC KeyTypes, einschließlich:
 - CloudHSM KeyAttributesMap für RSA und EC public. KeyTypes Ein Beispiel finden Sie [hier](#).
 - [X509 EncodedKeySpec](#) für RSA und EC Public Key
 - [RSAPublicKeySpec](#) für RSA Public Key
 - [ECPublicKeySpec](#) für EC Public Key
- Für die `generatePrivate` Methode KeyFactory von For werden die folgenden [KeySpec](#)-Klassen unterstützt:
 - CloudHSM KeyAttributesMap für RSA und EC KeyTypes, einschließlich:
 - CloudHSM KeyAttributesMap für RSA und EC public. KeyTypes Ein Beispiel finden Sie [hier](#).
 - [PKCS8EncodedKeySpec](#) sowohl für EC als auch für RSA Private Key
 - [RSAPrivateCrtKeySpec](#) für RSA Private Key
 - [ECPrivateKeySpec](#) für privaten EC-Schlüssel

KeyFactory Die `translateKey` Methode von For nimmt jeden Schlüssel auf, der das [Key Interface](#) implementiert.

Anmerkungen zum Mechanismus

[1] Gemäß den NIST-Richtlinien ist dies für Cluster im FIPS-Modus nach 2023 nicht zulässig. Für Cluster im Nicht-FIPS-Modus ist dies auch nach 2023 zulässig. Details dazu finden Sie unter [FIPS-140-Konformität: Mechanismus 2024 nicht mehr unterstützt](#).

[2] Die wichtigsten Ableitungsfunktionen (KDFs) sind in der [NIST-Sonderveröffentlichung](#) 800-56A Revision 3 spezifiziert.

Unterstützte Java-Schlüsselattribute für AWS CloudHSM Client SDK 5

Dieses Thema enthält Informationen zu unterstützten Java-Schlüsselattributen für das AWS CloudHSM Client SDK 5. Dieses Thema beschreibt, wie Sie eine proprietäre Erweiterung für den

JCE-Anbieter verwenden, um Schlüsselattribute festzulegen. Verwenden Sie diese Erweiterung, um unterstützte Schlüsselattribute und ihre Werte während der folgenden Vorgänge festzulegen:

- Schlüsselgenerierung
- Schlüsselimport

Beispiele für die Verwendung von Schlüsselattributen finden Sie unter [the section called “Codebeispiele”](#).

Themen

- [Grundlegendes zu Attributen](#)
- [Unterstützte Attribute](#)
- [Festlegen von Attributen für einen Schlüssel](#)

Grundlegendes zu Attributen

Mithilfe von Schlüsselattributen legen Sie fest, welche Aktionen für Schlüsselobjekte zulässig sind, einschließlich öffentlicher, privater oder geheimer Schlüssel. Schlüsselattribute und -werte definieren Sie bei der Erstellung von Schlüsselobjekten.

Die Java Cryptography Extension (JCE) gibt nicht an, wie Sie Werte für Schlüsselattribute festlegen sollten. Daher sind die meisten Aktionen standardmäßig zulässig. Im Gegensatz dazu definiert der PKCS# 11-Standard einen umfassenden Satz von Attributen mit restriktiveren Standardeinstellungen. Ab dem JCE-Anbieter 3.1 steht eine proprietäre Erweiterung AWS CloudHSM zur Verfügung, mit der Sie restriktivere Werte für häufig verwendete Attribute festlegen können.

Unterstützte Attribute

Sie können Werte für die Attribute festlegen, die in der folgenden Tabelle aufgeführt sind. Als bewährte Methode legen Sie nur Werte für Attribute fest, die Sie einschränken möchten. Wenn Sie keinen Wert angeben, wird der in der Tabelle unten angegebene Standardwert AWS CloudHSM verwendet. Eine leere Zelle in der Spalte „Standardwert“ gibt an, dass dem Attribut kein spezifischer Standardwert zugewiesen ist.

Attribut	Standardwert			Hinweise
	Symmetrischer Schlüssel	Öffentlicher Schlüssel im Schlüsselpaar	Privater Schlüssel im Schlüsselpaar	
DECRYPT	TRUE		TRUE	„True“ gibt an, dass Sie den Schlüssel zur Entschlüsselung eines beliebigen Puffers verwenden können. Sie legen dies im Allgemeinen auf „FALSE“ fest für einen Schlüssel, dessen WRAP auf wahr festgelegt ist.
DERIVE				Ermöglicht die Verwendung eines Schlüssels zur Ableitung anderer Schlüssel.
ENCRYPT	TRUE	TRUE		„True“ gibt an, dass Sie den Schlüssel zur Verschlüsselung eines beliebigen Puffers verwenden können.

Attribut	Standardwert			Hinweise
	Symmetrischer Schlüssel	Öffentlicher Schlüssel im Schlüsselpaar	Privater Schlüssel im Schlüsselpaar	
EXTRACTABLE	TRUE		TRUE	„True“ gibt an, dass Sie diesen Schlüssel aus dem HSM exportieren können.
ID				Ein benutzerdefinierter Wert, der zur Identifizierung des Schlüssels verwendet wird.
KEY_TYPE				Wird verwendet, um den Schlüsseltyp zu identifizieren (AES DESede, generisches Geheimnis, EC oder RSA).

Attribut	Standardwert			Hinweise
	Symmetrischer Schlüssel	Öffentlicher Schlüssel im Schlüsselpaar	Privater Schlüssel im Schlüsselpaar	
LABEL				Eine benutzerdefinierte Zeichenfolge, mit der Sie Schlüssel auf Ihrem HSM bequem identifizieren können. Um den bewährten Methoden zu folgen, verwenden Sie für jeden Schlüssel eine eindeutige Bezeichnung, damit er später leichter zu finden ist.
LOCAL				Weist auf einen vom HSM generierten Schlüssel hin.

Attribut	Standardwert			Hinweise
	Symmetrischer Schlüssel	Öffentlicher Schlüssel im Schlüsselpaar	Privater Schlüssel im Schlüsselpaar	
OBJECT_CLASS				Wird verwendet, um die Objektklasse eines Schlüssels (SecretKey, PublicKey oder PrivateKey) zu identifizieren.
PRIVATE	TRUE	TRUE	TRUE	„True“ gibt an, dass ein Benutzer erst auf den Schlüssel zugreifen darf, wenn der Benutzer authentifiziert ist. Aus Gründen der Übersichtlichkeit können Benutzer AWS CloudHSM erst auf Schlüssel zugreifen, wenn sie authentifiziert wurden, auch wenn dieses Attribut auf FALSE gesetzt ist.

Attribut	Standardwert			Hinweise
	Symmetrischer Schlüssel	Öffentlicher Schlüssel im Schlüsselpaar	Privater Schlüssel im Schlüsselpaar	
SIGN	TRUE		TRUE	„True“ gibt an, dass Sie den Schlüssel verwenden können, um einen Hashwert zu signieren. Für öffentliche und private Schlüssel, die Sie archiviert haben, ist dies im Allgemeinen auf „FALSE“ festgelegt.
SIZE				Ein Attribut, das die Größe eines Schlüssels definiert. Weitere Informationen zu den unterstützten Schlüsselgrößen finden Sie unter Unterstützte Mechanismen für das Client-SDK 5.

Attribut	Standardwert			Hinweise
	Symmetrischer Schlüssel	Öffentlicher Schlüssel im Schlüsselpaar	Privater Schlüssel im Schlüsselpaar	
TOKEN	FALSE	FALSE	FALSE	Ein permanenter Schlüssel, der HSMs im gesamten Cluster repliziert und in Backups enthalten ist. TOKEN = FALSCH impliziert einen flüchtigen Schlüssel, der automatisch gelöscht wird, wenn die Verbindung zum HSM unterbrochen oder abgemeldet wird.
UNWRAP	TRUE		TRUE	„True“ gibt an, dass Sie mit dem Schlüssel einen anderen Schlüssel entpacken (importieren) können.

Attribut	Standardwert			Hinweise
	Symmetrischer Schlüssel	Öffentlicher Schlüssel im Schlüsselpaar	Privater Schlüssel im Schlüsselpaar	
VERIFY	TRUE	TRUE		„True“ gibt an, dass Sie den Schlüssel verwenden können, um eine Signatur zu überprüfen. Für private Schlüssel ist dies im Allgemeinen auf „FALSE“ festgelegt.
WRAP	TRUE	TRUE		„True“ gibt an, dass Sie den Schlüssel zum Packen eines anderen Schlüssels verwenden können. Für private Schlüssel legen Sie dies in der Regel auf „FALSE“ fest.

Attribut	Standardwert			Hinweise
	Symmetrischer Schlüssel	Öffentlicher Schlüssel im Schlüsselpaar	Privater Schlüssel im Schlüsselpaar	
WRAP_WITH_TRUSTED	FALSE		FALSE	Wahr bedeutet, dass ein Schlüssel nur mit Schlüsseln, deren TRUSTED-Attribut auf wahr gesetzt ist, ein- und ausgepackt werden kann. Sobald bei einem Schlüssel WRAP_WITH_TRUSTED auf wahr gesetzt wurde, ist dieses Attribut schreibgeschützt und kann nicht auf Falsch gesetzt werden. Weitere Informationen zum Trust-Wrapping finden Sie unter Verwenden vertrauenswürdigem Schlüssel zur Steuerung

Attribut	Standardwert			Hinweise
	Symmetrischer Schlüssel	Öffentlicher Schlüssel im Schlüsselpaar	Privater Schlüssel im Schlüsselpaar	
				von Schlüssel-Unwraps.

 Note

Sie erhalten eine breitere Unterstützung für Attribute in der PKCS #11-Bibliothek. Weitere Informationen finden Sie unter [Unterstützte PKCS #11-Attribute](#).

Festlegen von Attributen für einen Schlüssel

`KeyAttributesMap` ist ein Java-Map-ähnliches Objekt, mit dem Sie Attributwerte für Schlüsselobjekte festlegen können. Die Methoden für `KeyAttributesMap` funktionieren ähnlich den Methoden, die für die Java-Map-Manipulation verwendet werden.

Sie haben zwei Optionen, um benutzerdefinierte Werte für Attribute festzulegen:

- Verwenden Sie die in der folgenden Tabelle aufgeführten Methoden
- Verwenden Sie Builder-Muster, die später in diesem Dokument gezeigt werden

Attributzuordnungsobjekte unterstützen die folgenden Methoden zum Festlegen von Attributen:

Operation	Rückgabewert	KeyAttributesMap - Methode
Abrufen des Werts eines Schlüsselattributs für einen vorhandenen Schlüssel	Objekt (das den Wert enthält) oder null	<code>get(keyAttribute)</code>
Eingeben des Werts eines Schlüsselattributs	Der vorherige Wert, der dem Schlüsselattribut zugeordnet	<code>put(keyAttribute, Wert)</code>

Operation	Rückgabewert	KeyAttributesMap - Methode
	t ist, oder null, wenn keine Zuordnung für ein Schlüsselattribut vorhanden ist	
Eingeben von Werten für mehrere Schlüsselattribute	–	putAll () keyAttributesMap
Entfernen eines Schlüssel-Wert-Paares aus der Attributzuordnung	Der vorherige Wert, der dem Schlüsselattribut zugeordnet ist, oder null, wenn keine Zuordnung für ein Schlüsselattribut vorhanden ist	remove(keyAttribute)

 Note

Alle Attribute, die Sie nicht explizit bestimmen, werden auf die Standardwerte festgelegt, die in der vorherigen Tabelle in [the section called “Unterstützte Attribute”](#) aufgeführt sind.

Festlegen von Attributen für ein Schlüsselpaar

Verwenden Sie die Java-Klasse `KeyPairAttributesMap`, um Schlüsselattribute für ein Schlüsselpaar zu verarbeiten. `KeyPairAttributesMap` fasst zwei `KeyAttributesMap`-Objekte zusammen, eines für einen öffentlichen Schlüssel und eines für einen privaten Schlüssel.

Um einzelne Attribute für den öffentlichen und den privaten Schlüssel separat festzulegen, können Sie die `put()`-Methode für das entsprechende `KeyAttributes`-Zuordnungsobjekt für diesen Schlüssel verwenden. Verwenden Sie die `getPublic()`-Methode, um die Attributzuordnung für den öffentlichen Schlüssel abzurufen, und die `getPrivate()`-Methode, um die Attributzuordnung für den privaten Schlüssel abzurufen. Geben Sie den Wert mehrerer Schlüsselattribute für öffentliche und private Schlüsselpaare zusammen ein, indem Sie die `putAll()`-Methode für eine Attributzuordnung von Schlüsselpaaren als Argument verwenden.

Codebeispiele für die AWS CloudHSM Softwarebibliothek für Java für Client SDK 5

Dieses Thema enthält Ressourcen und Informationen zu Java-Codebeispielen für das AWS CloudHSM Client SDK 5.

Voraussetzungen

Bevor Sie die Beispiele ausführen, müssen Sie die Umgebung einrichten:

- Installieren und konfigurieren Sie den [Java Cryptographic Extension \(JCE\)](#)-Anbieter.
- Legen Sie einen gültigen [HSM-Benutzernamen und ein Passwort fest](#). Crypto-Benutzer (CU)-Berechtigungen sind ausreichend für diese Aufgaben. Ihre Anwendung verwendet diese Anmeldeinformationen, um sich für die einzelnen Beispiele beim HSM anzumelden.
- Entscheiden Sie, wie Anmeldeinformationen für den [JCE-Anbieter](#) bereitgestellt werden sollen.

Codebeispiele

Die folgenden Codebeispiele zeigen Ihnen, wie Sie den [AWS CloudHSM JCE-Anbieter](#) verwenden, um grundlegende Aufgaben auszuführen. Weitere Codebeispiele finden Sie unter [GitHub](#).

- [Anmeldung bei einem HSM](#)
- [Verwalten von Schlüsseln](#)
- [Generieren von symmetrischen Schlüsseln](#)
- [Generieren von asymmetrischen Schlüsseln](#)
- [Verschlüsseln und Entschlüsseln mit AES-GCM](#)
- [Verschlüsseln und Entschlüsseln mit AES-CTR](#)
- [Mit DESede -ECB verschlüsseln und entschlüsseln](#), siehe Hinweis [1](#)
- [Signieren und Verifizieren mit RSA-Schlüsseln](#)
- [Signieren und Verifizieren mit EC-Schlüsseln](#)
- [Unterstützte Schlüsselattribute verwenden](#)
- [Verwenden des CloudHSM-Schlüsselspeichers](#)

[1] Gemäß den NIST-Richtlinien ist dies für Cluster im FIPS-Modus nach 2023 nicht zulässig. Für Cluster im Nicht-FIPS-Modus ist dies auch nach 2023 zulässig. Details dazu finden Sie unter [FIPS-140-Konformität: Mechanismus 2024 nicht mehr unterstützt](#).

AWS CloudHSM JCE-Anbieter Javadocs

Verwenden Sie den JCE-Anbieter Javadocs, um Nutzungsinformationen zu Java-Typen und -Methoden abzurufen, die im AWS CloudHSM JCE SDK definiert sind. Informationen zum Herunterladen der neuesten Javadocs für AWS CloudHSM finden Sie im [AWS CloudHSM neueste Version des Client-SDK](#) Abschnitt auf der Download-Seite.

Sie können Javadocs in eine integrierte Entwicklungsumgebung (IDE) importieren oder sie in einem Webbrowser anzeigen.

AWS CloudHSM KeyStore Java-Klasse für Client SDK 5

Die AWS CloudHSM KeyStore Klasse stellt einen PKCS12 Schlüsselspeicher für spezielle Zwecke bereit. Dieser Schlüsselspeicher kann Zertifikate zusammen mit Ihren Schlüsseldaten speichern und sie mit den in AWS CloudHSM gespeicherten Schlüsseldaten korrelieren. Die AWS CloudHSM KeyStore Klasse implementiert das KeyStore Service Provider Interface (SPI) der Java Cryptography Extension (JCE). [Weitere Informationen zur Verwendung KeyStore finden Sie unter Class. KeyStore](#)

Note

Da es sich bei Zertifikaten um öffentliche Informationen handelt und um die Speicherkapazität für kryptografische Schlüssel zu maximieren, AWS CloudHSM wird das Speichern von Zertifikaten auf HSMs nicht unterstützt.

Wählen Sie den entsprechenden Schlüsselspeicher für AWS CloudHSM Client SDK 5

Der Anbieter AWS CloudHSM Java Cryptographic Extension (JCE) bietet ein spezielles AWS CloudHSM an. KeyStore Die AWS CloudHSM KeyStore Klasse unterstützt das Auslagern wichtiger Operationen in das HSM, die lokale Speicherung von Zertifikaten und zertifikatsbasierte Operationen.

Laden Sie das spezielle CloudHSM KeyStore wie folgt:

```
KeyStore ks = KeyStore.getInstance("CloudHSM")
```

Initialisieren Sie das Client-SDK 5 AWS CloudHSM KeyStore

Melden Sie sich genauso AWS CloudHSM KeyStore an, wie Sie sich beim JCE-Anbieter anmelden. Sie können entweder Umgebungsvariablen oder die Systemeigenschaftendatei verwenden, und Sie

sollten sich anmelden, bevor Sie CloudHSM KeyStore verwenden. Ein Beispiel für die Anmeldung bei einem HSM mit dem JCE-Anbieter finden Sie unter [Anmeldung an HSM](#).

Falls gewünscht, können Sie ein Passwort angeben, um die lokale PKCS12 Datei zu verschlüsseln, die Schlüsselspeicherdaten enthält. Wenn Sie den AWS CloudHSM Keystore erstellen, legen Sie das Passwort fest und geben es an, wenn Sie die Methoden `load`, `set` und `get` verwenden.

Instanzieren Sie ein neues CloudHSM-Objekt KeyStore wie folgt:

```
ks.load(null, null);
```

Schreiben Sie Schlüsselspeicherdaten mit der `store`-Methode in eine Datei. Von da an können Sie den vorhandenen Schlüsselspeicher mit der `load`-Methode mit der Quelldatei und dem Passwort wie folgt laden:

```
ks.load(inputStream, password);
```

Verwenden Sie unser Client SDK 5 AWS CloudHSM KeyStore AWS CloudHSM

AWS CloudHSM KeyStore entspricht der [KeyStoreJCE-Klassenspezifikation](#) und bietet die folgenden Funktionen.

- `load`

Lädt den Schlüsselspeicher aus dem angegebenen Eingabe-Stream. Wenn beim Speichern des Schlüsselspeichers ein Passwort festgelegt wurde, muss dasselbe Passwort angegeben werden, damit das Laden erfolgreich ist. Setzen Sie beide Parameter auf `null`, um einen neuen leeren Schlüsselspeicher zu initialisieren.

```
KeyStore ks = KeyStore.getInstance("CloudHSM");  
ks.load(inputStream, password);
```

- `aliases`

Gibt eine Aufzählung der Aliasnamen aller Einträge in der angegebenen Schlüsselspeicherinstanz zurück. Zu den Ergebnissen gehören Objekte, die lokal in der PKCS12 Datei gespeichert sind, und Objekte, die sich auf dem HSM befinden.

Beispiel-Code:

```
KeyStore ks = KeyStore.getInstance("CloudHSM");
for(Enumeration<String> entry = ks.aliases(); entry.hasMoreElements();) {
    String label = entry.nextElement();
    System.out.println(label);
}
```

- **containsalias**

Gibt „true“ zurück, wenn der Schlüsselspeicher Zugriff auf mindestens ein Objekt mit dem angegebenen Alias hat. Der Schlüsselspeicher überprüft Objekte, die lokal in der PKCS12 Datei gespeichert sind, und Objekte, die sich auf dem HSM befinden.

- **deleteEntry**

Löscht einen Zertifikatseintrag aus der lokalen PKCS12 Datei. Das Löschen von Schlüsseldaten, die in einem HSM gespeichert sind, wird mit dem nicht unterstützt. AWS CloudHSM KeyStore Sie können Schlüssel mit der `destroy`-Methode der [Destroyable](#)-Schnittstelle löschen.

```
((Destroyable) key).destroy();
```

- **getCertificate**

Gibt das Zertifikat zurück, das einem Alias zugeordnet ist, falls verfügbar. Wenn der Alias nicht existiert oder auf ein Objekt verweist, das kein Zertifikat ist, gibt die Funktion NULL zurück.

```
KeyStore ks = KeyStore.getInstance("CloudHSM");
Certificate cert = ks.getCertificate(alias);
```

- **getCertificateAlias**

Gibt den Namen (Alias) des ersten Schlüsselspeichereintrags zurück, dessen Daten mit dem angegebenen Zertifikat übereinstimmen.

```
KeyStore ks = KeyStore.getInstance("CloudHSM");
String alias = ks.getCertificateAlias(cert);
```

- **getCertificateChain**

Gibt die Zertifikatkette zurück, die dem angegebenen Alias zugeordnet ist. Wenn der Alias nicht existiert oder auf ein Objekt verweist, das kein Zertifikat ist, gibt die Funktion NULL zurück.

- **getCreationDate**

Gibt das Erstellungsdatum des durch den angegebenen Alias identifizierten Eintrags zurück. Wenn kein Erstellungsdatum verfügbar ist, gibt die Funktion das Datum zurück, an dem das Zertifikat gültig wurde.

- `getKey`

`getKey` wird an das HSM übergeben und gibt ein Schlüsselobjekt zurück, das dem angegebenen Label entspricht. Da es das HSM `getKey` direkt abfragt, kann es für jeden Schlüssel auf dem HSM verwendet werden, unabhängig davon, ob er von der generiert wurde. `KeyStore`

```
Key key = ks.getKey(keyLabel, null);
```

- `isCertificateEntry`

Überprüft, ob der Eintrag mit dem angegebenen Alias einen Zertifikateintrag darstellt.

- `isKeyEntry`

Überprüft, ob der Eintrag mit dem angegebenen Alias einen Schlüsseleintrag darstellt. Die Aktion durchsucht sowohl die PKCS12 Datei als auch das HSM nach dem Alias.

- `setCertificateEntry`

Weist dem angegebenen Alias das angegebene Zertifikat zu. Wenn der angegebene Alias bereits verwendet wird, um einen Schlüssel oder ein Zertifikat zu identifizieren, wird ein `KeyStoreException` ausgelöst. Sie können JCE-Code verwenden, um das Schlüsselobjekt abzurufen, und dann die `KeyStore setKeyEntry` Methode verwenden, um das Zertifikat dem Schlüssel zuzuordnen.

- `setKeyEntry` mit `byte[]`-Schlüssel

Diese API wird derzeit vom Client-SDK 5 nicht unterstützt.

- `setKeyEntry` mit Key-Objekt

Weist den angegebenen Schlüssel dem angegebenen Alias zu und speichert ihn im HSM. Wenn der Schlüssel noch nicht im HSM vorhanden ist, wird er als extrahierbarer Sitzungsschlüssel in das HSM importiert.

Wenn das Key-Objekt vom Typ `PrivateKey` ist, muss es von einer entsprechenden Zertifikatkette begleitet werden.

Wenn der Alias bereits vorhanden ist, wird ein `SetKeyEntry`- Aufruf ausgelöst, und `KeyStoreException` verhindert, dass der Schlüssel überschrieben wird. Wenn der Schlüssel überschrieben werden muss, verwenden Sie hierfür KMU oder JCE.

- `engineSize`

Gibt die Anzahl der Einträge im Schlüsselspeicher zurück.

- `store`

Speichert den Schlüsselspeicher für den angegebenen Ausgabestrom als PKCS12 Datei und sichert ihn mit dem angegebenen Passwort. Darüber hinaus bleiben alle geladenen Schlüssel (die über `setKey`-Aufrufe gesetzt werden) bestehen.

Erweiterte Konfigurationen für AWS CloudHSM JCE für Client SDK 5

Der AWS CloudHSM JCE-Anbieter umfasst die folgenden erweiterten Konfigurationen, die nicht Teil der allgemeinen Konfigurationen sind, die die meisten Kunden verwenden.

- [Verbindung zu mehreren Clustern herstellen](#)
- [Schlüsselextraktion mit JCE](#)
- [Versuchen Sie erneut, die Konfiguration für JCE zu konfigurieren](#)

Verbindung zu mehreren AWS CloudHSM Clustern mit dem JCE-Anbieter herstellen

Diese Konfiguration ermöglicht es einer einzelnen Client-Instance, mit mehreren AWS CloudHSM Clustern zu kommunizieren. Wenn Sie eine einzelne Instance haben, die nur mit einem einzigen Cluster kommuniziert, kann diese Konfiguration in einigen Anwendungsfällen zu Kosteneinsparungen führen. Die `CloudHsmProvider` Klasse AWS CloudHSM ist die Implementierung der [Provider-Klasse von Java Security](#). Jede Instanz dieser Klasse stellt eine Verbindung zu Ihrem gesamten AWS CloudHSM Cluster dar. Sie instanziiieren diese Klasse und fügen sie der Liste der Java-Security-Anbieter hinzu, sodass Sie mit ihr über Standard-JCE-Klassen interagieren können.

Das folgende Beispiel instanziiert diese Klasse und fügt sie der Liste der Java-Security-Anbieter hinzu:

```
if (Security.getProvider(CloudHsmProvider.PROVIDER_NAME) == null) {
```

```
Security.addProvider(new CloudHsmProvider());  
}
```

`CloudHsmProvider` kann auf zwei Arten konfiguriert werden:

1. Mit Datei konfigurieren (Standardkonfiguration)
2. Mit Code konfigurieren

In den folgenden Themen werden diese Konfigurationen und das Herstellen einer Verbindung zu mehreren Clustern beschrieben.

Themen

- [Konfigurieren Sie die AWS CloudHSM `CloudHsmProvider` Klasse mit einer Datei \(Standardkonfiguration\)](#)
- [Konfigurieren Sie die AWS CloudHSM `CloudHsmProvider` Klasse mithilfe von Code](#)
- [Connect zu mehreren AWS CloudHSM Clustern her](#)

Konfigurieren Sie die AWS CloudHSM **`CloudHsmProvider`** Klasse mit einer Datei (Standardkonfiguration)

Die AWS CloudHSM `CloudHsmProvider` Klasse wird standardmäßig mit einer Datei konfiguriert.

Wenn Sie `CloudHsmProvider` mit dem Standardkonstruktor instanziiieren, sucht er unter Linux standardmäßig nach einer Konfigurationsdatei im `/opt/cloudhsm/etc/cloudhsm-jce.cfg`-Pfad. Diese Konfigurationsdatei kann mit der Datei `configure-jce` konfiguriert werden.

Ein mit dem Standardkonstruktor erstelltes Objekt verwendet den standardmäßigen CloudHSM-Anbiaternamen `CloudHSM`. Der Anbietername ist nützlich, um mit JCE zu interagieren und ihm mitzuteilen, welcher Anbieter für verschiedene Operationen verwendet werden soll. Ein Beispiel für die Verwendung des CloudHSM-Anbiaternamens für den Chiffriervorgang lautet wie folgt:

```
Cipher cipher = Cipher.getInstance("AES/GCM/NoPadding", "CloudHSM");
```

Konfigurieren Sie die AWS CloudHSM **`CloudHsmProvider`** Klasse mithilfe von Code

Ab Version 5.8.0 des Client SDK können Sie die AWS CloudHSM `CloudHsmProvider` Klasse auch mithilfe von Java-Code konfigurieren. Dazu verwenden Sie ein Objekt der

`CloudHsmProviderConfig`-Klasse. Sie können `CloudHsmProviderConfigBuilder` verwenden, um dieses Objekt zu erstellen.

`CloudHsmProvider` hat einen anderen Konstruktor, der das `CloudHsmProviderConfig`-Objekt aufnimmt, wie das folgende Beispiel zeigt.

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);
```

In diesem Beispiel lautet der Name des JCE-Anbieters. `CloudHsmCluster1` Dies ist der Name, den die Anwendung dann verwenden kann, um mit JCE zu interagieren:

Example

```
Cipher cipher = Cipher.getInstance("AES/GCM/NoPadding", "CloudHsmCluster1");
```

Alternativ können Anwendungen auch das oben erstellte Anbieter-Objekt verwenden, um JCE mitzuteilen, dass es diesen Anbieter für die Operation verwenden soll:

```
Cipher cipher = Cipher.getInstance("AES/GCM/NoPadding", provider);
```

Wenn mit der `withClusterUniqueIdentifier`-Methode kein eindeutiger Bezeichner angegeben wird, wird ein zufällig generierter Anbietername für Sie erstellt. Um diesen zufällig generierten Bezeichner zu erhalten, können Anwendungen `provider.getName()` aufrufen, um den Bezeichner abzurufen.

Connect zu mehreren AWS CloudHSM Clustern her

Jedes `CloudHsmProvider` steht für eine Verbindung zu Ihrem AWS CloudHSM Cluster. Wenn Sie mit einem anderen Cluster aus derselben Anwendung kommunizieren möchten, können Sie ein anderes Objekt von `CloudHsmProvider` mit Konfigurationen für Ihren anderen Cluster erstellen

und mit diesem anderen Cluster entweder über das Anbieterobjekt oder über den Anbieternamen interagieren, wie im folgenden Beispiel gezeigt.

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);
}
```

Nachdem Sie beide Anbieter (beide Cluster) oben konfiguriert haben, können Sie entweder über das Anbieterobjekt oder über den Anbieternamen mit ihnen interagieren.

Ausgehend von diesem Beispiel, das zeigt, wie man mit jemandem spricht `cluster1`, könnten Sie das folgende Beispiel für eine AES/GCM/NoPadding Operation verwenden:

```
Cipher cipher = Cipher.getInstance("AES/GCM/NoPadding", provider1);
```


Und in derselben Anwendung zur Generierung von „AES“-Schlüsseln auf dem zweiten Cluster mit dem Anbieternamen könnten Sie auch das folgende Beispiel verwenden:

```
Cipher cipher = Cipher.getInstance("AES/GCM/NoPadding", provider2.getName());
```

Schlüsselextraktion mit JCE für AWS CloudHSM

Die Java Cryptography Extension (JCE) verwendet eine Architektur, mit der verschiedene Kryptografie-Implementierungen integriert werden können. AWS CloudHSM liefert einen solchen JCE-Anbieter aus, der kryptografische Operationen an das HSM auslagert. Damit die meisten anderen JCE-Anbieter mit Schlüsseln arbeiten können, die in AWS CloudHSM gespeichert sind, müssen sie die Schlüsselbytes von Ihnen HSMs im Klartext in den Speicher Ihres Computers extrahieren, damit sie sie verwenden können. HSMs Erlauben normalerweise nur das Extrahieren von Schlüsseln als verpackte Objekte, nicht als Klartext. Um Anwendungsfälle der anbieterübergreifenden Integration zu unterstützen, ist jedoch eine optionale Konfigurationsoption möglich, mit der die Extraktion der Schlüsselbytes im Klartext AWS CloudHSM ermöglicht wird.

Important

JCE lagert Operationen immer dann aus, AWS CloudHSM wenn der AWS CloudHSM CloudHSM-Anbieter angegeben oder ein AWS CloudHSM Schlüsselobjekt verwendet wird. Sie müssen Schlüssel nicht im Klartext extrahieren, wenn Sie erwarten, dass Ihre Operation innerhalb des HSM stattfindet. Die Schlüsselextraktion im Klartext ist nur erforderlich, wenn Ihre Anwendung aufgrund von Einschränkungen durch eine Drittanbieterbibliothek oder einen JCE-Anbieter keine sicheren Mechanismen wie das sogenannte Wrapping und Unwrapping eines Schlüssels verwenden kann.

Der AWS CloudHSM JCE-Anbieter ermöglicht standardmäßig das Extrahieren von öffentlichen Schlüsseln, sodass er mit externen JCE-Anbietern funktioniert. Die folgenden Methoden sind immer zulässig:

Klasse	Methode	Format (getEncoded)
EcPublicKey	getEncoded()	X.509
	getW()	–

Klasse	Methode	Format (getEncoded)
RSAPublicSchlüssel	getEncoded()	X.509
	getPublicExponent()	–
CloudHsmRsaPrivateCrtKey	getPublicExponent()	–

Der AWS CloudHSM JCE-Anbieter erlaubt standardmäßig keine Extraktion von Schlüsselbytes im Klartext für private oder geheime Schlüssel. Wenn Ihr Anwendungsfall dies erfordert, können Sie die Extraktion von Schlüsselbytes im Klartext für private oder geheime Schlüssel unter den folgenden Bedingungen aktivieren:

1. Das EXTRACTABLE-Attribut für private und geheime Schlüssel ist auf true gesetzt.
 - Standardmäßig ist das EXTRACTABLE-Attribut für private und geheime Schlüssel auf true gesetzt. EXTRACTABLE-Schlüssel sind Schlüssel, die aus dem HSM exportiert werden dürfen. Weitere Informationen finden Sie unter Unterstützte Java-Attribute für [Client-SDK 5](#).
2. Das WRAP_WITH_TRUSTED-Attribut für private und geheime Schlüssel ist auf falsch gesetzt.
 - getEncoded, getPrivateExponent und getS können nicht mit privaten Schlüsseln verwendet werden, die nicht in Klartext exportiert werden können. WRAP_WITH_TRUSTED erlaubt nicht, dass Ihre privaten Schlüssel in Klartext aus dem HSM exportiert werden. Weitere Informationen finden Sie unter [Verwenden vertrauenswürdiger Schlüssel zur Steuerung des Entpackens von Schlüsseln](#).

Erlaubt dem JCE-Anbieter, geheime private Schlüssel aus zu extrahieren AWS CloudHSM

Gehen Sie wie folgt vor, damit der AWS CloudHSM JCE-Anbieter Ihre privaten Schlüssel extrahieren kann.

Important

Diese Konfigurationsänderung ermöglicht das Extrahieren aller EXTRACTABLE-Schlüsselbytes aus Ihrem HSM-Cluster im Klartext. Aus Sicherheitsgründen sollten Sie erwägen, Schlüssel mithilfe von [Methoden zum Wrapping von Schlüsseln](#) sicher aus dem HSM zu extrahieren. Dadurch wird ein unbeabsichtigtes Extrahieren Ihrer Schlüsselbytes aus dem HSM verhindert.

1. Verwenden Sie die folgenden Befehle, um das Extrahieren Ihrer privaten oder geheimen Schlüssel in JCE zu ermöglichen:

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. Sobald Sie die Extraktion Ihres eindeutigen Schlüssels aktivieren, sind die folgenden Methoden zur Extraktion privater Schlüssel in den Speicher aktiviert.

Klasse	Methode	Format (getEncoded)
Key (Schlüssel)	getEncoded()	RAW
ECPrivateSchlüssel	getEncoded()	PKCS#8
	getS()	–
RSAPrivateCrtKey	getEncoded()	X.509
	getPrivateExponent()	–
	getPrimeP()	–
	getPrimeQ()	–
	getPrimeExponentP ()	–
	getPrimeExponentQ ()	–
	getCrtCoefficient()	–

Wenn Sie das Standardverhalten wiederherstellen und JCE nicht erlauben möchten, Schlüssel im Klartext zu exportieren, führen Sie den folgenden Befehl aus:

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
```

Befehle für JCE wiederholen für AWS CloudHSM

AWS CloudHSM Das Client-SDK 5.8.0 und höher verfügen über eine integrierte automatische Wiederholungsstrategie, mit der HSM-gedrosselte Operationen von der Clientseite aus wiederholt werden. Wenn ein HSM Operationen drosselt, weil es mit der Ausführung früherer Operationen zu beschäftigt ist und keine weiteren Anfragen annehmen kann, versucht der Client, gedrosselte Operationen bis zu dreimal zu wiederholen, während er SDKs sich exponentiell zurückzieht. Diese automatische Wiederholungsstrategie kann auf einen von zwei Modi eingestellt werden: aus und Standard.

- **aus:** Das Client-SDK führt bei gedrosselten Vorgängen durch das HSM keine Wiederholungsstrategie durch.
- **Standard:** Dies ist der Standardmodus für das Client-SDK 5.8.0 und höher. In diesem Modus wiederholt der Client automatisch gedrosselte Operationen, indem er SDKs sich exponentiell zurückzieht.

Weitere Informationen finden Sie unter [HSM-Drosselung](#).

Stellen Sie den Modus für Wiederholungsbefehle auf Aus

Linux

So setzen Sie Wiederholungsbefehle auf off für Client-SDK 5 unter Linux

- Sie können den folgenden Befehl verwenden, um die Konfiguration der Wiederholungen auf den off-Modus zu setzen:

```
$ sudo /opt/cloudhsm/bin/configure-jce --default-retry-mode off
```

Windows

So setzen Sie Wiederholungsbefehle auf off für Client-SDK 5 unter Windows

- Sie können den folgenden Befehl verwenden, um die Konfiguration der Wiederholungen auf den off-Modus zu setzen:

```
PS C:\> & "C:\Program Files\Amazon\CloudHSM\bin\configure-jce.exe" --default-retry-mode off
```

Verwenden der vorherigen SDK-Version zum Arbeiten mit AWS CloudHSM

 Die SDK-Versionen 5.8.0 und früher haben das Ende der Unterstützung erreicht. Nach dem 31. März 2025 wird die Dokumentation für die SDK-Versionen 3.4.4 und früher nicht mehr verfügbar sein.

AWS CloudHSM umfasst zwei Hauptversionen des Client-SDK:

- Client-SDK 5: Dies ist unser neuestes und standardmäßiges Client-SDK. Informationen zu den Vorteilen und Nutzen, die es bietet, finden Sie unter [Vorteile von AWS CloudHSM Client SDK 5](#).

- Client-SDK 3: Dies ist unser älteres Client-SDK. Es enthält einen vollständigen Satz von Komponenten für die Kompatibilität von plattform- und sprachbasierten Anwendungen und Verwaltungstools.

Anweisungen zur Migration von Client SDK 3 auf Client SDK 5 finden Sie unter [Migration von AWS CloudHSM Client SDK 3 zu Client SDK 5](#).

In diesem Thema wird das Client SDK 3 beschrieben. Informationen darüber, welche Version des Client-SDK Sie verwenden, finden Sie unter [Überprüfen Sie Ihre AWS CloudHSM Client-SDK-Version](#).

Zum Herunterladen siehe [Downloads](#).

Topics

- [Aktualisieren Sie das AWS CloudHSM Client SDK 3 unter Linux](#)
- [AWS CloudHSM Unterstützte Plattformen vom Client SDK 3](#)
- [PKCS #11 -Bibliothek für AWS CloudHSM Client SDK 3](#)
- [OpenSSL Dynamic Engine für AWS CloudHSM Client SDK 3](#)
- [JCE-Anbieter für AWS CloudHSM Client SDK 3](#)
- [Kryptografie-API: Next Generation \(CNG\) und Key Storage Providers \(KSP\) für AWS CloudHSM](#)

Aktualisieren Sie das AWS CloudHSM Client SDK 3 unter Linux

 Die SDK-Versionen 5.8.0 und früher haben das Ende der Unterstützung erreicht. Nach dem 31. März 2025 wird die Dokumentation für die SDK-Versionen 3.4.4 und früher nicht mehr verfügbar sein.

Bei AWS CloudHSM Client SDK 3.1 und höher müssen die Version des Client-Daemons und aller Komponenten, die Sie installieren, für das Upgrade übereinstimmen. Für alle Linux-basierten Systeme müssen Sie einen einzigen Befehl verwenden, um den Client-Daemon stapelweise mit derselben Version der PKCS-#11-Bibliothek, dem Java Cryptographic Extension-Anbieter (JCE) oder der OpenSSL Dynamic Engine zu aktualisieren. Diese Anforderung gilt nicht für Windows-basierte Systeme, da die Binärdateien für die CNG- und KSP-Anbieter bereits im Paket des Client-Daemon enthalten sind.

So ermitteln Sie die Client-Daemon-Version

- Verwenden Sie auf einem Red Hat-basierten Linux-System (einschließlich Amazon Linux und CentOS) den folgenden Befehl:

```
rpm -qa | grep ^cloudhsm
```

- Verwenden Sie auf einem Debian-basierten Linux-System den folgenden Befehl:

```
apt list --installed | grep ^cloudhsm
```

- Verwenden Sie auf einem Windows-System den folgenden Befehl:

```
wmic product get name,version
```

Themen

- [Voraussetzungen](#)
- [Schritt 1: Beenden des Client-Daemon](#)
- [Schritt 2: Upgrade des Client-SDK](#)
- [Schritt 3: Starten des Client-Daemon](#)

Voraussetzungen

Laden Sie die neueste Version des AWS CloudHSM Client-Daemons herunter und wählen Sie Ihre Komponenten aus.

Note

Sie müssen nicht alle Komponenten installieren. Für jede installierte Komponente müssen Sie diese Komponente aktualisieren, damit sie mit der Version des Client-Daemon übereinstimmt.

Neuester Linux-Client-Daemon

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
```

Neueste PKCS-#11-Bibliothek

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
```

Neuester 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
```

Neuester JCE-Anbieter

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
```

Schritt 1: Beenden des Client-Daemon

Verwenden Sie den folgenden Befehl, um den Client-Daemon zu beenden.

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
```

Schritt 2: Upgrade des Client-SDK

Der folgende Befehl zeigt die Syntax, die zum Aktualisieren des Client-Daemon und der Komponenten erforderlich ist. Entfernen Sie vor dem Ausführen des Befehls alle Komponenten, die Sie nicht aktualisieren möchten.

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-dyn-latest.el8.x86_64.rpm> \  
    <./cloudhsm-client-jce-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>
```

Schritt 3: Starten des Client-Daemon

Verwenden Sie den folgenden Befehl, um den Client-Daemon zu starten.

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 Unterstützte Plattformen vom Client SDK 3

 Die SDK-Versionen 5.8.0 und früher haben das Ende der Unterstützung erreicht. Nach dem 31. März 2025 wird die Dokumentation für die SDK-Versionen 3.4.4 und früher nicht mehr verfügbar sein.

AWS CloudHSM Client SDK 3 erfordert einen Client-Daemon und bietet Befehlszeilentools wie CloudHSM Management Utility (CMU), Key Management Utility (KMU) und das Konfigurationstool.

Die Basisunterstützung ist für jede Version des Client-SDK unterschiedlich. AWS CloudHSM In der Regel entspricht die Plattformunterstützung für Komponenten in einem SDK der Basisunterstützung, das ist jedoch nicht immer der Fall. Um die Plattformunterstützung für eine bestimmte Komponente zu ermitteln, stellen Sie zunächst sicher, dass die gewünschte Plattform im Basisbereich für das SDK angezeigt wird, und suchen Sie dann im Komponentenabschnitt nach Ausschlüssen oder anderen relevanten Informationen.

Die Plattformunterstützung ändert sich im Laufe der Zeit. Frühere Versionen des CloudHSM-Client SDK unterstützen möglicherweise nicht alle hier aufgeführten Betriebssysteme. Ermitteln Sie anhand der Versionshinweise die Betriebssystemunterstützung für frühere Versionen des CloudHSM-Client SDK. Weitere Informationen finden Sie unter [Downloads für das AWS CloudHSM Client-SDK](#).

AWS CloudHSM unterstützt nur 64-Bit-Betriebssysteme.

Themen

- [Linux-Unterstützung für AWS CloudHSM Client SDK 3](#)
- [Windows-Unterstützung für AWS CloudHSM Client SDK 3](#)
- [HSM-Kompatibilität für AWS CloudHSM Client SDK 3](#)

Linux-Unterstützung für AWS CloudHSM Client SDK 3

AWS CloudHSM Das Client SDK 3 unterstützt die folgenden Linux-Betriebssysteme und -Plattformen.

- 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] Keine Unterstützung für OpenSSL Dynamic Engine. Weitere Informationen finden Sie unter [OpenSSL Dynamic Engine](#).

[2] Keine Unterstützung für Client-SDK 3.3.0 und höher.

[3] SDK 3.4 ist die letzte unterstützte Version auf Ubuntu 16.04.

[4] SDK 3.4 ist die letzte unterstützte Version auf CentOS 8.3+.

Windows-Unterstützung für AWS CloudHSM Client SDK 3

AWS CloudHSM Das Client SDK 3 unterstützt die folgenden Versionen von Windows Server.

- Microsoft Windows Server 2012
- Microsoft Windows Server 2012 R2
- Microsoft Windows Server 2016
- Microsoft Windows Server 2019

HSM-Kompatibilität für AWS CloudHSM Client SDK 3

In der folgenden Tabelle wird die Kompatibilität AWS CloudHSM des Client SDK 3 für HSMs beschrieben.

hsm1.medium	hsm2m. mittel
Kompatibel mit Client-Version SDK 3.1.0 und höher.	Nicht unterstützt

PKCS #11 -Bibliothek für AWS CloudHSM Client SDK 3

PKCS #11 ist ein Standard für die Ausführung kryptografischer Operationen auf Hardware-Sicherheitsmodulen (HSM) in. AWS CloudHSM

Informationen zu Bootstrapping finden Sie unter [Verbinden mit dem Cluster](#).

Themen

- [Installieren Sie die PKCS #11 -Bibliothek für AWS CloudHSM Client SDK 3](#)
- [Authentifizieren Sie sich bei der PKCS #11 -Bibliothek für Client SDK 3 AWS CloudHSM](#)

- [Unterstützte Schlüsseltypen für die PKCS #11 -Bibliothek für AWS CloudHSM Client SDK 3](#)
- [Unterstützte Mechanismen für AWS CloudHSM Client SDK 3](#)
- [Unterstützte API-Operationen für AWS CloudHSM Client SDK 3](#)
- [Schlüsselattribute in der PKCS #11 -Bibliothek für AWS CloudHSM Client-SDK 3](#)
- [Codebeispiele für die PKCS #11 -Bibliothek für AWS CloudHSM Client SDK 3](#)

Installieren Sie die PKCS #11 -Bibliothek für AWS CloudHSM Client SDK 3

Dieses Thema enthält Anweisungen zur Installation der PKCS #11 -Bibliothek für die AWS CloudHSM Client SDK 3-Versionsserie. Weitere Informationen zum Client-SDK oder zur PKCS #11-Bibliothek finden Sie unter [Verwenden des Client-SDK](#) und der [PKCS #11-Bibliothek](#).

Voraussetzungen für das Client-SDK 3

Für die PKCS #11 -Bibliothek ist der AWS CloudHSM Client erforderlich.

Wenn Sie den AWS CloudHSM Client noch nicht installiert und konfiguriert haben, tun Sie das jetzt, indem Sie den Schritten unter [Installieren des Clients \(Linux\)](#) folgen. Nachdem Sie den Client installiert und konfiguriert haben, verwenden Sie den folgenden Befehl ein, um ihn zu starten.

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
```

Installieren Sie die PKCS #11-Bibliothek für das Client-SDK 3

Mit dem folgenden Befehl laden Sie die PKCS #11-Bibliothek herunter und installieren sie.

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
```

- Wenn auf der EC2-Instance, auf der Sie die PKCS #11-Bibliothek installiert haben, keine anderen Komponenten aus dem Client-SDK 3 installiert sind, müssen Sie das Client-SDK 3 booten. Sie müssen dies nur einmal auf jeder Instance mit einer Komponente aus dem Client-SDK 3 tun.
- Die PKCS #11-Bibliotheksdateien finden Sie an den folgenden Speicherorten:

Linux-Binärdateien, Konfigurationsskripten, Zertifikate und Protokolldateien:

```
/opt/cloudhsm/lib
```

Authentifizieren Sie sich bei der PKCS #11 -Bibliothek für Client SDK 3 AWS CloudHSM

Wenn Sie die PKCS #11 -Bibliothek verwenden, wird Ihre Anwendung unter einem bestimmten [Crypto User \(CU\)](#) in Ihrem Konto ausgeführt. HSMs AWS CloudHSM Ihre Anwendung kann nur die Schlüssel anzeigen und verwalten, die der CU besitzt und freigibt. Sie können eine vorhandene CU in Ihrer verwenden HSMs oder eine neue CU erstellen. Informationen zur [Verwaltung CUs finden Sie unter HSM-Benutzer mit CloudHSM CLI verwalten und HSM-Benutzer mit CloudHSM Management Utility \(CMU\) verwalten](#).

Um den CU für die PKCS #11-Bibliothek anzugeben, verwenden Sie den Pin-Parameter der PKCS #11-Funktion [C_Login](#). Denn der AWS CloudHSM Pin-Parameter hat das folgende Format:

```
<CU_user_name>:<password>
```

Mit dem folgenden Befehl wird beispielsweise dem PKCS #11-Bibliothek-Pin der CU mit dem Benutzernamen CryptoUser und dem Passwort CUPassword123! zugewiesen.

```
CryptoUser:CUPassword123!
```

Unterstützte Schlüsseltypen für die PKCS #11 -Bibliothek für AWS CloudHSM Client SDK 3

Die PKCS #11 -Bibliothek unterstützt die folgenden Schlüsseltypen mit dem AWS CloudHSM Client-SDK 3.

Schlüsseltyp	Description
RSA	Generieren Sie RSA-Schlüssel mit 2048 bis 4096 Bit, in Schritten von 256 Bit.
EC	Generieren Sie Schlüssel mit den Kurven secp224r1 (P-224), secp256r1 (P-256), secp256k1 (Blockchain), secp384r1 (P-384) und secp521r1 (P-521).
AES	Generieren der 128-, 192- und 256-Bit-AES-Schlüssel.
DES3 (Dreifaches DES)	Generieren Sie 192-Bit-Schlüssel DES3 . Eine bevorstehende Änderung finden Sie im Hinweis 1 unten.
GENERIC_SECRET	Generieren Sie generische Geheimnisse mit 1 bis 64 Byte.

- [1] Gemäß den NIST-Richtlinien ist dies für Cluster im FIPS-Modus nach 2023 nicht zulässig. Für Cluster im Nicht-FIPS-Modus ist dies auch nach 2023 zulässig. Details dazu finden Sie unter [FIPS-140-Konformität: Mechanismus 2024 nicht mehr unterstützt](#).

Unterstützte Mechanismen für AWS CloudHSM Client SDK 3

Die PKCS #11 -Bibliothek unterstützt die folgenden Algorithmen für das AWS CloudHSM Client-SDK 3:

- Verschlüsselung und Entschlüsselung — AES-CBC, AES-CTR, AES-ECB, AES-GCM, -CBC, -ECB, RSA-OAEP und RSA-PKCS DES3 DES3
- Signieren und Verifizieren – RSA, HMAC und ECDSA; mit und ohne Hashing
- SHA1 SHA256 SHA384Hash/Digest SHA224 —,,, und SHA512
- Key Wrap – AES Key Wrap⁴, AES-GCM, RSA-AES und RSA-OAEP
- Wichtige Ableitung — ECDH, -108 CTR KDF ⁵ SP800

Die Funktionstabelle des PKCS #11-Bibliotheksmechanismus

Die PKCS #11-Bibliothek entspricht Version 2.40 der PKCS #11-Spezifikation. Um eine kryptographische Funktion mit PKCS#11 aufzurufen, rufen Sie eine Funktion mit einem bestimmten Mechanismus auf. Die folgende Tabelle fasst die Kombinationen von Funktionen und Mechanismen zusammen, die von AWS CloudHSM unterstützt werden.

Tabelle zur Interpretation der unterstützten PKCS #11-Mechanismen und -Funktionen

Ein ✓ -Zeichen bedeutet, dass der Mechanismus für die AWS CloudHSM Funktion unterstützt wird. Wir unterstützen nicht alle möglichen Funktionen, die in der PKCS #11 Spezifikation aufgeführt sind. Ein ✕ -Zeichen AWS CloudHSM bedeutet, dass der Mechanismus für die angegebene Funktion noch nicht unterstützt wird, obwohl der PKCS #11 -Standard dies zulässt. Leere Zellen zeigen an, dass der PKCS #11-Standard den Mechanismus für die angegebene Funktion nicht unterstützt.

Unterstützte PKCS #11-Bibliotheksmechanismen und -funktionen

Mechanismus	Funktionen						
	Schlüssel generieren oder Schlüssel paar	Signieren und prüfen	SR und VR	Digest	Verschlüsseln/ Entschlüsseln	Schlüssel ableiten	Wickeln & UnWrap
CKM_RSA_PKCS_KEY_PAIR_GEN	✓						
CKM_RSA_X_9_31_KEY_PAIR_GEN	✓ ²						
CKM_RSA_X_509		✓			✓		

Mechanismus	Funktionen						
CKM_RSA_PKCS siehe Hinweis 8		✓ 1	✗		✓ 1		✓ 1
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					

Mechanismus	Funktionen						
CKM_SHA224_RSA_PKCS_PSS		✓ 3.2					
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_GENERATION	✓						
CKM_ECDSA		✓ 1					
CKM_ECDSA_SHA1		✓ 3.2					
CKM_ECDSA_SHA224		✓ 3.2					
CKM_ECDSA_SHA256		✓ 3.2					
CKM_ECDSA_SHA384		✓ 3.2					

Mechanismus	Funktionen						
CKM_ECDSA_SHA512		✓ 3.2					
CKM_ECDH1_DERIVE						✓ 5	
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 siehe Hinweis 8	✓						

Mechanismus	Funktionen						
CKM_DES3_CBC siehe Hinweis 8					✓ 3.3		✗
CKM_DES3_CBC_PAD siehe Hinweis 8					✓		✗
CKM_DES3_ECB siehe Hinweis 8					✓		✗
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					

Mechanismus	Funktionen						
CKM_SHA256			✓ 3.1				
CKM_SHA256_HMAC		✓ 3.3					
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

Mechanismus	Funktionen						
CKM_CLOUD HSM_AES_K EY_WRAP_P KCS5_PAD							✓ 7.1
CKM_CLOUD HSM_AES_K EY_WRAP_Z ERO_PAD							✓ 7.1

Anmerkungen zum Mechanismus

- [1] Nur Single-Part-Operationen.
- [2] Der Mechanismus ist funktionell identisch mit dem CKM_RSA_PKCS_KEY_PAIR_GEN-Mechanismus, bietet aber stärkere Garantien für die p- und q-Generierung.
- [3.1] AWS CloudHSM geht je nach Client-SDK unterschiedlich an das Hashing heran. Beim Client-SDK 3 hängt der Ort, an dem wir das Hashing durchführen, von der Datengröße ab und davon, ob Sie einteilige oder mehrteilige Operationen verwenden.

Einteilige Operationen im Client-SDK 3

In Tabelle 3.1 ist die maximale Datensatzgröße für jeden Mechanismus für das Client-SDK 3 aufgeführt. Der gesamte Hash wird innerhalb des HSM berechnet. Keine Unterstützung für Datengrößen über 16 KB.

Tabelle 3.1, Maximale Datensatzgröße für einteilige Operationen

Mechanismus	Maximale Datengröße
CKM_SHA_1	16296
CKM_SHA224	16264
CKM_SHA256	16296

Mechanismus	Maximale Datengröße
CKM_SHA384	16232
CKM_SHA512	16232

Mehrteilige Operationen Client-SDK 3

Support für Datengrößen über 16 KB, aber die Datengröße bestimmt, wo das Hashing stattfindet. Datenpuffer mit weniger als 16 KB werden innerhalb des HSM gehasht. Puffer zwischen 16 KB und der maximalen Datengröße für Ihr System werden lokal in der Software gehasht. Denken Sie daran: Für Hash-Funktionen sind keine kryptografischen Geheimnisse erforderlich, sodass Sie sie sicher außerhalb des HSM berechnen können.

- [3.2] AWS CloudHSM geht je nach Client-SDK unterschiedlich an das Hashing heran. Beim Client-SDK 3 hängt der Ort, an dem wir das Hashing durchführen, von der Datengröße ab und davon, ob Sie einteilige oder mehrteilige Operationen verwenden.

Einteilige Operationen Client-SDK 3

In Tabelle 3.2 ist die maximale Datensatzgröße für jeden Mechanismus für das Client-SDK 3 aufgeführt. Keine Unterstützung für Datengrößen über 16 KB.

Tabelle 3.2, Maximale Datensatzgröße für einteilige Operationen

Mechanismus	Maximale Datengröße
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

Mechanismus	Maximale Datengröße
CKM_SHA256_RSA_PKCS_PSS	16296
CKM_SHA384_RSA_PKCS_PSS	16232
CKM_SHA512_RSA_PKCS_PSS	16232
CKM_ECDSA_SHA1	16296
CKM_ECDSA_SHA224	16264
CKM_ECDSA_SHA256	16296
CKM_ECDSA_SHA384	16232
CKM_ECDSA_SHA512	16232

Mehrteilige Operationen Client-SDK 3

Support für Datengrößen über 16 KB, aber die Datengröße bestimmt, wo das Hashing stattfindet. Datenpuffer mit weniger als 16 KB werden innerhalb des HSM gehasht. Puffer zwischen 16 KB und der maximalen Datengröße für Ihr System werden lokal in der Software gehasht. Denken Sie daran: Für Hash-Funktionen sind keine kryptografischen Geheimnisse erforderlich, sodass Sie sie sicher außerhalb des HSM berechnen können.

- [3.3] Wenn der Datenpuffer die maximale Datengröße überschreitet, führt die Operation bei der Bearbeitung von Daten mittels eines der folgenden Mechanismen zu einem Fehler. Für diese Mechanismen muss die gesamte Datenverarbeitung innerhalb des HSM erfolgen. Die folgende Tabelle listet die maximale Datengröße für jeden Mechanismus auf:

Tabelle 3.3, Maximale Datensatzgröße

Mechanismus	Maximale Datengröße
CKM_SHA_1_HMAC	16288
CKM_SHA224_HMAC	16256
CKM_SHA256_HMAC	16288

Mechanismus	Maximale Datengröße
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

- [4] Bei AES-GCM-Verschlüsselungen akzeptiert HSM keine Initialisierungsvektor (IV)-Daten von der Anwendung. Sie müssen einen erzeugten IV verwenden. Das vom HSM bereitgestellte 12 Byte IV wird in den referenzierten Speicherbereich geschrieben, auf den das pIV-Element der von Ihnen bereitgestellten CK_GCM_PARAMS-Parameterstruktur zeigt. Um Missverständnissen vorzubeugen: Das PKCS#11-SDK in Version 1.1.1 und höher erzwingt, dass pIV auf einen auf Null zurückgesetzten Puffer verweist, wenn die AES-GCM-Verschlüsselung initialisiert wird.
- [5] Nur Client-SDK 3. Der Mechanismus ist zur Unterstützung von SSL/TLS-Auslagerungsfällen implementiert und wird nur teilweise innerhalb des HSM ausgeführt. Bevor Sie diesen Mechanismus verwenden, lesen Sie „Problem: ECDH-Schlüsselableitung wird nur teilweise innerhalb des HSM ausgeführt“ in [Bekannte Probleme mit der PKCS #11 -Bibliothek für AWS CloudHSM](#). CKM_ECDH1_DERIVE unterstützt nicht die secp521r1 (P-521) Kurve.
- [6] Die folgenden CK_MECHANISM_TYPE und CK_RSA_PKCS_MGF_TYPE werden als CK_RSA_PKCS_OAEP_PARAMS für CKM_RSA_PKCS_OAEP unterstützt:
 - CKM_SHA_1 mit CKG_MGF1_SHA1
 - CKM_SHA224 mit CKG_MGF1_SHA224
 - CKM_SHA256 mit CKG_MGF1_SHA256
 - CKM_SHA384 mit CKM_MGF1_SHA384
 - CKM_SHA512 mit CKM_MGF1_SHA512
- [7.1] Vom Anbieter definierter Mechanismus. Um die von anbieterdefinierten CloudHSM-Mechanismen zu verwenden, müssen PKCS #11 -Anwendungen während der Kompilierung `/opt/cloudhsm/include/pkcs11/cloudhsm_pkcs11_vendor_defs.h` enthalten.

CKM_CLOUDHSM_AES_GCM: Dieser proprietäre Mechanismus ist eine programmatisch sicherere Alternative zum Standard CKM_AES_GCM. Er stellt die vom HSM generierte IV dem Chiffretext voran, anstatt sie zurück in die CK_GCM_PARAMS- Struktur zu schreiben, die während der Chiffrierinitialisierung bereitgestellt wird. Sie können diesen Mechanismus mit C_Encrypt-, C_WrapKey-, C_Decrypt- und C_UnwrapKey-Funktionen verwenden. Bei Verwendung dieses Mechanismus muss die pIV-Variable in der CK_GCM_PARAMS-Struktur auf NULL gesetzt werden. Wenn Sie diesen Mechanismus mit C_Decrypt und C_UnwrapKey verwenden, wird erwartet, dass der IV dem Verschlüsselungstext vorangestellt wird, der entpackt werden soll.

CKM_CLOUDHSM_AES_KEY_WRAP_PKCS5_PAD: AES-Verschlüsselung mit PKCS #5 Padding

CKM_CLOUDHSM_AES_KEY_WRAP_ZERO_PAD: AES-Verschlüsselung mit Zero Padding

Weitere Informationen zum AES-Packen mit Schlüssel finden Sie unter [AES Key Wrapping](#).

- [8] Gemäß den NIST-Richtlinien ist dies für Cluster im FIPS-Modus nach 2023 nicht zulässig. Für Cluster im Nicht-FIPS-Modus ist dies auch nach 2023 zulässig. Details dazu finden Sie unter [FIPS-140-Konformität: Mechanismus 2024 nicht mehr unterstützt](#).

Unterstützte API-Operationen für AWS CloudHSM Client SDK 3

Die PKCS #11 -Bibliothek unterstützt die folgenden PKCS #11 -API-Operationen für das AWS CloudHSM Client-SDK 3.

- 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 (Nur Unterstützung für Client-SDK 3)
- C_SignRecoverInit (Nur Unterstützung für Client-SDK 3)

- C_SignUpdate
- C_UnWrapKey
- C_Verify
- C_VerifyFinal
- C_VerifyInit
- C_VerifyRecover (Nur Unterstützung für Client-SDK 3)
- C_VerifyRecoverInit (Nur Unterstützung für Client-SDK 3)
- C_VerifyUpdate
- C_WrapKey

Schlüsselattribute in der PKCS #11 -Bibliothek für AWS CloudHSM das Client-SDK 3

Bei einem Schlüsselobjekt kann es sich um einen öffentlichen, privaten oder geheimen Schlüssel handeln. Für ein Schlüsselobjekt genehmigte Aktionen werden durch Attribute angegeben. Attribute werden bei der Erstellung des Schlüsselobjekts bestimmt. Wenn Sie die PKCS #11 -Bibliothek für verwenden, weisen wir Standardwerte zu AWS CloudHSM, die im PKCS #11 -Standard spezifiziert sind.

AWS CloudHSM unterstützt nicht alle in der PKCS #11 -Spezifikation aufgeführten Attribute. Wir erfüllen die Spezifikation für alle von uns unterstützten Attribute. Diese Attribute sind in den entsprechenden Tabellen aufgeführt.

Kryptografische Funktionen wie C_CreateObject, C_GenerateKey, C_GenerateKeyPair, C_UnwrapKey und C_DeriveKey zum Erstellen, Ändern oder Kopieren von Objekten benötigen eine Attributvorlage für einen ihrer Parameter. Weitere Informationen zum Übertragen von Attributvorlagen während der Erstellung von Objekten finden Sie im Beispiel unter [Generieren von Schlüsseln über die PKCS #11-Bibliothek](#).

Die folgenden Themen enthalten weitere Informationen zu AWS CloudHSM Schlüsselattributen für Client SDK 3.

Themen

- [Tabelle mit PKCS #11 -Bibliotheksattributen für AWS CloudHSM Client SDK 3](#)
- [Ändern der PKCS #11 -Bibliotheksattribute für das AWS CloudHSM Client-SDK 3](#)
- [Interpretieren der Fehlercodes der PKCS #11 -Bibliothek für das AWS CloudHSM Client-SDK 3](#)

Tabelle mit PKCS #11 -Bibliotheksattributen für AWS CloudHSM Client SDK 3

Die PKCS #11 -Bibliothekstabelle für AWS CloudHSM Client SDK 3 enthält eine Liste von Attributen, die sich je nach Schlüsseltyp unterscheiden. Sie gibt an, ob ein bestimmtes Attribut für einen bestimmten Schlüsseltyp unterstützt wird, wenn eine bestimmte kryptografische Funktion mit verwendet wird. AWS CloudHSM

Legende:

- ✓ gibt an, dass CloudHSM das Attribut für den spezifischen Schlüsseltyp unterstützt.
- ✗ gibt an, dass CloudHSM das Attribut für den spezifischen Schlüsseltyp nicht unterstützt.
- R gibt an, dass der Attributwert für den spezifischen Schlüsseltyp als schreibgeschützt festgelegt ist.
- S zeigt an, dass das Attribut von `GetAttributeValue` nicht gelesen werden kann, da hierbei Groß- und Kleinschreibung beachtet werden muss.
- Eine leere Zelle in der Spalte „Standardwert“ gibt an, dass dem Attribut kein spezifischer Standardwert zugewiesen ist.

GenerateKeyPair

Attribut	Schlüsseltyp				Default Value (Standardwert)
	EC privat	EC öffentlich	RSA privat	RSA öffentlich	
CKA_CLASS	✓	✓	✓	✓	
CKA_KEY_TYPE	✓	✓	✓	✓	
CKA_LABEL	✓	✓	✓	✓	
CKA_ID	✓	✓	✓	✓	

Attribut	Schlüsseltyp				Default Value (Standard wert)
CKA_LOCAL		R	R	R	True
CKA_TOKEN		✓	✓	✓	False
CKA_PRIVATE		✓ ¹	✓ ¹	✓ ¹	True
CKA_ENCRYPT		✗	✓	✗	False
CKA_DECRYPT		✓	✗	✓	Falsch
CKA_DERIVE		✓	✓	✓	False
CKA_MODIFIABLE		✓ ¹	✓ ¹	✓ ¹	True
CKA_DESTROYABLE		✓	✓	✓	True
CKA_SIGN		✓	✗	✓	False
CKA_SIGN_RECOVER		✗	✗	✓ ³	
CKA_VERIFY		✗	✓	✗	Falsch

Attribut	Schlüsseltyp				Default Value (Standard wert)
CKA_VERIFY_RECOVER	×	×	×	✓ ⁴	
CKA_WRAP	×	✓	×	✓	Falsch
CKA_WRAP_TEMPLATE	×	✓	×	✓	
CKA_TRUSTED	×	✓	×	✓	Falsch
CKA_WRAP_WITH_TRUSTED	✓	×	✓	×	Falsch
CKA_UNWRAP	✓	×	✓	×	False
CKA_UNWRAP_TEMPLATE	✓	×	✓	×	
CKA_SENSITIVE	✓	×	✓	×	True
CKA_ALWAYS_SENSITIVE	R	×	R	×	
CKA_EXTRACTABLE	✓	×	✓	×	True

Attribut	Schlüsseltyp				Default Value (Standard wert)
CKA_NEVER_EXTRACTABLE	R	×	R	×	
CKA_MODULUS	×	×	×	×	
CKA_MODULUS_BITS	×	×	×	✓ ²	
CKA_PRIME_1	×	×	×	×	
CKA_PRIME_2	×	×	×	×	
CKA_COEFFICIENT	×	×	×	×	
CKA_EXPONENT_1	×	×	×	×	
CKA_EXPONENT_2	×	×	×	×	
CKA_PRIVATE_EXPONENT	×	×	×	×	
CKA_PUBLIC_EXPONENT	×	×	×	✓ ²	

Attribut	Schlüsseltyp				Default Value (Standard wert)
CKA_EC_PA RAMS	×	✓ ²	×	×	
CKA_EC_PO INT	×	×	×	×	
CKA_VALUE	×	×	×	×	
CKA_VALUE _LEN	×	×	×	×	
CKA_CHECK _VALUE	R	R	R	R	

GenerateKey

Attribut	Schlüsseltyp			Default Value (Standard wert)
	AES	DES3	Allgemein er geheimer Schlüssel	
CKA_CLASS	✓	✓	✓	
CKA_KEY_T YPE	✓	✓	✓	
CKA_LABEL	✓	✓	✓	
CKA_ID	✓	✓	✓	

Attribut	Schlüsseltyp				Default Value (Standard wert)
CKA_LOCAL		R	R	R	True
CKA_TOKEN		✓	✓	✓	False
CKA_PRIVATE		✓ ¹	✓ ¹	✓ ¹	True
CKA_ENCRYPT		✓	✓	✗	False
CKA_DECRYPT		✓	✓	✗	Falsch
CKA_DERIVE		✓	✓	✓	False
CKA_MODIFIABLE		✓ ¹	✓ ¹	✓ ¹	True
CKA_DESTROYABLE		✓	✓	✓	True
CKA_SIGN		✓	✓	✓	True
CKA_SIGN_RECOVER		✗	✗	✗	
CKA_VERIFY		✓	✓	✓	True
CKA_VERIFY_RECOVER		✗	✗	✗	
CKA_WRAP		✓	✓	✗	False

Attribut	Schlüsseltyp			Default Value (Standard wert)
CKA_WRAP_TEMPLATE	✓	✓	✗	
CKA_TRUSTED	✓	✓	✗	Falsch
CKA_WRAP_WITH_TRUSTED	✓	✓	✓	Falsch
CKA_UNWRAP	✓	✓	✗	False
CKA_UNWRAP_TEMPLATE	✓	✓	✗	
CKA_SENSITIVE	✓	✓	✓	True
CKA_ALWAYS_SENSITIVE	✗	✗	✗	
CKA_EXTRACTABLE	✓	✓	✓	True
CKA_NEVER_EXTRACTABLE	R	R	R	
CKA_MODULUS	✗	✗	✗	

Attribut	Schlüsseltyp			Default Value (Standard wert)
CKA_MODUL US_BITS	×	×	×	
CKA_PRIME _1	×	×	×	
CKA_PRIME _2	×	×	×	
CKA_COEFF ICIENT	×	×	×	
CKA_EXPON ENT_1	×	×	×	
CKA_EXPON ENT_2	×	×	×	
CKA_PRIVA TE_EXPONE NT	×	×	×	
CKA_PUBLI C_EXPONEN T	×	×	×	
CKA_EC_PA RAMS	×	×	×	
CKA_EC_PO INT	×	×	×	
CKA_VALUE	×	×	×	

Attribut	Schlüsseltyp			Default Value (Standard wert)
CKA_VALUE_LEN		✓ ²	✗	✓ ²
CKA_CHECK_VALUE		R	R	R

CreateObject

Attribut	Schlüsseltyp							Default Value (Standard wert)
	EC privat	EC öffentlic h	RSA privat	RSA öffentlic h	AES	DES3	Allgemein er geheimer Schlüssel	
CKA_CLASS	✓ ²	✓ ²	✓ ²	✓ ²	✓ ²	✓ ²	✓ ²	
CKA_KEY_T YPE	✓ ²	✓ ²	✓ ²	✓ ²	✓ ²	✓ ²	✓ ²	
CKA_LABEL	✓	✓	✓	✓	✓	✓	✓	
CKA_ID	✓	✓	✓	✓	✓	✓	✓	
CKA_LOCAL	R	R	R	R	R	R	R	Falsch
CKA_TOKEN	✓	✓	✓	✓	✓	✓	✓	False

Attribut	Schlüsseltyp							Default Value (Standard wert)
CKA_PRIVATE	✓ ¹	✓ ¹	✓ ¹	✓ ¹	✓ ¹	✓ ¹	✓ ¹	True
CKA_ENCRYPT	✗	✗	✗	✓	✓	✓	✗	False
CKA_DECRYPT	✗	✗	✓	✗	✓	✓	✗	Falsch
CKA_DERIVE	✓	✓	✓	✓	✓	✓	✓	False
CKA_MODIFIABLE	✓ ¹	✓ ¹	✓ ¹	✓ ¹	✓ ¹	✓ ¹	✓ ¹	True
CKA_DESTROYABLE	✓	✓	✓	✓	✓	✓	✓	True
CKA_SIGN	✓	✗	✓	✗	✓	✓	✓	False
CKA_SIGN_RECOVER	✗	✗	✓ ³	✗	✗	✗	✗	Falsch
CKA_VERIFY	✗	✓	✗	✓	✓	✓	✓	Falsch
CKA_VERIFY_RECOVER	✗	✗	✗	✓ ⁴	✗	✗	✗	
CKA_WRAP	✗	✗	✗	✓	✓	✓	✗	Falsch
CKA_WRAP_TEMPLATE	✗	✓	✗	✓	✓	✓	✗	

Attribut	Schlüsseltyp							Default Value (Standard wert)
CKA_TRUSTED	✗	✓	✗	✓	✓	✓	✗	Falsch
CKA_WRAP_WITH_TRUSTED	✓	✗	✓	✗	✓	✓	✓	Falsch
CKA_UNWRAP	✗	✗	✓	✗	✓	✓	✗	False
CKA_UNWRAP_TEMPLATE	✓	✗	✓	✗	✓	✓	✗	
CKA_SENSITIVE	✓	✗	✓	✗	✓	✓	✓	True
CKA_ALWAYS_SENSITIVE	R	✗	R	✗	R	R	R	
CKA_EXTRACTABLE	✓	✗	✓	✗	✓	✓	✓	True
CKA_NEVER_EXTRACTABLE	R	✗	R	✗	R	R	R	
CKA_MODULUS	✗	✗	✓ ²	✓ ²	✗	✗	✗	
CKA_MODULUS_BITS	✗	✗	✗	✗	✗	✗	✗	

Attribut	Schlüsseltyp							Default Value (Standard wert)
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_PARAMETERS	✓ ²	✓ ²	×	×	×	×	×	
CKA_EC_POINT	×	✓ ²	×	×	×	×	×	
CKA_VALUE	✓ ²	×	×	×	✓ ²	✓ ²	✓ ²	
CKA_VALUE_LEN	×	×	×	×	×	×	×	

Attribut	Schlüsseltyp							Default Value (Standard wert)
CKA_CHECK_VALUE	R	R	R	R	R	R	R	

UnwrapKey

Attribut	Schlüsseltyp						Default Value (Standard wert)
		EC privat	RSA privat	AES	DES3	Allgemeiner geheimer Schlüssel	
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	Falsch
CKA_TOKEN		✓	✓	✓	✓	✓	False

Attribut	Schlüsseltyp						Default Value (Standard wert)
CKA_PRIVATE	✓ ¹	✓ ¹	✓ ¹	✓ ¹	✓ ¹		True
CKA_ENCRYPT	✗	✗	✓	✓	✗		False
CKA_DECRYPT	✗	✓	✓	✓	✗		Falsch
CKA_DERIVE	✓	✓	✓	✓	✓		False
CKA_MODIFIABLE	✓ ¹	✓ ¹	✓ ¹	✓ ¹	✓ ¹		True
CKA_DESTROYABLE	✓	✓	✓	✓	✓		True
CKA_SIGN	✓	✓	✓	✓	✓		False
CKA_SIGN_RECOVER	✗	✓ ³	✗	✗	✗		Falsch
CKA_VERIFY	✗	✗	✓	✓	✓		Falsch
CKA_VERIFY_RECOVER	✗	✗	✗	✗	✗		
CKA_WRAP	✗	✗	✓	✓	✗		Falsch
CKA_UNWRAP	✗	✓	✓	✓	✗		False

Attribut	Schlüsseltyp						Default Value (Standard wert)
CKA_SENSITIVE		✓	✓	✓	✓	✓	True
CKA_EXTRACTABLE		✓	✓	✓	✓	✓	True
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		✗	✗	✗	✗	✗	

Attribut	Schlüsseltyp					Default Value (Standard wert)
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

Attribut	Schlüsseltyp			Default Value (Standard wert)
	AES	DES3	Allgemeiner geheimer Schlüssel	

Attribut	Schlüsseltyp			Default Value (Standard wert)
CKA_CLASS		✓ ²	✓ ²	✓ ²
CKA_KEY_T YPE		✓ ²	✓ ²	✓ ²
CKA_LABEL		✓	✓	✓
CKA_ID		✓	✓	✓
CKA_LOCAL		R	R	R
CKA_TOKEN		✓	✓	✓
CKA_PRIVATE		✓ ¹	✓ ¹	✓ ¹
CKA_ENCRYPT		✓	✓	✗
CKA_DECRYPT		✓	✓	✗
CKA_DERIVE		✓	✓	✓
CKA_MODIFIABLE		✓ ¹	✓ ¹	✓ ¹
CKA_DESTROYABLE		✓ ¹	✓ ¹	✓ ¹
CKA_SIGN		✓	✓	✓
CKA_SIGN_RECOVER		✗	✗	✗

Attribut	Schlüsseltyp				Default Value (Standard wert)
CKA_VERIFY		✓	✓	✓	Falsch
CKA_VERIFY_RECOVER		✗	✗	✗	
CKA_WRAP		✓	✓	✗	Falsch
CKA_UNWRAP		✓	✓	✗	False
CKA_SENSITIVE		✓	✓	✓	True
CKA_EXTRACTABLE		✓	✓	✓	True
CKA_NEVER_EXTRACTABLE		R	R	R	
CKA_ALWAYS_SENSITIVE		R	R	R	
CKA_MODULUS		✗	✗	✗	
CKA_MODULUS_BITS		✗	✗	✗	
CKA_PRIME_1		✗	✗	✗	

Attribut	Schlüsseltyp			Default Value (Standard wert)
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	×	×	×	
CKA_VALUE_LEN	✓ ²	×	✓ ²	
CKA_CHECK_VALUE	R	R	R	

GetAttributeValue

Attribut	Schlüsseltyp							
	EC privat	EC öffentlic h	RSA privat	RSA öffentlic h	AES	DES3	Allgemein er geheimer Schlüssel	
CKA_CLASS	✓	✓	✓	✓	✓	✓	✓	
CKA_KEY_T YPE	✓	✓	✓	✓	✓	✓	✓	
CKA_LABEL	✓	✓	✓	✓	✓	✓	✓	
CKA_ID	✓	✓	✓	✓	✓	✓	✓	
CKA_LOCAL	✓	✓	✓	✓	✓	✓	✓	
CKA_TOKEN	✓	✓	✓	✓	✓	✓	✓	
CKA_PRIVA TE	✓ ¹	✓ ¹	✓ ¹	✓ ¹	✓ ¹	✓ ¹	✓ ¹	
CKA_ENCRY PT	✗	✗	✗	✓	✓	✓	✗	
CKA_DECRY PT	✗	✗	✓	✗	✓	✓	✗	
CKA_DERIV E	✓	✓	✓	✓	✓	✓	✓	
CKA_MODIF IABLE	✓	✓	✓	✓	✓	✓	✓	

Attribut	Schlüsseltyp						
CKA_DESTR OYABLE	✓	✓	✓	✓	✓	✓	✓
CKA_SIGN	✓	✗	✓	✗	✓	✓	✓
CKA_SIGN_ RECOVER	✗	✗	✓	✗	✗	✗	✗
CKA_VERIF Y	✗	✓	✗	✓	✓	✓	✓
CKA_VERIF Y_RECOVER	✗	✗	✗	✓	✗	✗	✗
CKA_WRAP	✗	✗	✗	✓	✓	✓	✗
CKA_WRAP_ TEMPLATE	✗	✓	✗	✓	✓	✓	✗
CKA_TRUST ED	✗	✓	✗	✓	✓	✓	✓
CKA_WRAP_ WITH_TRUS TED	✓	✗	✓	✗	✓	✓	✓
CKA_UNWRA P	✗	✗	✓	✗	✓	✓	✗
CKA_UNWRA P_TEMPLAT E	✓	✗	✓	✗	✓	✓	✗
CKA_SENSI TIVE	✓	✗	✓	✗	✓	✓	✓

Attribut	Schlüsseltyp						
CKA_EXTRA_CTABLE	✓	✗	✓	✗	✓	✓	✓
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	✗	✗	✗	✗
CKA_COEFFICIENT	✗	✗	S	✗	✗	✗	✗
CKA_EXPONENT_1	✗	✗	S	✗	✗	✗	✗
CKA_EXPONENT_2	✗	✗	S	✗	✗	✗	✗
CKA_PRIVATE_EXPONENT	✗	✗	S	✗	✗	✗	✗

Attribut	Schlüsseltyp						
CKA_PUBLIC_EXPONENT	×	×	✓	✓	×	×	×
CKA_EC_PARAMS	✓	✓	×	×	×	×	×
CKA_EC_POINT	×	✓	×	×	×	×	×
CKA_VALUE	S	×	×	×	✓ ²	✓ ²	✓ ²
CKA_VALUE_LEN	×	×	×	×	✓	×	✓
CKA_CHECK_VALUE	✓	✓	✓	✓	✓	✓	×

Anmerkungen zu Attributen

- [1] Dieses Attribut wird teilweise von der Firmware unterstützt und muss explizit nur auf den Standardwert festgelegt werden.
- [2] Obligatorisches Attribut.
- [3] Nur Client-SDK 3. Das Attribut CKA_SIGN_RECOVER ist vom Attribut CKA_SIGN abgeleitet. Falls dieses festgelegt wird, kann es ausschließlich auf den Wert festgelegt werden, der auch für CKA_SIGN eingestellt ist. Falls es nicht festgelegt wird, leitet es den Standardwert von CKA_SIGN ab. Da CloudHSM nur RSA-basierte, wiederherstellbare Signaturmechanismen unterstützt, ist dieses Attribut derzeit nur auf öffentliche RSA-Schlüssel anwendbar.
- [4] Nur Client-SDK 3. Das Attribut CKA_VERIFY_RECOVER ist vom Attribut CKA_VERIFY abgeleitet. Falls dieses festgelegt wird, kann es ausschließlich auf den Wert festgelegt werden, der auch für CKA_VERIFY eingestellt ist. Falls es nicht festgelegt wird, leitet es den Standardwert von CKA_VERIFY ab. Da CloudHSM nur RSA-basierte, wiederherstellbare Signaturmechanismen unterstützt, ist dieses Attribut derzeit nur auf öffentliche RSA-Schlüssel anwendbar.

Ändern der PKCS #11 -Bibliotheksattribute für das AWS CloudHSM Client-SDK 3

Einige Attribute eines Objekts können geändert werden, nachdem das Objekt erstellt wurde, andere dagegen nicht. Um Attribute zu ändern, verwenden Sie den Befehl [setAttribute](#) aus `cloudhsm_mgmt_util`. Sie können auch eine Liste der Attribute und der Konstanten, die sie repräsentieren, ableiten, indem Sie den Befehl [listAttribute](#) aus `cloudhsm_mgmt_util` verwenden.

Die folgende Liste zeigt Attribute, die nach der Erstellung eines Objekts geändert werden dürfen.

- CKA_LABEL
- CKA_TOKEN

Note

Änderungen sind nur zum Umwandeln eines Sitzungsschlüssels in einen Token-Schlüssel zulässig. Verwenden Sie den Befehl [setAttribute](#) aus `key_mgmt_util`, um den Wert des Attributs zu ändern.

- CKA_ENCRYPT
- CKA_DECRYPT
- CKA_SIGN
- CKA_VERIFY
- CKA_WRAP
- CKA_UNWRAP
- CKA_LABEL
- CKA_SENSITIVE
- CKA_DERIVE

Note

Dieses Attribut unterstützt die Schlüsselableitung. Es muss für alle öffentlichen Schlüssel `False` lauten und kann nicht auf `True` festgelegt werden. Für geheime und private EC-Schlüssel kann es auf `True` oder `False` eingestellt werden.

- CKA_TRUSTED

 Note

Dieses Attribut kann nur vom Verschlüsselungsverantwortlichen (CO) auf `True` oder `False` festgelegt werden.

- `CKA_WRAP_WITH_TRUSTED`

 Note

Wenden Sie dieses Attribut auf einen exportierbaren Datenschlüssel an, um anzugeben, dass Sie diesen Schlüssel nur mit Schlüsseln packen können, die als `CKA_TRUSTED` markiert sind. Sobald Sie `CKA_WRAP_WITH_TRUSTED` auf „true“ setzen, wird das Attribut schreibgeschützt und Sie können das Attribut nicht mehr ändern oder entfernen.

Interpretieren der Fehlercodes der PKCS #11 -Bibliothek für das AWS CloudHSM Client-SDK 3

Die Angabe eines PKCS #11 -Bibliotheksattributs in der Vorlage, das von einem bestimmten Schlüssel nicht unterstützt wird, führt zu einem Fehler. Die folgende Tabelle enthält die Fehlercodes, die generiert werden, wenn Sie gegen die Spezifikationen verstoßen:

Fehlercode	Beschreibung
<code>CKR_TEMPLATE_INCONSISTENT</code>	Diese Fehlermeldung erhalten Sie, wenn Sie in der Attributvorlage ein Attribut angeben, das zwar der PKCS #11-Spezifikation entspricht, jedoch nicht von CloudHSM unterstützt wird.
<code>CKR_ATTRIBUTE_TYPE_INVALID</code>	Diese Fehlermeldung erhalten Sie, wenn Sie den Wert für ein Attribut abrufen, das zwar der PKCS #11-Spezifikation entspricht, jedoch nicht von CloudHSM unterstützt wird.
<code>CKR_ATTRIBUTE_INCOMPLETE</code>	Diese Fehlermeldung erhalten Sie, wenn Sie in der Attributvorlage das obligatorische Attribut nicht angeben.

Fehlercode	Beschreibung
CKR_ATTRIBUTE_READ_ONLY	Diese Fehlermeldung erhalten Sie, wenn Sie in der Attributvorlage ein schreibgeschütztes Attribut angeben.

Codebeispiele für die PKCS #11 -Bibliothek für AWS CloudHSM Client SDK 3

Die Codebeispiele auf dieser GitHub Seite zeigen Ihnen, wie Sie grundlegende Aufgaben mit der PKCS #11 -Bibliothek für erledigen. AWS CloudHSM

Voraussetzungen für den Beispiel-Code

Bevor Sie die Beispiele ausführen, führen Sie die folgenden Schritte aus, um Ihre Umgebung einzurichten:

- Installieren und konfigurieren Sie die [PKCS #11-Bibliothek](#) für das Client-SDK 3.
- Richten Sie einen [Crypto-Benutzer \(CU\)](#) ein. Ihre Anwendung verwendet dieses HSM-Konto, um die Codebeispiele auf dem HSM auszuführen.

Codebeispiele

Codebeispiele für die AWS CloudHSM Softwarebibliothek für PKCS #11 finden Sie unter. [GitHub](#)
Dieses Repository enthält Beispiele für allgemeine Vorgänge mit PKCS #11, einschließlich Verschlüsselung, Entschlüsselung, Signieren und Verifizieren.

- [Schlüssel generieren \(AES, RSA, EC\)](#)
- [Schlüsselattribute auflisten](#)
- [Verschlüsseln und Entschlüsseln von Daten mit AES GCM](#)
- [Verschlüsseln und Entschlüsseln von Daten mit AES_CTR](#)
- [Verschlüsseln und Entschlüsseln von Daten mit 3DES](#)
- [Signieren und Verifizieren von Daten mit RSA](#)
- [Ableiten von Schlüsseln mit HMAC KDF](#)
- [Verpacken und Entpacken von Schlüsseln mit AES und PKCS #5 Padding](#)
- [Verpacken und Entpacken von Schlüsseln mit AES ohne Padding](#)
- [Verpacken und Entpacken von Schlüsseln mit AES und Zero Padding](#)

- [Packen und Entpacken von Schlüsseln mit AES-GCM](#)
- [Schlüssel mit RSA packen und entpacken](#)

OpenSSL Dynamic Engine für AWS CloudHSM Client SDK 3

Mit der AWS CloudHSM OpenSSL Dynamic Engine können Sie kryptografische Operationen über die OpenSSL-API auf Ihren CloudHSM-Cluster auslagern.

AWS CloudHSM Das Client-SDK 3 benötigt einen Client-Daemon, um eine Verbindung zum Cluster herzustellen. Es unterstützt:

- RSA-Schlüsselgenerierung für 2048-, 3072- und 4096-Bit-Schlüssel.
- RSA Signieren/Überprüfen.
- RSA Verschlüsseln/Entschlüsseln.
- Generierung einer Zufallszahl, die kryptografisch sicher und FIPS-validiert ist.

Verwenden Sie die folgenden Abschnitte, um die AWS CloudHSM dynamische Engine für OpenSSL zu installieren und zu konfigurieren.

Themen

- [Voraussetzungen für OpenSSL Dynamic Engine mit AWS CloudHSM Client SDK 3](#)
- [Installieren Sie die OpenSSL Dynamic Engine für AWS CloudHSM Client SDK 3](#)
- [Verwenden Sie die OpenSSL Dynamic Engine für AWS CloudHSM Client SDK 3](#)

Voraussetzungen für OpenSSL Dynamic Engine mit AWS CloudHSM Client SDK 3

Informationen zur Plattformunterstützung finden Sie unter [AWS CloudHSM Unterstützte Plattformen vom Client SDK 3](#).

Bevor Sie die AWS CloudHSM dynamische Engine für OpenSSL mit Client SDK 3 verwenden können, benötigen Sie den AWS CloudHSM Client.

Der Client ist ein Daemon, der eine end-to-end verschlüsselte Kommunikation mit dem HSMs in Ihrem Cluster herstellt, und die OpenSSL-Engine kommuniziert lokal mit dem Client. Informationen zur Installation und Konfiguration des AWS CloudHSM Clients finden Sie unter. [Installieren des Clients \(Linux\)](#) Verwenden Sie dann den folgenden Befehl, um ihn zu starten.

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
```

Installieren Sie die OpenSSL Dynamic Engine für AWS CloudHSM Client SDK 3

In den folgenden Schritten wird beschrieben, wie Sie die AWS CloudHSM dynamische Engine für OpenSSL mit dem Client SDK 3 installieren und konfigurieren. Weitere Informationen zum Upgrade finden Sie unter [Aktualisieren Sie das Client-SDK 3](#).

Zum Installieren und Konfigurieren der OpenSSL-Engine

1. Verwenden Sie die folgenden Befehle zum Herunterladen und Installieren der OpenSSL-Engine.

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
```

Die OpenSSL-Engine ist unter `/opt/cloudhsm/lib/libcloudhsm_openssl.so` installiert.

2. Verwenden Sie den folgenden Befehl ein, um eine Umgebungsvariable `n3fips_password` mit den Anmeldeinformationen eines Crypto-Benutzers (CU) anzulegen.

```
$ export n3fips_password=<HSM user name>:<password>
```

Verwenden Sie die OpenSSL Dynamic Engine für AWS CloudHSM Client SDK 3

Um die AWS CloudHSM dynamische Engine für OpenSSL von einer OpenSSL-integrierten Anwendung aus zu verwenden, stellen Sie sicher, dass Ihre Anwendung die angegebene dynamische OpenSSL-Engine verwendet. `cloudhsm` Die freigegebene Bibliothek für die Dynamic Engine finden Sie unter `/opt/cloudhsm/lib/libcloudhsm_openssl.so`.

Um die AWS CloudHSM dynamische Engine für OpenSSL von der OpenSSL-Befehlszeile aus zu verwenden, verwenden Sie die `-engine` Option, um die OpenSSL-Dynamik-Engine mit dem Namen anzugeben. `cloudhsm` Beispiel:

```
$ openssl s_server -cert <server.crt> -key <server.key> -engine cloudhsm
```

JCE-Anbieter für AWS CloudHSM Client SDK 3

Der AWS CloudHSM JCE-Anbieter ist eine Provider-Implementierung, die auf dem Java Cryptographic Extension (JCE) Provider-Framework basiert. JCE ermöglicht es Ihnen, kryptografische Operationen mit dem Java Development Kits (JDK) durchzuführen. In diesem Handbuch wird der AWS CloudHSM JCE-Anbieter manchmal als JCE-Anbieter bezeichnet. Verwenden Sie den JCE-Anbieter und das JDK, um kryptografische Operationen auf das HSM auszulagern.

Themen

- [Installieren Sie den JCE-Anbieter für AWS CloudHSM Client SDK 3](#)
- [Grundlagen der Schlüsselverwaltung im JCE-Anbieter für Client SDK 3 AWS CloudHSM](#)
- [Unterstützte Mechanismen für Client SDK 3 für AWS CloudHSM Client SDK 3](#)
- [Unterstützte Java-Schlüsselattribute für AWS CloudHSM Client SDK 3](#)
- [Codebeispiele für die AWS CloudHSM Softwarebibliothek für Java für Client SDK 3](#)
- [AWS CloudHSM KeyStore Java-Klasse für Client SDK 3](#)

Installieren Sie den JCE-Anbieter für AWS CloudHSM Client SDK 3

Bevor Sie den JCE-Anbieter verwenden können, benötigen Sie den AWS CloudHSM Client.

Der Client ist ein Daemon, der eine end-to-end verschlüsselte Kommunikation mit dem HSMs in Ihrem Cluster herstellt. Der JCE-Anbieter kommuniziert lokal mit dem Client. Wenn Sie das AWS CloudHSM Client-Paket noch nicht installiert und konfiguriert haben, tun Sie dies jetzt, indem Sie die Schritte unter [Installieren des Clients \(Linux\)](#) befolgen. Nachdem Sie den Client installiert und konfiguriert haben, verwenden Sie den folgenden Befehl ein, um ihn zu starten.

Beachten Sie, dass der JCE-Anbieter nur unter Linux und kompatiblen Betriebssystemen unterstützt wird.

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
```

Verwenden Sie die folgenden Abschnitte, um den Anbieter zu installieren, zu überprüfen und Anmeldeinformationen für ihn bereitzustellen.

Themen

- [Schritt 1: Installieren Sie den JCE-Anbieter](#)
- [Schritt 2: Überprüfen Sie die Installation](#)
- [Schritt 3: Geben Sie dem JCE-Anbieter Anmeldeinformationen an](#)

Schritt 1: Installieren Sie den JCE-Anbieter

Verwenden Sie die folgenden Befehle, um den JCE-Anbieter herunterzuladen und zu installieren. Dieser Anbieter wird nur von Linux und kompatiblen Betriebssystemen unterstützt.

Note

Informationen zum Upgraden finden Sie unter [Aktualisieren Sie das Client-SDK 3](#).

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
```

Nachdem Sie die vorangegangenen Befehle ausgeführt haben, finden Sie die folgenden JCE-Anbieter-Dateien:

- /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

Schritt 2: Überprüfen Sie die Installation

Führen Sie grundlegende Funktionen des HSM aus, um die Installation zu überprüfen.

Um die Installation des JCE-Anbieters zu überprüfen

1. (Optional) Falls Sie Java nicht bereits in Ihrer Umgebung installiert haben, verwenden Sie den folgenden Befehl zur Installation.

Linux (and compatible libraries)

```
$ sudo yum install java-1.8.0-openjdk
```

Ubuntu

```
$ sudo apt-get install openjdk-8-jre
```

2. Verwenden Sie die folgenden Befehle, um die erforderlichen Umgebungsvariablen festzulegen. Ersetzen Sie *<HSM user name>* und *<password>* durch die Anmeldeinformationen eines Krypto-Benutzers (CU).

```
$ export LD_LIBRARY_PATH=/opt/cloudhsm/lib
```

```
$ export HSM_PARTITION=PARTITION_1
```

```
$ export HSM_USER=<HSM user name>
```

```
$ export HSM_PASSWORD=<password>
```

3. Führen Sie den folgenden Befehl aus, um den Test der grundlegenden Funktionen zu starten. Wenn dies erfolgreich ist, sollte die Ausgabe des Befehls ähnlich wie die folgende sein.

```
$ 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)
```

Schritt 3: Geben Sie dem JCE-Anbieter Anmeldeinformationen an

HSMs Sie müssen Ihre Java-Anwendung authentifizieren, bevor die Anwendung sie verwenden kann. Jede Anwendung kann eine Sitzung verwenden. HSMs authentifizieren Sie eine Sitzung, indem Sie entweder die explizite oder die implizite Anmeldemethode verwenden.

Explizite Anmeldemethode – Mit dieser Methode können Sie CloudHSM-Anmeldeinformationen direkt in der Anwendung bereitstellen. Es verwendet die `LoginManager.login()`-Methode, bei der Sie den CU-Benutzernamen, das Passwort und die HSM-Partitions-ID übergeben. Weitere Informationen zur Verwendung der expliziten Anmeldemethode finden Sie im Code-Beispiel [Anmelden bei einem HSM](#).

Implizite Anmeldemethode – Mit dieser Methode können Sie CloudHSM-Anmeldeinformationen entweder in einer neuen Property-Datei, über Systemeigenschaften oder als Umgebungsvariablen festlegen.

- Neue Eigenschaftsdatei – Erstellen Sie eine neue Datei namens `HsmCredentials.properties` und fügen Sie sie zum CLASSPATH Ihrer Anwendung hinzu. Die Datei sollte Folgendes enthalten:

```
HSM_PARTITION = PARTITION_1  
HSM_USER = <HSM user name>  
HSM_PASSWORD = <password>
```

- Systemeigenschaften – Anmeldeinformationen über Systemeigenschaften fest, wenn Sie Ihre Anwendung ausführen. Die folgenden Beispiele zeigen zwei verschiedene Möglichkeiten:

```
$ 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>");
```

- Umgebungsvariablen – Anmeldeinformationen als Umgebungsvariablen festlegen.

```
$ export HSM_PARTITION=PARTITION_1  
$ export HSM_USER=<HSM user name>  
$ export HSM_PASSWORD=<password>
```

Anmeldeinformationen sind möglicherweise nicht verfügbar, wenn die Anwendung sie nicht bereitstellt oder wenn Sie eine Operation ausführen, bevor der HSM die Sitzung authentifiziert. In diesen Fällen sucht die CloudHSM-Softwarebibliothek für Java in der folgenden Reihenfolge nach den Anmeldeinformationen:

1. `HsmCredentials.properties`
2. Systemeigenschaften
3. Umgebungsvariablen

Fehlerbehandlung

Die Fehlerbehandlung ist bei der expliziten Anmeldung einfacher als bei der impliziten Anmeldemethode. Wenn Sie die Klasse `LoginManager` verwenden, haben Sie mehr Kontrolle darüber, wie Ihre Anwendung mit Fehlern umgeht. Die implizite Anmeldemethode macht es

schwierig, die Fehlerbehandlung zu verstehen, wenn die Anmeldeinformationen ungültig sind oder Probleme bei der HSMs Authentifizierung der Sitzung auftreten.

Grundlagen der Schlüsselverwaltung im JCE-Anbieter für Client SDK 3 AWS CloudHSM

Die Grundlagen der Schlüsselverwaltung im JCE-Anbieter umfassen den Import von Schlüsseln, den Export von Schlüsseln, das Laden von Schlüsseln per Handle oder das Löschen von Schlüsseln. Weitere Informationen zur Schlüsselverwaltung finden Sie im Codebeispiel [Schlüssel verwalten](#).

Weitere Codebeispiele für JCE-Anbieter finden Sie auch in [Codebeispiele](#).

Unterstützte Mechanismen für Client SDK 3 für AWS CloudHSM Client SDK 3

Dieses Thema enthält Informationen zu den unterstützten Mechanismen für den JCE-Anbieter mit dem AWS CloudHSM Client-SDK 3. Informationen zu den Schnittstellen und Engine-Klassen der Java Cryptography Architecture (JCA), die von unterstützt werden AWS CloudHSM, finden Sie in den folgenden Themen.

Themen

- [Unterstützte Schlüssel](#)
- [Unterstützte Verschlüsselungen](#)
- [Unterstützte Digests](#)
- [Unterstützte Hash-basierter Nachrichtenauthentifizierungscode \(HMAC\)-Algorithmen](#)
- [Unterstützte Mechanismen sign/verify](#)
- [Anmerkungen zum Mechanismus](#)

Unterstützte Schlüssel

Mit der AWS CloudHSM Softwarebibliothek für Java können Sie die folgenden Schlüsseltypen generieren.

- AES – 128-, 192- und 256-Bit AES-Schlüssel.
- DESede — 92-Bit-3DES-Schlüssel. Eine bevorstehende Änderung finden Sie im Hinweis [1](#) unten.
- ECC-Schlüsselpaare für die NIST secp256r1- (P-256), secp384r1- (P-384) und secp256k1-Kurven (Blockchain).
- RSA – 2048-Bit- bis 4096-Bit-RSA-Schlüssel, in Schritten von je 256 Bit.

Zusätzlich zu den Standardparametern unterstützen wir die folgenden Parameter für die einzelnen generierten Schlüssel.

- **Label:** Eine Schlüsselbezeichnung anhand der Sie nach Schlüsseln suchen können.
- **isExtractable:** Gibt an, ob ein Schlüssel vom HSM exportiert werden kann.
- **isPersistent:** Gibt an, ob der Schlüssel nach Beenden der aktuellen Sitzung weiterhin im HSM bleibt.

Note

Die Java-Bibliothek Version 3.1 bietet die Möglichkeit, Parameter genauer zu spezifizieren. Weitere Informationen finden Sie unter [Unterstützte Java-Attribute](#).

Unterstützte Verschlüsselungen

Die AWS CloudHSM Softwarebibliothek für Java unterstützt die folgenden Kombinationen aus Algorithmus, Modus und Polsterung.

Algorithmus	Mode	Padding	Hinweise
AES	CBC	AES/CBC/N oPadding	Implementiert Cipher.EN CRYPT_MODE und Cipher.DE CRYPT_MODE .
		AES/CBC/P KCS5Padding	
AES	ECB	AES/ECB/N oPadding	Implementiert Cipher.EN CRYPT_MODE und Cipher.DE CRYPT_MODE . Transformation AES verwenden.
		AES/ECB/P KCS5Padding	
AES	CTR	AES/CTR/N oPadding	Implementiert Cipher.EN

Algorithmus	Mode	Padding	Hinweise
			CRYPT_MODE und Cipher.DE CRYPT_MODE .
AES	GCM	AES/GCM/N oPadding	Implementiert Cipher.EN CRYPT_MODE und Cipher.DE CRYPT_MOD E , Cipher.WR AP_MODE und Cipher.UN WRAP_MODE . HSM ignoriert den Initialisierungsvektor (IV) der Anforderung während der AES- GCM-Datenversc hlüsselung und verwendet stattdess en einen selbst generierten IV. Nach Abschluss der Operation müssen Sie Cipher.getIV() abrufen, um den IV zu erhalten.

Algorithmus	Mode	Padding	Hinweise
AESWrap	ECB	AESWrap/ECB/ ZeroPadding AESWrap/ECB/ NoPadding AESWrap/ECB/ PKCS5Padding	Implementiert Cipher.WR AP_MODE und Cipher.UN WRAP_MODE . Transformation AES verwenden.
DESede (Dreifaches DES)	CBC	DESede/CBC/ NoPadding DESede/CBC/ PKCS5Padding	Implementiert Cipher.EN CRYPT_MODE und Cipher.DE CRYPT_MODE . Die Schlüsselerstellun gsvorgänge akzeptier en die Größen 168 oder 192 Bits. Intern sind jedoch alle DESede Schlüssel 192 Bit groß. Eine bevorstehende Änderung finden Sie im Hinweis 1 unten.

Algorithmus	Mode	Padding	Hinweise
DESede (Dreifaches DES)	ECB	DESede/ECB/NoPadding DESede/ECB/PKCS5Padding	Implementiert <code>Cipher.ENCRYPT_MODE</code> und <code>Cipher.DECRYPT_MODE</code>. Die Schlüsselerstellungsvorgänge akzeptieren die Größen 168 oder 192 Bits. Intern sind jedoch alle DESede Schlüssel 192 Bit groß. Eine bevorstehende Änderung finden Sie im Hinweis 1 unten.
RSA	ECB	RSA/ECB/NoPadding RSA/ECB/PKCS1Padding	Implementiert <code>Cipher.ENCRYPT_MODE</code> und <code>Cipher.DECRYPT_MODE</code>. Eine bevorstehende Änderung finden Sie im Hinweis 1 unten.

Algorithmus	Mode	Padding	Hinweise
RSA	ECB	RSA/ECB/0 AEPPadding	Implementiert Cipher.EN CRYPT_MOD E , Cipher.DE CRYPT_MOD E , Cipher.WR AP_MODE und Cipher.UN WRAP_MODE .
		RSA/ECB/0 AEPWithSH A-1ANDMGF 1Padding	
		RSA/ECB/0 AEPWithSH A-224ANDM GF1Padding	0AEPPadding ist 0AEP mit den SHA-1 Padding-Typ.
		RSA/ECB/0 AEPWithSH A-256ANDM GF1Padding	
		RSA/ECB/0 AEPWithSH A-384ANDM GF1Padding	
		RSA/ECB/0 AEPWithSH A-512ANDM GF1Padding	
RSAAESWrap	ECB	0AEPPADDING	Implementiert Cipher.WR AP_Mode und Cipher.UN WRAP_MODE .

Unterstützte Digests

Die AWS CloudHSM Softwarebibliothek für Java unterstützt die folgenden Message Digests.

- SHA-1
- SHA-224
- SHA-256
- SHA-384
- SHA-512

Note

Daten mit weniger als 16 KB werden im HSM mit einem Hash-Wert versehen. Größere Daten werden lokal in der Software mit einem Hash-Wert versehen.

Unterstützte Hash-basierter Nachrichtenauthentifizierungscode (HMAC)-Algorithmen

Die AWS CloudHSM Softwarebibliothek für Java unterstützt die folgenden HMAC-Algorithmen.

- HmacSHA1
- HmacSHA224
- HmacSHA256
- HmacSHA384
- HmacSHA512

Unterstützte Mechanismen sign/verify

Die AWS CloudHSM Softwarebibliothek für Java unterstützt die folgenden Arten der Signatur und Überprüfung.

RSA-Signatortypen

- NONEwithRSA
- SHA1withRSA

- SHA224withRSA
- SHA256withRSA
- SHA384withRSA
- SHA512withRSA
- SHA1withRSA/PSS
- SHA224withRSA/PSS
- SHA256withRSA/PSS
- SHA384withRSA/PSS
- SHA512withRSA/PSS

ECDSA-Signatortypen

- NONEwithECDSA
- SHA1withECDSA
- SHA224withECDSA
- SHA256withECDSA
- SHA384withECDSA
- SHA512withECDSA

Anmerkungen zum Mechanismus

[1] Gemäß den NIST-Richtlinien ist dies für Cluster im FIPS-Modus nach 2023 nicht zulässig. Für Cluster im Nicht-FIPS-Modus ist dies auch nach 2023 zulässig. Details dazu finden Sie unter [FIPS-140-Konformität: Mechanismus 2024 nicht mehr unterstützt](#).

Unterstützte Java-Schlüsselattribute für AWS CloudHSM Client SDK 3

In diesem Thema wird beschrieben, wie Sie eine proprietäre Erweiterung für die Java-Bibliothek, Version 3.1, verwenden, um Schlüsselattribute für das AWS CloudHSM Client SDK 3 festzulegen. Verwenden Sie diese Erweiterung, um unterstützte Schlüsselattribute und ihre Werte während der folgenden Vorgänge festzulegen:

- Schlüsselgenerierung

- Schlüsselimport
- Schlüssel entpacken

Note

Die Erweiterung zum Festlegen benutzerdefinierter Schlüsselattribute ist eine optionale Funktion. Wenn Sie bereits über einen in der Java-Bibliothek Version 3.0 funktionierenden Code verfügen, müssen Sie diesen Code nicht ändern. Schlüssel, die Sie erstellen, enthalten weiterhin dieselben Attribute wie zuvor.

Themen

- [Grundlegendes zu Attributen](#)
- [Unterstützte Attribute](#)
- [Festlegen von Attributen für einen Schlüssel](#)
- [Zusammenführung](#)

Grundlegendes zu Attributen

Mithilfe von Schlüsselattributen legen Sie fest, welche Aktionen für Schlüsselobjekte zulässig sind, einschließlich öffentlicher, privater oder geheimer Schlüssel. Schlüsselattribute und -werte definieren Sie bei der Erstellung von Schlüsselobjekten.

Die Java Cryptography Extension (JCE) gibt jedoch nicht an, wie Sie Werte für Schlüsselattribute festlegen sollten. Daher sind die meisten Aktionen standardmäßig zulässig. Im Gegensatz dazu definiert der PKCS# 11-Standard einen umfassenden Satz von Attributen mit restriktiveren Standardeinstellungen. Ab der Java-Bibliothek Version 3.1 bietet CloudHSM eine proprietäre Erweiterung, mit der Sie restriktivere Werte für häufig verwendete Attribute festlegen können.

Unterstützte Attribute

Sie können Werte für die Attribute festlegen, die in der folgenden Tabelle aufgeführt sind. Als bewährte Methode legen Sie nur Werte für Attribute fest, die Sie einschränken möchten. Wenn Sie keinen Wert angeben, verwendet CloudHSM den in der folgenden Tabelle angegebenen Standardwert. Eine leere Zelle in der Spalte „Standardwert“ gibt an, dass dem Attribut kein spezifischer Standardwert zugewiesen ist.

Attribut	Standardwert			Hinweise
	Symmetrischer Schlüssel	Öffentlicher Schlüssel im Schlüsselpaar	Privater Schlüssel im Schlüsselpaar	
CKA_TOKEN	FALSE	FALSE	FALSE	Ein permanenter Schlüssel, der HSMs im gesamten Cluster repliziert und in Backups enthalten ist. CKA_TOKEN = FALSE impliziert einen Sitzungsschlüssel, der nur auf ein HSM geladen und automatisch gelöscht wird, wenn die Verbindung zum HSM unterbrochen wird.
CKA_LABEL				Eine benutzerdefinierte Zeichenfolge. Sie ermöglicht Ihnen, Schlüssel auf Ihrem HSM leicht zu identifizieren.
CKA_EXTRACTABLE	TRUE		TRUE	„True“ gibt an, dass Sie

Attribut	Standardwert			Hinweise
				diesen Schlüssel aus dem HSM exportieren können.
CKA_ENCRYPT	TRUE	TRUE		„True“ gibt an, dass Sie den Schlüssel zur Verschlüsselung eines beliebigen Puffers verwenden können.
CKA_DECRYPT	TRUE		TRUE	„True“ gibt an, dass Sie den Schlüssel zur Entschlüsselung eines beliebigen Puffers verwenden können. Sie legen dies im Allgemeinen auf „FALSE“ fest für einen Schlüssel, dessen CKA_WRAP auf „True“ festgelegt ist.

Attribut	Standardwert			Hinweise
CKA_WRAP	TRUE	TRUE		„True“ gibt an, dass Sie den Schlüssel zum Packen eines anderen Schlüssels verwenden können. Für private Schlüssel legen Sie dies in der Regel auf „FALSE“ fest.
CKA_UNWRAP	TRUE		TRUE	„True“ gibt an, dass Sie mit dem Schlüssel einen anderen Schlüssel entpacken (importieren) können.

Attribut	Standardwert			Hinweise
CKA_SIGN	TRUE		TRUE	„True“ gibt an, dass Sie den Schlüssel verwenden können, um einen Hashwert zu signieren. Für öffentliche und private Schlüssel, die Sie archiviert haben, ist dies im Allgemeinen auf „FALSE“ festgelegt.
CKA_VERIFY	TRUE	TRUE		„True“ gibt an, dass Sie den Schlüssel verwenden können, um eine Signatur zu überprüfen. Für private Schlüssel ist dies im Allgemeinen auf „FALSE“ festgelegt.

Attribut	Standardwert			Hinweise
CKA_PRIVATE	TRUE	TRUE	TRUE	„True“ gibt an, dass ein Benutzer erst auf den Schlüssel zugreifen darf, wenn der Benutzer authentifiziert ist. Zur Verdeutlichung: Benutzer können erst dann auf Schlüssel in CloudHSM zugreifen, wenn sie authentifiziert sind, selbst wenn dieses Attribut auf „FALSE“ gesetzt ist.

 Note

Sie erhalten eine breitere Unterstützung für Attribute in der PKCS #11-Bibliothek. Weitere Informationen finden Sie unter [Unterstützte PKCS #11-Attribute](#).

Festlegen von Attributen für einen Schlüssel

CloudHsmKeyAttributesMap ist ein [Java-Map](#)-ähnliches Objekt, mit dem Sie Attributwerte für Schlüsselobjekte festlegen können. Die Methoden für CloudHsmKeyAttributesMap funktionieren ähnlich den Methoden, die für die Java-Map-Manipulation verwendet werden.

Sie haben zwei Optionen, um benutzerdefinierte Werte für Attribute festzulegen:

- Verwenden Sie die in der folgenden Tabelle aufgeführten Methoden
- Verwenden Sie Builder-Muster, die später in diesem Dokument gezeigt werden

Attributzuordnungsobjekte unterstützen die folgenden Methoden zum Festlegen von Attributen:

Operation	Rückgabewert	CloudHSMKeyAttributesMap -Methode
Abrufen des Werts eines Schlüsselattributs für einen vorhandenen Schlüssel	Objekt (das den Wert enthält) oder null	get(keyAttribute)
Eingeben des Werts eines Schlüsselattributs	Der vorherige Wert, der dem Schlüsselattribut zugeordnet ist, oder null, wenn keine Zuordnung für ein Schlüsselattribut vorhanden ist	put(keyAttribute, Wert)
Eingeben von Werten für mehrere Schlüsselattribute	–	putAll () keyAttributesMap
Entfernen eines Schlüssel-Wert-Paares aus der Attributzuordnung	Der vorherige Wert, der dem Schlüsselattribut zugeordnet ist, oder null, wenn keine Zuordnung für ein Schlüsselattribut vorhanden ist	remove(keyAttribute)

Note

Alle Attribute, die Sie nicht explizit bestimmen, werden auf die Standardwerte festgelegt, die in der vorherigen Tabelle in [the section called “Unterstützte Attribute”](#) aufgeführt sind.

Beispiel für Builder-Muster

Entwickler finden es im Allgemeinen bequemer, Klassen über das Builder-Muster zu verwenden.
Beispiele:

```

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();

```

Entwickler können auch vordefinierte Attributsätze verwenden, um bewährte Methoden in Schlüsselvorlagen leicht durchzusetzen. Beispiel:

```

//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
```

Festlegen von Attributen für ein Schlüsselpaar

Verwenden Sie die Java-Klasse `CloudHsmKeyPairAttributesMap`, um Schlüsselattribute für ein Schlüsselpaar zu verarbeiten. `CloudHsmKeyPairAttributesMap` fasst zwei `CloudHsmKeyAttributesMap`-Objekte zusammen, eines für einen öffentlichen Schlüssel und eines für einen privaten Schlüssel.

Um einzelne Attribute für den öffentlichen und den privaten Schlüssel separat festzulegen, können Sie die `put()`-Methode für das entsprechende `CloudHsmKeyAttributes`-Zuordnungsobjekt für diesen Schlüssel verwenden. Verwenden Sie die `getPublic()`-Methode, um die Attributzuordnung für den öffentlichen Schlüssel abzurufen, und die `getPrivate()`-Methode, um die Attributzuordnung für den privaten Schlüssel abzurufen. Geben Sie den Wert mehrerer Schlüsselattribute für öffentliche und private Schlüsselpaare zusammen ein, indem Sie die `putAll()`-Methode für eine Attributzuordnung von Schlüsselpaaren als Argument verwenden.

Beispiel für Builder-Muster

Entwickler finden es im Allgemeinen bequemer, Schlüsselattribute über das Builder-Muster festzulegen. Beispiel:

```
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

Weitere Informationen zu dieser proprietären Erweiterung finden Sie im [Javadoc-Archiv](#) und im [Beispiel auf GitHub](#). Um Javadoc zu erkunden, laden Sie das Archiv herunter und erweitern Sie es.

Zusammenführung

Gehen Sie folgendermaßen vor, um Schlüsselattribute für Ihre Schlüsselvorgänge festzulegen:

1. Instanzieren Sie `CloudHsmKeyAttributesMap` für symmetrische Schlüssel oder `CloudHsmKeyPairAttributesMap` für Schlüsselpaare.
2. Definieren Sie das Attributobjekt aus Schritt 1 mit den erforderlichen Schlüsselattributen und -werten.
3. Instanzieren Sie eine `Cavium*ParameterSpec`-Klasse, die Ihrem spezifischen Schlüsseltyp entspricht, und übergeben Sie dieses konfigurierte Attributobjekt an seinen Konstruktor.
4. Übergeben Sie dieses `Cavium*ParameterSpec`-Objekt an eine entsprechende Crypto-Klasse oder -Methode.

Als Referenz enthält die folgende Tabelle die `Cavium*ParameterSpec`-Klassen und -Methoden, die benutzerdefinierte Schlüsselattribute unterstützen.

Schlüsseltyp	Parameter-Spezifikationsklasse	Beispiel-Konstrukoren
Basisklasse	<code>CaviumKeyGenAlgorithmParameterSpec</code>	<code>CaviumKeyGenAlgorithmParameterSpec(CloudHsmKeyAttributesMap keyAttributesMap)</code>

Schlüsseltyp	Parameter-Spezifikationsklasse	Beispiel-Konstrukturen
DES	<code>CaviumDESKeyGenParameterSpec</code>	<code>CaviumDESKeyGenParameterSpec(int keySize, byte[] iv, CloudHsmKeyAttributesMap keyAttributesMap)</code>
RSA	<code>CaviumRSAKeyGenParameterSpec</code>	<code>CaviumRSAKeyGenParameterSpec(int keysize, BigInteger publicExponent, CloudHsmKeyPairAttributesMap keyPairAttributesMap)</code>
Secret	<code>CaviumGenericSecretKeyGenParameterSpec</code>	<code>CaviumGenericSecretKeyGenParameterSpec(int size, CloudHsmKeyAttributesMap keyAttributesMap)</code>
AES	<code>CaviumAESKeyGenParameterSpec</code>	<code>CaviumAESKeyGenParameterSpec(int keySize, byte[] iv, CloudHsmKeyAttributesMap keyAttributesMap)</code>

Schlüsseltyp	Parameter-Spezifikationsklasse	Beispiel-Konstruktor
EC	CaviumECGenParameterSpec	CaviumECGenParameterSpec(String stdName, CloudHsmKeyPairAttributesMap keyPairAttributesMap)

Beispielcode: Generieren und Packen eines Schlüssels

Diese kurzen Codebeispiele veranschaulichen die Schritte für zwei verschiedene Vorgänge: Schlüsselgenerierung und Schlüsselverpackung:

```
// 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);
```

Codebeispiele für die AWS CloudHSM Softwarebibliothek für Java für Client SDK 3

Dieses Thema enthält Ressourcen und Informationen zu Java-Codebeispielen für das AWS CloudHSM Client SDK 3.

Voraussetzungen

Bevor Sie die Beispiele ausführen, müssen Sie die Umgebung einrichten:

- Installieren und konfigurieren Sie den [Java Cryptographic Extension \(JCE\)-Anbieter](#) und das [AWS CloudHSM -Client-Paket](#).
- Legen Sie einen gültigen [HSM-Benutzernamen und ein Passwort fest](#). Crypto-Benutzer (CU)-Berechtigungen sind ausreichend für diese Aufgaben. Ihre Anwendung verwendet diese Anmeldeinformationen, um sich für die einzelnen Beispiele beim HSM anzumelden.
- Entscheiden Sie, wie Anmeldeinformationen für den [JCE-Anbieter](#) bereitgestellt werden sollen.

Codebeispiele

Die folgenden Codebeispiele zeigen Ihnen, wie Sie den [AWS CloudHSM JCE-Anbieter](#) verwenden, um grundlegende Aufgaben auszuführen. Weitere Codebeispiele finden Sie unter [GitHub](#).

- [Anmeldung bei einem HSM](#)
- [Verwalten von Schlüsseln](#)
- [Generieren eines AES-Schlüssels](#)
- [Verschlüsseln und Entschlüsseln mit AES-GCM](#)
- [Verschlüsseln und Entschlüsseln mit AES-CTR](#)
- [Mit D3DES-ECB verschlüsseln und entschlüsseln](#) siehe Hinweis [1](#)
- [Packen und Entpacken von Schlüsseln mit AES-GCM](#)
- [Verpacken und Entpacken von Schlüsseln mit AES](#)
- [Schlüssel mit RSA packen und entpacken](#)
- [Unterstützte Schlüsselattribute verwenden](#)
- [Schlüssel im Schlüsselspeicher aufzählen](#)
- [Verwenden des CloudHSM-Schlüsselspeichers](#)
- [Signieren von Nachrichten in einem Multi-Thread-Beispiel](#)
- [Signieren und Verifizieren mit EC-Schlüsseln](#)

[1] Gemäß den NIST-Richtlinien ist dies für Cluster im FIPS-Modus nach 2023 nicht zulässig. Für Cluster im Nicht-FIPS-Modus ist dies auch nach 2023 zulässig. Details dazu finden Sie unter [FIPS-140-Konformität: Mechanismus 2024 nicht mehr unterstützt](#).

AWS CloudHSM KeyStore Java-Klasse für Client SDK 3

Die AWS CloudHSM **KeyStore** Klasse stellt einen speziellen PKCS12 Schlüsselspeicher bereit, der den Zugriff auf AWS CloudHSM Schlüssel über Anwendungen wie Keytool und Jarsigner ermöglicht. Dieser Schlüsselspeicher kann Zertifikate zusammen mit Ihren Schlüsseldaten speichern und sie mit den in AWS CloudHSM gespeicherten Schlüsseldaten korrelieren.

Note

Da es sich bei Zertifikaten um öffentliche Informationen handelt und um die Speicherkapazität für kryptografische Schlüssel zu maximieren, wird das Speichern von Zertifikaten AWS CloudHSM nicht unterstützt. HSMs

Die AWS CloudHSM KeyStore Klasse implementiert das KeyStore Service Provider Interface (SPI) der Java Cryptography Extension (JCE). [Weitere Informationen zur Verwendung KeyStore finden Sie unter Class. KeyStore](#)

Wählen Sie den entsprechenden Schlüsselspeicher für AWS CloudHSM Client SDK 3

Der AWS CloudHSM Java Cryptographic Extension (JCE) -Anbieter verfügt über einen standardmäßigen, schreibgeschützten Passthrough-Schlüsselspeicher, der alle Transaktionen an das HSM weiterleitet. Dieser Standardschlüsselspeicher unterscheidet sich von dem Spezialschlüsselspeicher. AWS CloudHSM KeyStore In den meisten Situationen erhalten Sie eine bessere Laufzeitleistung und mehr Durchsatz, wenn Sie den Standard verwenden. Sie sollten den nur AWS CloudHSM KeyStore für Anwendungen verwenden, bei denen Sie Unterstützung für Zertifikate und zertifikatsbasierte Operationen benötigen und zusätzlich wichtige Operationen an das HSM auslagern möchten.

Obwohl beide Schlüsselspeicher den JCE-Anbieter für Operationen verwenden, sind sie unabhängige Entitäten und tauschen keine Informationen miteinander aus.

Laden Sie den Standard-Schlüsselspeicher für Ihre Java-Anwendung wie folgt:

```
KeyStore ks = KeyStore.getInstance("Cavium");
```

Laden Sie das spezielle CloudHSM KeyStore wie folgt:

```
KeyStore ks = KeyStore.getInstance("CloudHSM")
```

Initialisieren Sie das SDK 3 für den Client AWS CloudHSM KeyStore

Melden Sie sich auf AWS CloudHSM KeyStore die gleiche Weise an, wie Sie sich beim JCE-Anbieter anmelden. Sie können entweder Umgebungsvariablen oder die Systemeigenschaftendatei verwenden, und Sie sollten sich anmelden, bevor Sie CloudHSM KeyStore verwenden. Ein Beispiel für die Anmeldung bei einem HSM mit dem JCE-Anbieter finden Sie unter [Anmeldung an HSM](#).

Falls gewünscht, können Sie ein Passwort angeben, um die lokale PKCS12 Datei zu verschlüsseln, die Schlüsselspeicherdaten enthält. Wenn Sie den AWS CloudHSM Keystore erstellen, legen Sie das Passwort fest und geben es an, wenn Sie die Methoden load, set und get verwenden.

Instanzieren Sie ein neues CloudHSM-Objekt KeyStore wie folgt:

```
ks.load(null, null);
```

Schreiben Sie Schlüsselspeicherdaten mit der store-Methode in eine Datei. Von da an können Sie den vorhandenen Schlüsselspeicher mit der load-Methode mit der Quelldatei und dem Passwort wie folgt laden:

```
ks.load(inputStream, password);
```

Verwenden Sie das SDK 3 für den Client AWS CloudHSM KeyStore AWS CloudHSM

[Ein KeyStore CloudHSM-Objekt wird in der Regel über eine Drittanbieteranwendung wie jarsigner oder keytool verwendet.](#) Sie können auch direkt mit Code auf das Objekt zugreifen.

AWS CloudHSM KeyStore entspricht der [KeyStoreJCE-Klassenspezifikation](#) und bietet die folgenden Funktionen.

- load

Lädt den Schlüsselspeicher aus dem angegebenen Eingabe-Stream. Wenn beim Speichern des Schlüsselspeichers ein Passwort festgelegt wurde, muss dasselbe Passwort angegeben werden, damit das Laden erfolgreich ist. Setzen Sie beide Parameter auf null, um einen neuen leeren Schlüsselspeicher zu initialisieren.

```
KeyStore ks = KeyStore.getInstance("CloudHSM");  
ks.load(inputStream, password);
```

- **aliases**

Gibt eine Aufzählung der Aliasnamen aller Einträge in der angegebenen Schlüsselspeicherinstanz zurück. Zu den Ergebnissen gehören Objekte, die lokal in der PKCS12 Datei gespeichert sind, und Objekte, die sich auf dem HSM befinden.

Beispiel-Code:

```
KeyStore ks = KeyStore.getInstance("CloudHSM");  
for(Enumeration<String> entry = ks.aliases(); entry.hasMoreElements();) {  
    String label = entry.nextElement();  
    System.out.println(label);  
}
```

- **ContainsAlias**

Gibt „true“ zurück, wenn der Schlüsselspeicher Zugriff auf mindestens ein Objekt mit dem angegebenen Alias hat. Der Schlüsselspeicher überprüft Objekte, die lokal in der PKCS12 Datei gespeichert sind, und Objekte, die sich auf dem HSM befinden.

- **DeleteEntry**

Löscht einen Zertifikatseintrag aus der lokalen PKCS12 Datei. Das Löschen von Schlüsseldaten, die in einem HSM gespeichert sind, wird nicht unterstützt. AWS CloudHSM KeyStore Sie können Schlüssel mit dem [Key_mgmt_util](#)-Tool von CloudHSM löschen.

- **GetCertificate**

Gibt das Zertifikat zurück, das einem Alias zugeordnet ist, falls verfügbar. Wenn der Alias nicht existiert oder auf ein Objekt verweist, das kein Zertifikat ist, gibt die Funktion NULL zurück.

```
KeyStore ks = KeyStore.getInstance("CloudHSM");  
Certificate cert = ks.getCertificate(alias)
```

- **GetCertificateAlias**

Gibt den Namen (Alias) des ersten Schlüsselspeichereintrags zurück, dessen Daten mit dem angegebenen Zertifikat übereinstimmen.


```
KeyStore ks = KeyStore.getInstance("CloudHSM");  
String alias = ks.getCertificateAlias(cert)
```

- **GetCertificateChain**

Gibt die Zertifikatkette zurück, die dem angegebenen Alias zugeordnet ist. Wenn der Alias nicht existiert oder auf ein Objekt verweist, das kein Zertifikat ist, gibt die Funktion NULL zurück.

- **GetCreationDate**

Gibt das Erstellungsdatum des durch den angegebenen Alias identifizierten Eintrags zurück. Wenn kein Erstellungsdatum verfügbar ist, gibt die Funktion das Datum zurück, an dem das Zertifikat gültig wurde.

- **GetKey**

GetKey wird an das HSM übergeben und gibt ein Schlüsselobjekt zurück, das dem angegebenen Label entspricht. Da es das HSM getKey direkt abfragt, kann es für jeden Schlüssel auf dem HSM verwendet werden, unabhängig davon, ob er von der generiert wurde. KeyStore

```
Key key = ks.getKey(keyLabel, null);
```

- **IsCertificateEntry**

Überprüft, ob der Eintrag mit dem angegebenen Alias einen Zertifikateintrag darstellt.

- **IsKeyEntry**

Überprüft, ob der Eintrag mit dem angegebenen Alias einen Schlüsseleintrag darstellt. Die Aktion durchsucht sowohl die PKCS12 Datei als auch das HSM nach dem Alias.

- **SetCertificateEntry**

Weist dem angegebenen Alias das angegebene Zertifikat zu. Wenn der angegebene Alias bereits verwendet wird, um einen Schlüssel oder ein Zertifikat zu identifizieren, wird ein `KeyStoreException` ausgelöst. Sie können JCE-Code verwenden, um das Schlüsselobjekt abzurufen, und dann die `KeyStore SetKeyEntry` Methode verwenden, um das Zertifikat dem Schlüssel zuzuordnen.

- **SetKeyEntry mit byte[]-Schlüssel**

Diese API wird derzeit vom Client-SDK 3 nicht unterstützt.

- **SetKeyEntry mit Key-Objekt**

Weist den angegebenen Schlüssel dem angegebenen Alias zu und speichert ihn im HSM. Wenn das Key-Objekt nicht vom Typ `CaviumKey` ist, wird der Schlüssel als extrahierbarer Sitzungsschlüssel in das HSM importiert.

Wenn das Key-Objekt vom Typ `PrivateKey` ist, muss es von einer entsprechenden Zertifikatkette begleitet werden.

Wenn der Alias bereits vorhanden ist, wird ein `SetKeyEntry`- Aufruf ausgelöst, und `KeyStoreException` verhindert, dass der Schlüssel überschrieben wird. Wenn der Schlüssel überschrieben werden muss, verwenden Sie hierfür KMU oder JCE.

- `EngineSize`

Gibt die Anzahl der Einträge im Schlüsselspeicher zurück.

- `Store`

Speichert den Schlüsselspeicher für den angegebenen Ausgabestrom als PKCS12 Datei und sichert ihn mit dem angegebenen Passwort. Darüber hinaus bleiben alle geladenen Schlüssel (die über `setKey`-Aufrufe gesetzt werden) bestehen.

Kryptografie-API: Next Generation (CNG) und Key Storage Providers (KSP) für AWS CloudHSM

Der AWS CloudHSM Client für Windows umfasst CNG- und KSP-Anbieter.

Schlüsselspeicheranbieter (KSPs) ermöglichen das Speichern und Abrufen von Schlüsseln. Wenn Sie beispielsweise auf einem Windows-Server die Rolle Microsoft Active Directory-Zertifikatsdienste (AD CS) hinzufügen und einen neuen privaten Schlüssel für die Zertifizierungsstelle (CA) erstellen, können Sie KSP die Verwaltung des Schlüsselspeichers überlassen. Wenn Sie die AD CS-Rolle konfigurieren, können Sie diesen KSP wählen. Weitere Informationen finden Sie unter [Erstellen einer Windows Server-CA](#).

Kryptografie-API: Next-Generation (CNG) ist eine Kryptografie-API für das Betriebssystem Microsoft Windows. Mit CNG können Kryptografietechniken zum Schützen von Windows-Anwendungen einsetzen. Auf hoher Ebene bietet die AWS CloudHSM Implementierung von CNG die folgenden Funktionen:

- Kryptografische Primitive — ermöglichen es Ihnen, grundlegende kryptografische Operationen durchzuführen.
- Schlüsselimport und -export — ermöglicht es Ihnen, asymmetrische Schlüssel zu importieren und zu exportieren.
- Data Protection API (CNG DPAPI) — ermöglicht Ihnen das einfache Verschlüsseln und Entschlüsseln von Daten.
- Schlüsselspeicherung und -abruf — ermöglicht es Ihnen, den privaten Schlüssel eines asymmetrischen key pair sicher zu speichern und zu isolieren.

Themen

- [Überprüfen Sie die KSP- und CNG-Anbieter für AWS CloudHSM](#)
- [Voraussetzungen für die Verwendung des AWS CloudHSM Windows-Clients](#)
- [Zuordnen eines AWS CloudHSM Schlüssels zu einem Zertifikat](#)
- [Codebeispiel für CNG-Anbieter für AWS CloudHSM](#)

Überprüfen Sie die KSP- und CNG-Anbieter für AWS CloudHSM

Die KSP- und CNG-Anbieter werden bei der Installation des AWS CloudHSM Windows-Clients installiert. Sie können den Client installieren, indem Sie die unter [Installieren des Clients \(Windows\)](#) beschriebenen Schritte durchführen.

Verwenden Sie die folgenden Abschnitte, um die Installation der Anbieter zu überprüfen.

Konfigurieren Sie den AWS CloudHSM Windows-Client und führen Sie ihn aus

Bevor Sie den Windows CloudHSM-Client starten können, müssen Sie die Anforderungen unter [Voraussetzungen](#) erfüllen. Aktualisieren Sie anschließend die von den Anbietern verwendeten Konfigurationsdateien und starten Sie den Client, indem Sie die folgenden Schritte ausführen. Sie müssen diese Schritte ausführen, wenn Sie die KSP- und CNG-Anbieter zum ersten Mal verwenden und nachdem Sie Ihrem Cluster etwas hinzugefügt oder daraus entfernt HSMs haben. Auf diese Weise AWS CloudHSM ist es möglich, Daten zu synchronisieren und die Konsistenz HSMs im gesamten Cluster aufrechtzuerhalten.

Schritt 1: Stoppen Sie den AWS CloudHSM Client

Bevor Sie die von den Anbietern verwendeten Konfigurationsdateien aktualisieren, beenden Sie den AWS CloudHSM Client. Ist der Client bereits angehalten, hat das Ausführen des Anhaltebefehls keine Auswirkungen.

- Für Windows-Client 1.1.2 und höher:

```
C:\Program Files\Amazon\CloudHSM>net.exe stop AWSCloudHSMClient
```

- Für Windows-Clients 1.1.1 und früher:

Verwenden Sie Strg + C im Befehlsfenster, in dem Sie den AWS CloudHSM Client gestartet haben.

Schritt 2: Aktualisieren Sie die AWS CloudHSM Konfigurationsdateien

In diesem Schritt -a wird der Parameter des [Configure-Tools](#) verwendet, um der Konfigurationsdatei die elastic network interface (ENI) -IP-Adresse eines der HSMs im Cluster befindlichen Objekte hinzuzufügen.

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

Um die ENI-IP-Adresse eines HSM in Ihrem Cluster abzurufen, navigieren Sie zur AWS CloudHSM Konsole, wählen Sie Cluster und dann den gewünschten Cluster aus. Sie können auch den [DescribeClusters](#) Vorgang, den Befehl [describe-clusters](#) oder das Cmdlet verwenden. [Get-HSM2Cluster](#) PowerShell Geben Sie nur eine ENI-IP-Adresse ein. Es ist unerheblich, welche ENI-IP-Adresse Sie verwenden.

Schritt 3: Starten Sie den Client AWS CloudHSM

Starten Sie als Nächstes den AWS CloudHSM Client oder starten Sie ihn neu. Wenn der AWS CloudHSM Client gestartet wird, verwendet er die ENI-IP-Adresse in seiner Konfigurationsdatei, um den Cluster abzufragen. Anschließend fügt er der Cluster-Informationsdatei die ENI-IP-Adressen aller Benutzer HSMs im Cluster hinzu.

- Für Windows-Client 1.1.2 und höher:

```
C:\Program Files\Amazon\CloudHSM>net.exe start AWSCloudHSMClient
```

- Für Windows-Clients 1.1.1 und früher:

```
C:\Program Files\Amazon\CloudHSM>start "cloudhsm_client" cloudhsm_client.exe C:  
\ProgramData\Amazon\CloudHSM\data\cloudhsm_client.cfg
```

Überprüfen der KSP- und CNG-Anbieter

Sie können einen der folgenden Befehle verwenden, um zu ermitteln, welche Provider im System installiert sind. Die Befehle listen die registrierten KSP- und CNG-Provider auf. Der AWS CloudHSM - Client muss zum betreffenden Zeitpunkt nicht ausgeführt werden.

```
PS C:\> & "C:\Program Files\Amazon\CloudHSM\ksp_config.exe" -enum
```

```
PS C:\> & "C:\Program Files\Amazon\CloudHSM\cng_config.exe" -enum
```

Um zu überprüfen, ob die KSP- und CNG-Anbieter auf Ihrer Windows-Server-EC2-Instance installiert sind, sollten Sie die folgenden Einträge in der Liste sehen:

```
Cavium CNG Provider  
Cavium Key Storage Provider
```

Wenn der CNG-Provider fehlt, führen Sie folgenden Befehl aus.

```
PS C:\> & "C:\Program Files\Amazon\CloudHSM\cng_config.exe" -register
```

Wenn der KSP-Anbieter fehlt, führen Sie folgenden Befehl aus.

```
PS C:\> & "C:\Program Files\Amazon\CloudHSM\ksp_config.exe" -register
```

Voraussetzungen für die Verwendung des AWS CloudHSM Windows-Clients

Bevor Sie den AWS CloudHSM Windows-Client starten und die KSP- und CNG-Anbieter verwenden können, müssen Sie die Anmeldeinformationen für das HSM auf Ihrem System einrichten. Sie können Anmeldeinformationen entweder über die Windows-Anmeldeinformationsverwaltung oder die Systemumgebungsvariable festlegen. Wir empfehlen, zum Speichern der Anmeldeinformationen die Windows-Anmeldeinformationsverwaltung zu verwenden. Diese Option ist in der AWS CloudHSM Client-Version 2.0.4 und höher verfügbar. Die Verwendung von Umgebungsvariablen ist einfacher einzurichten, aber weniger sicher als die Windows-Anmeldeinformationsverwaltung.

Windows-Anmeldeinformationsverwaltung

Sie können entweder das Dienstprogramm `set_cloudhsm_credentials` oder die Schnittstelle der Windows-Anmeldeinformationsverwaltung verwenden.

- Verwenden des Dienstprogramms **`set_cloudhsm_credentials`**:

Das Dienstprogramm `set_cloudhsm_credentials` ist in Ihrem Windows-Installationsprogramm enthalten. Sie können dieses Dienstprogramm verwenden, um HSM-Anmeldeinformationen bequem an die Windows-Anmeldeinformationsverwaltung zu übergeben. Wenn Sie dieses Dienstprogramm aus der Quelle kompilieren möchten, können Sie den Python-Code verwenden, der im Installationsprogramm enthalten ist.

1. Rufen Sie den Ordner `C:\Program Files\Amazon\CloudHSM\tools\` auf.
2. Führen Sie die Datei `set_cloudhsm_credentials.exe` mit den Parametern für CU-Benutzername und -Passwort aus.

```
set_cloudhsm_credentials.exe --username <CU USER> --password <CU PASSWORD>
```

- Verwenden der Schnittstelle der Anmeldeinformationsverwaltung:

Sie können die Schnittstelle der Anmeldeinformationsverwaltung verwenden, um Ihre Anmeldeinformationen manuell zu verwalten.

1. Um die Anmeldeinformationsverwaltung zu öffnen, geben Sie `credential manager` in das Suchfeld in der Taskleiste ein und wählen Sie Anmeldeinformationsverwaltung aus.
2. Wählen Sie Windows-Anmeldeinformationen aus, um Windows-Anmeldeinformationen zu verwalten.
3. Wählen Sie Generische Anmeldeinformationen hinzufügen aus und füllen Sie die Details wie folgt aus:
 - Geben Sie unter Internet- oder Netzwerkadresse den Zielnamen als `cloudhsm_client` ein.
 - Geben Sie unter Benutzername und Passwort die CU-Anmeldeinformationen ein.
 - Klicken Sie auf OK.

Systemumgebungsvariablen

Sie können Systemumgebungsvariablen festlegen, die ein HSM und einen [Crypto-Benutzer](#) (Crypto User, CU) für Ihre Windows-Anwendung identifizieren. Sie können mit dem [setx-Befehl](#) Systemumgebungsvariablen konfigurieren oder Sie können [programmgesteuert](#) oder auf der

Registerkarte Advanced (Erweitert) in den System Properties (Systemeigenschaften) der Windows-Systemsteuerung permanente Systemumgebungsvariablen konfigurieren.

 Warning

Wenn Sie Anmeldeinformationen über Systemumgebungsvariablen festlegen, ist das Passwort in Klartext auf dem System eines Benutzers verfügbar. Verwenden Sie die Windows-Anmeldeinformationsverwaltung, um dieses Problem zu beheben.

Konfigurieren Sie die folgenden Systemumgebungsvariablen:

n3fips_password=<CU USERNAME>:<CU PASSWORD>

Identifiziert einen [Crypto-Benutzer](#) (CU, Crypto User) im HSM und stellt alle erforderlichen Anmeldeinformationen bereit. Ihre Anwendung führt die Authentifizierung durch und wird als dieser CU ausgeführt. Die Anwendung verfügt über die Berechtigungen dieses CU und kann nur die Schlüssel anzeigen und verwalten, die sich im Besitz dieses CU befinden und von ihm freigegeben werden. Um einen neuen CU zu erstellen, verwenden Sie [createUser](#). Verwenden Sie [ListUsers CUs](#), um vorhandene zu finden.

Beispiel:

```
setx /m n3fips_password test_user:password123
```

Zuordnen eines AWS CloudHSM Schlüssels zu einem Zertifikat

Bevor Sie AWS CloudHSM Schlüssel mit Tools von Drittanbietern wie denen von Microsoft verwenden können [SignTool](#), müssen Sie die Metadaten des Schlüssels in den lokalen Zertifikatsspeicher importieren und die Metadaten einem Zertifikat zuordnen. Um die Metadaten des Schlüssels zu importieren, verwenden Sie das Dienstprogramm `import_key.exe`, das in CloudHSM Version 3.0 und höher enthalten ist. Die folgenden Schritte enthalten zusätzliche Informationen und eine Beispielausgabe.

Schritt 1: Importieren Ihres Zertifikats

Unter Windows sollten Sie auf das Zertifikat doppelklicken können, um es in den lokalen Zertifikatsspeicher zu importieren.

Wenn ein Doppelklick jedoch nicht funktioniert, verwenden Sie das [Microsoft Certreq-Tool](#), um das Zertifikat in den Zertifikatmanager zu importieren. Beispiel:

```
certreq -accept <certificatename>
```

Wenn diese Aktion fehlschlägt und Sie den Fehler Key not found erhalten, fahren Sie mit Schritt 2 fort. Wenn das Zertifikat in Ihrem Schlüsselspeicher angezeigt wird, haben Sie die Aufgabe abgeschlossen, und es ist keine weitere Aktion erforderlich.

Schritt 2: Erfassen von Zertifikatinformationen

Wenn der vorherige Schritt nicht erfolgreich war, müssen Sie Ihren privaten Schlüssel einem Zertifikat zuordnen. Bevor Sie jedoch die Zuordnung erstellen können, müssen Sie zuerst den eindeutigen Containernamen und die Seriennummer des Zertifikats finden. Verwenden Sie ein Hilfsprogrammcertutil, z. B. um die benötigten Zertifikatsinformationen anzuzeigen. Die folgende Beispielausgabe von certutil zeigt den Namen und die Seriennummer des Containers.

```
===== 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
```

Schritt 3: Verknüpfen Sie den AWS CloudHSM privaten Schlüssel mit dem Zertifikat

Um den Schlüssel mit dem Zertifikat zu verknüpfen, müssen Sie zunächst [den AWS CloudHSM Client-Daemon starten](#). Verwenden Sie dann import_key.exe (in CloudHSM Version 3.0 und höher enthalten), um den privaten Schlüssel dem Zertifikat zuzuordnen. Wenn Sie das Zertifikat angeben, verwenden Sie seinen einfachen Containernamen. Das folgende Beispiel zeigt den Befehl und die Antwort. Diese Aktion kopiert nur die Metadaten des Schlüssels; der Schlüssel verbleibt auf dem HSM.

```
$> import_key.exe -RSA CertReq-39c04db0-6aa9-4310-93db-db0d9669f42c
```



```
Successfully opened Microsoft Software Key Storage Provider : 0NCryptOpenKey failed :
80090016
```

Schritt 4: Aktualisieren des Zertifikatspeichers

Stellen Sie sicher, dass der AWS CloudHSM Client-Daemon noch läuft. Verwenden Sie dann das `certutil Verb,-repairstore`, um die Seriennummer des Zertifikats zu aktualisieren. Das folgende Beispiel zeigt den Befehl und die Ausgabe. Informationen zu dem [-repairstoreVerb](#) finden Sie in der Microsoft-Dokumentation.

```
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.
```

Nachdem Sie die Seriennummer des Zertifikats aktualisiert haben, können Sie dieses Zertifikat und den entsprechenden AWS CloudHSM privaten Schlüssel mit jedem Signaturtool eines Drittanbieters unter Windows verwenden.

Codebeispiel für CNG-Anbieter für AWS CloudHSM

 **** Nur Beispielcode — Nicht für Produktionszwecke geeignet ****

Dieser Beispielcode dient lediglich der Veranschaulichung. Führen Sie den Code nicht in der Produktionsumgebung aus.

Das folgende Beispiel zeigt, wie Sie die registrierten Crypto-Anbieter auf Ihrem System auflisten, um den mit dem CloudHSM-Client für Windows installierten CNG-Anbieter zu finden. Das Beispiel zeigt außerdem, wie ein asymmetrisches Schlüsselpaar erstellt und zum Signieren von Daten verwendet wird.

Important

Bevor Sie dieses Beispiel ausführen, müssen Sie die HSM-Anmeldeinformationen wie in den Voraussetzungen beschrieben einrichten. Details hierzu finden Sie unter [Voraussetzungen für die Verwendung des AWS CloudHSM Windows-Clients](#).

```
// 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;
}
```

Integration von Drittanbieteranwendungen mit AWS CloudHSM

Einige der [Anwendungsfälle](#) für AWS CloudHSM beinhalten die Integration von Softwareanwendungen von Drittanbietern mit dem HSM in Ihrem AWS CloudHSM Cluster. Durch die Integration von Software von Drittanbietern können Sie eine Vielzahl von sicherheitsrelevanten Zielen erreichen. AWS CloudHSM In den folgenden Themen wird beschrieben, wie Sie einige dieser Ziele umsetzen.

Themen

- [Verbessern Sie die Sicherheit Ihres Webservers mit SSL/TLS Offload in AWS CloudHSM](#)
- [Konfigurieren Sie Windows Server als Zertifizierungsstelle \(CA\) mit AWS CloudHSM](#)
- [Transparente Datenverschlüsselung \(TDE\) für Oracle-Datenbanken mit AWS CloudHSM](#)
- [Verwenden Sie Microsoft SignTool mit AWS CloudHSM , um Dateien zu signieren](#)
- [Integration von Java Keytool und Jarsigner mit AWS CloudHSM](#)
- [Verwenden Sie das Microsoft Manifest Generation and Editing Tool \(Mage.exe\) mit AWS CloudHSM , um Dateien zu signieren](#)
- [Integrationen anderer Drittanbieter mit AWS CloudHSM](#)

Verbessern Sie die Sicherheit Ihres Webservers mit SSL/TLS Offload in AWS CloudHSM

Webserver und ihre Clients (Webbrowser) können die Protokolle Secure Sockets Layer (SSL) oder Transport Layer Security (TLS) verwenden, um die Identität des Webservers zu bestätigen und eine sichere Verbindung herzustellen, die Webseiten oder andere Daten über das Internet sendet und empfängt. Dies wird allgemein als HTTPS bezeichnet. Der Webserver verwendet ein öffentlich-privates key pair und ein SSL/TLS öffentliches Schlüsselzertifikat, um eine HTTPS-Sitzung mit jedem Client einzurichten. Dieser Prozess erfordert eine Menge Rechenleistung für Webserver, aber Sie können einen Teil davon auf Ihren AWS CloudHSM Cluster auslagern, was als SSL-Beschleunigung bezeichnet wird. Das Auslagern reduziert den Rechenaufwand auf Ihren Webservern und bietet zusätzliche Sicherheit, da die privaten Schlüssel der Server darin gespeichert werden. HSMs

Die folgenden Themen bieten einen Überblick über die AWS CloudHSM Funktionsweise von SSL/TLS Offload with sowie Tutorials zur Einrichtung von SSL/TLS Offload With auf den folgenden Plattformen. AWS CloudHSM

Für Linux verwenden Sie OpenSSL Dynamic Engine auf der [NGINX](#)- oder [Apache-HTTP-Server](#)-Webserversoftware

Für Windows verwenden Sie die Webserver-Software [Internet Information Services \(IIS\) für Windows Server](#).

Themen

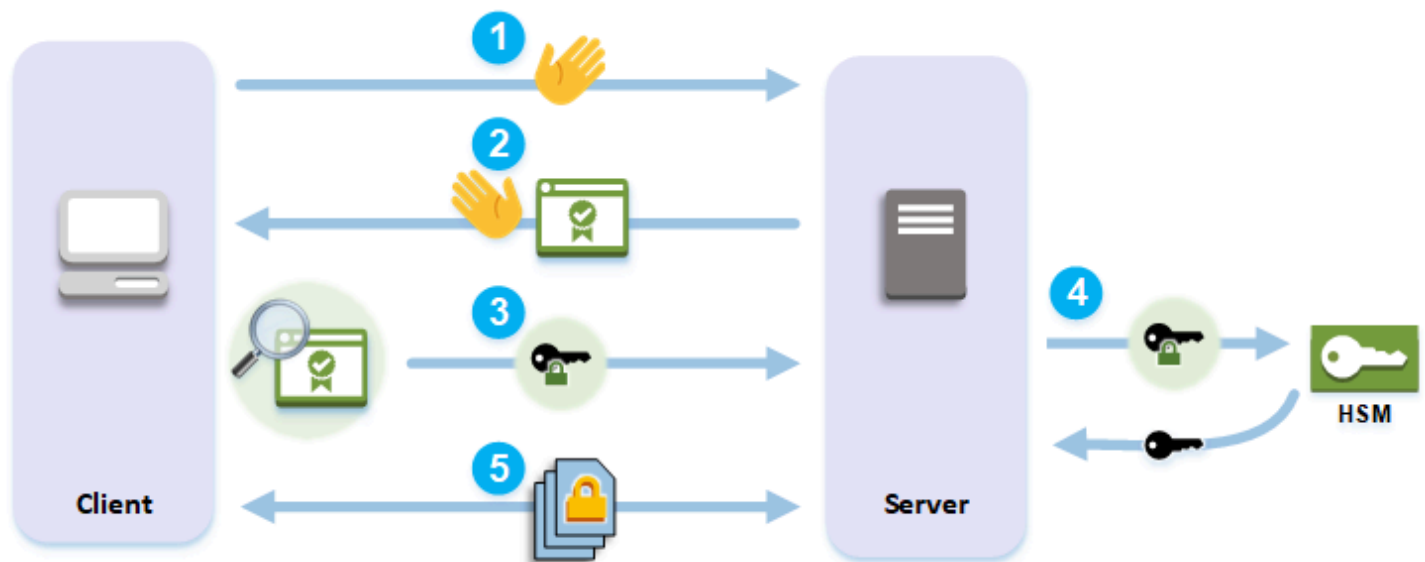
- [So SSL/TLS funktioniert Offload mit AWS CloudHSM](#)
- [AWS CloudHSM SSL/TLS-Offload unter Linux mit NGINX oder Apache mit OpenSSL](#)
- [AWS CloudHSM SSL/TLS-Offload unter Linux mit NGINX oder mit OpenSSL Provider HAProxy](#)
- [AWS CloudHSM SSL/TLS-Offload unter Linux mit Tomcat mit JSSE](#)
- [AWS CloudHSM SSL/TLS-Offload unter Windows mithilfe von IIS mit KSP](#)
- [Einen Load Balancer mit Elastic Load Balancing hinzufügen für AWS CloudHSM\(optional\)](#)

So SSL/TLS funktioniert Offload mit AWS CloudHSM

Um eine HTTPS-Verbindung aufzubauen, führt Ihr Webserver einen Handshake-Prozess mit Clients durch. Im Rahmen dieses Prozesses verlagert der Server einen Teil der kryptografischen Verarbeitung auf den AWS CloudHSM Cluster, wie HSMs in der folgenden Abbildung dargestellt. Jeder Schritt des Prozesses wird im Folgenden erläutert.

Note

Das folgende Bild und der Prozess gehen davon aus, dass RSA für die Serververifikation und den Schlüsselaustausch verwendet wird. Der Prozess sieht etwas anders aus, wenn Diffie-Hellman anstelle von RSA verwendet wird.



1. Der Client sendet eine Hello-Nachricht an den Server.
2. Der Server antwortet mit einer Hello-Nachricht und sendet das Zertifikat des Servers.
3. Der Client führt die folgenden Aktionen durch:
 - a. Überprüft, ob das SSL/TLS Serverzertifikat mit einem Stammzertifikat signiert ist, dem der Client vertraut.
 - b. Er extrahiert den öffentlichen Schlüssel aus dem Zertifikat des Servers.
 - c. Generiert ein Pre-Master-Secret und verschlüsselt es mit dem öffentlichen Schlüssel des Servers.
 - d. Sendet das verschlüsselte Pre-Master-Secret an den Server.
4. Um das Pre-Master-Secret des Clients zu entschlüsseln, sendet der Server es an das HSM. Das HSM verwendet den privaten Schlüssel im HSM, um das Pre-Master-Secret zu entschlüsseln, und sendet dann das Pre-Master-Secret an den Server. Unabhängig davon verwenden Client und Server jeweils das Pre-Master-Secret und einige Informationen aus den Hello-Nachrichten, um ein Master-Secret zu berechnen.
5. Der Handshake-Vorgang ist beendet. Für den Rest der Sitzung werden alle Nachrichten, die zwischen dem Client und dem Server gesendet werden, mit Derivaten des Master-Geheimnisses verschlüsselt.

Informationen zur Konfiguration von SSL/TLS Offload mit AWS CloudHSM finden Sie in einem der folgenden Themen:

- [AWS CloudHSM SSL/TLS-Offload unter Linux mit NGINX oder Apache mit OpenSSL](#)

- [AWS CloudHSM SSL/TLS-Offload unter Linux mit Tomcat mit JSSE](#)
- [AWS CloudHSM SSL/TLS-Offload unter Windows mithilfe von IIS mit KSP](#)

AWS CloudHSM SSL/TLS-Offload unter Linux mit NGINX oder Apache mit OpenSSL

Dieses Thema enthält step-by-step Anweisungen zum Einrichten von SSL/TLS Offload mit AWS CloudHSM auf einem Linux-Webserver.

Themen

- [-Übersicht](#)
- [Schritt 1: Einrichten der Voraussetzungen](#)
- [Schritt 2: Generieren Sie den privaten Schlüssel und das Zertifikat SSL/TLS](#)
- [Schritt 3: Konfigurieren des Webserver](#)
- [Schritt 4: Aktivieren von HTTPS-Datenverkehr und Verifizieren des Zertifikats](#)

-Übersicht

Die [NGINX](#)- und [Apache HTTP Server](#)-Webserver-Software unter Linux ist nativ mit [OpenSSL](#) integriert, um HTTPS zu unterstützen. Die [AWS CloudHSM dynamische Engine für OpenSSL](#) bietet eine Schnittstelle, die es der Webserver-Software ermöglicht, die HSMs in Ihrem Cluster für kryptografisches Offloading und Schlüsselspeicherung zu verwenden. Die OpenSSL-Engine ist eine Brücke, die den Webserver mit dem AWS CloudHSM -Cluster verbindet.

Um diese Anleitung abzuschließen, müssen Sie zuerst entscheiden, ob Sie unter Linux die NGINX- oder Apache Webserver-Software verwenden möchten. Dann erfahren Sie, wie Sie folgende Aufgaben ausführen:

- Installieren der Web-Server-Software auf einer Amazon EC2-Instance
- Konfigurieren Sie die Webserver-Software so, dass sie HTTPS mit einem privaten Schlüssel unterstützt, der in Ihrem AWS CloudHSM -Cluster gespeichert ist.
- (Optional) Verwenden Sie Amazon EC2, um eine zweite Webserver-Instance zu erstellen, und Elastic Load Balancing, um einen Load Balancer zu erstellen. Mit einem Load Balancer kann die Leistung durch Verteilung der Arbeitslast auf mehrere Server gesteigert werden. Sie kann auch für Redundanz und eine höhere Verfügbarkeit sorgen, falls ein oder mehrere Webserver ausfallen.

Wenn Sie bereit sind, sehen Sie sich [Schritt 1: Einrichten der Voraussetzungen](#) an.

Schritt 1: Einrichten der Voraussetzungen

Verschiedene Plattformen erfordern unterschiedliche Voraussetzungen. Verwenden Sie den Abschnitt mit den Voraussetzungen unten, der zu Ihrer Plattform passt.

Voraussetzungen für das Client-SDK 5

Um das SSL/TLS Webserver-Offload mit dem Client SDK 5 einzurichten, benötigen Sie Folgendes:

- Ein aktiver AWS CloudHSM Cluster mit mindestens zwei Hardware-Sicherheitsmodulen (HSM)

Note

Sie können einen einzelnen HSM-Cluster verwenden, müssen aber zuerst die Haltbarkeit der Client-Schlüssel deaktivieren. Weitere Informationen finden Sie unter [Einstellungen für die Haltbarkeit von Client-Schlüsseln verwalten](#) und [Client-SDK 5 Configure Tool](#).

- Eine Amazon EC2-Instance, auf der ein Linux-Betriebssystem ausgeführt wird und die folgende Software installiert ist:
 - Ein Webserver (entweder NGINX oder Apache)
 - Die OpenSSL Dynamic Engine für Client-SDK 5
- Ein [Crypto-Benutzer](#) (CU), der den privaten Schlüssel des Webserver auf dem HSM besitzen und verwalten soll.

So richten Sie eine Linux-Webserver-Instance auf dem HSM ein und erstellen einen CU

1. Installieren und konfigurieren Sie die OpenSSL Dynamic Engine für AWS CloudHSM. Weitere Informationen zur Installation der OpenSSL Dynamic Engine finden Sie unter [OpenSSL Dynamic Engine für Client-SDK 5](#).
2. Installieren Sie auf einer EC2-Linux-Instance, die Zugriff auf Ihren Cluster hat, entweder den NGINX- oder den Apache-Webserver:

Amazon Linux 2

- Informationen zum Herunterladen der neuesten Version von NGINX auf Amazon Linux 2 finden Sie auf der [NGINX-Website](#).

Die neueste Version von NGINX, die für Amazon Linux 2 verfügbar ist, verwendet eine Version von OpenSSL, die neuer ist als die Systemversion von OpenSSL. Nach der Installation von NGINX müssen Sie einen symbolischen Link von der AWS CloudHSM OpenSSL Dynamic Engine-Bibliothek zu dem Speicherort erstellen, den diese Version von OpenSSL erwartet

```
$ 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 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
```

3. Verwenden Sie die CloudHSM-CLI, um einen [Crypto-Benutzer](#) zu erstellen. Weitere Informationen zur Verwaltung von HSM-Benutzern finden Sie unter [HSM-Benutzer mit der CloudHSM-CLI verwalten](#).

Tip

Merken Sie sich den CU-Benutzernamen und das Passwort. Sie benötigen sie später beim Generieren oder Importieren des privaten HTTPS-Schlüssels und -Zertifikats für Ihren Webserver.

Nachdem Sie diese Schritte abgeschlossen haben, fahren Sie mit [Schritt 2: Generieren Sie den privaten Schlüssel und das Zertifikat SSL/TLS](#) fort.

Hinweise

- Um Security-Enhanced Linux (SELinux) und Webserver zu verwenden, müssen Sie ausgehende TCP-Verbindungen auf Port 2223 zulassen, dem Port, den Client SDK 5 für die Kommunikation mit dem HSM verwendet.
- Um einen Cluster zu erstellen und zu aktivieren und einer EC2-Instance Zugriff auf den Cluster zu gewähren, führen Sie die Schritte unter [Erste Schritte mit AWS CloudHSM](#) aus. Die ersten Schritte bieten step-by-step Anweisungen zum Erstellen eines aktiven Clusters mit einem HSM und einer Amazon EC2 EC2-Client-Instance. Sie können diese Client-Instance als Ihren Webserver verwenden.
- Um zu vermeiden, dass die Haltbarkeit von Client-Schlüsseln deaktiviert wird, fügen Sie Ihrem Cluster mehr als ein HSM hinzu. Weitere Informationen finden Sie unter [Hinzufügen eines HSM zu einem Cluster AWS CloudHSM](#).
- Um sich mit Ihrer Client-Instance zu verbinden, können Sie SSH oder PuTTY verwenden. Weitere Informationen finden Sie unter [Herstellung einer Verbindung zu Ihrer Linux-Instance mit SSH](#) oder [Herstellung einer Verbindung zu Ihrer Linux-Instance von Windows mit PuTTY](#) in der Amazon EC2-Dokumentation.

Schritt 2: Generieren Sie den privaten Schlüssel und das Zertifikat SSL/TLS

Um HTTPS zu aktivieren, benötigt Ihre Webserver-Anwendung (NGINX oder Apache) einen privaten Schlüssel und ein entsprechendes Zertifikat. SSL/TLS Um das SSL/TLS Webserver-Offload mit verwenden zu können AWS CloudHSM, müssen Sie den privaten Schlüssel in einem HSM in Ihrem Cluster speichern. AWS CloudHSM Sie generieren zunächst einen privaten Schlüssel und verwenden diesen Schlüssel, um eine Certificate Signing Request (CSR) zu erstellen. Anschließend exportieren Sie einen gefälschten privaten PEM-Schlüssel aus dem HSM. Dabei handelt es sich um eine private Schlüsseldatei im PEM-Format, die einen Verweis auf den privaten Schlüssel enthält, der auf dem HSM gespeichert ist (es ist nicht der eigentliche private Schlüssel). Ihr Webserver verwendet die gefälschte private PEM-Schlüsseldatei, um den privaten Schlüssel auf dem HSM beim Ausladen zu identifizieren. SSL/TLS

Generieren eines privaten Schlüssels und Zertifikats

Generieren eines privaten Schlüssels

In diesem Abschnitt erfahren Sie, wie Sie mithilfe der [CloudHSM-CLI](#) ein Schlüsselpaar generieren. Sobald Sie ein key pair im HSM generiert haben, können Sie es als gefälschte PEM-Datei exportieren und das entsprechende Zertifikat generieren.

Installieren und konfigurieren Sie die CloudHSM-CLI

1. [Installieren und konfigurieren Sie](#) die CloudHSM-CLI.
2. Verwenden Sie den folgenden Befehl, um die CloudHSM-CLI zu starten.

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

3. Führen Sie den folgenden Befehl aus, um sich beim HSM anzumelden. Ersetzen Sie es *<user name>* durch den Benutzernamen Ihres Krypto-Benutzers

```
Command: login --username <user name> --role crypto-user
```

Generieren eines privaten Schlüssels

Je nach Anwendungsfall können Sie entweder ein RSA- oder ein EC-Schlüsselpaar generieren. Führen Sie eine der folgenden Aktionen aus:

- So erstellen Sie einen privaten RSA-Schlüssel in einem HSM

Verwenden Sie den [key generate-asymmetric-pair rsa](#)-Befehl, um ein RSA-Schlüsselpaar zu erzeugen. In diesem Beispiel wird ein RSA-Schlüsselpaar mit einem Modul von 2048, einem öffentlichen Exponenten von 65537, der Bezeichnung des öffentlichen Schlüssels von und der Bezeichnung des privaten Schlüssels von *tls_rsa_pub* generiert. *tls_rsa_private*

```
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
    }
}
}
}

```

- So erstellen Sie einen privaten EC-Schlüssel in einem HSM

Verwenden Sie den [key generate-asymmetric-pair ec](#)-Befehl, um ein EC-Schlüsselpaar zu generieren. In diesem Beispiel wird ein EC-Schlüsselpaar mit der prime256v1 Kurve (die der NID_X9_62_prime256v1 Kurve entspricht), einem öffentlichen Schlüssellabel von *tls_ec_pub* und einem privaten Schlüssellabel von *tls_ec_private* generiert.

```

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": "0x0000000000012000b",
      "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": "0x0000000000012000c",
  "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"
  }
}
}
}

```

Exportieren einer gefälschten privaten PEM-Schlüsseldatei

Sobald Sie einen privaten Schlüssel auf dem HSM haben, müssen Sie eine gefälschte private PEM-Schlüsseldatei exportieren. Diese Datei enthält nicht die eigentlichen Schlüsseldaten, ermöglicht es der OpenSSL Dynamic Engine jedoch, den privaten Schlüssel auf dem HSM zu identifizieren. Sie können dann den privaten Schlüssel verwenden, um eine Zertifikatsignierungsanforderung (CSR) zu erstellen und die CSR zu signieren, um das Zertifikat zu erstellen.

Verwenden Sie den [key generate-file](#) Befehl, um den privaten Schlüssel im falschen PEM-Format zu exportieren und in einer Datei zu speichern. Ersetzen Sie die folgenden Werte durch Ihre eigenen.

- *<private_key_label>*— Bezeichnung des privaten Schlüssels, den Sie im vorherigen Schritt generiert haben.
- *<web_server_fake_pem.key>*— Name der Datei, in die Ihr gefälschter PEM-Schlüssel geschrieben wird.

```
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"
  }
}
```

Beenden Sie die CloudHSM-CLI

Führen Sie den folgenden Befehl aus, um die CloudHSM-CLI zu beenden.

```
aws-cloudhsm > quit
```

Sie sollten jetzt eine neue Datei auf Ihrem System haben, die sich unter dem `<web_server_fake_pem.key>` im vorherigen Befehl angegebenen Pfad befindet. Diese Datei ist die gefälschte private PEM-Schlüsseldatei.

Generieren eines selbstsignierten Zertifikats

Sobald Sie einen gefälschten privaten PEM-Schlüssel generiert haben, können Sie diese Datei verwenden, um eine Zertifikatssignieranforderung (CSR) und ein Zertifikat zu generieren.

In einer Produktionsumgebung verwenden Sie in der Regel eine Zertifikatsstelle (CA) zum Erstellen eines Zertifikats aus einer CSR. Für eine Testumgebung ist keine CA erforderlich. Wenn Sie eine Zertifizierungsstelle verwenden, senden Sie ihnen die CSR-Datei und verwenden Sie ein signiertes SSL/TLS Zertifikat, das sie Ihnen auf Ihrem Webserver für HTTPS zur Verfügung stellen.

Als Alternative zur Verwendung einer CA können Sie die AWS CloudHSM OpenSSL Dynamic Engine verwenden, um ein selbstsigniertes Zertifikat zu erstellen. Selbstsignierte Zertifikate sind nicht vertrauenswürdig für Browser und sollten in Produktionsumgebungen nicht verwendet werden. Sie können in Testumgebungen verwendet werden.

Warning

Selbstsignierte Zertifikate sollten nur in einer Testumgebung verwendet werden. Für eine Produktionsumgebung, verwenden Sie eine sicherere Methode, wie z. B. eine Zertifikatsstelle, um ein Zertifikat zu erstellen.

Installieren und konfigurieren Sie die OpenSSL Dynamic Engine

1. Stellen Sie eine Verbindung mit Ihrer Client-Instance her.
2. [the section called “Installieren”](#)

Generieren eines Zertifikats

1. Besorgen Sie sich eine Kopie Ihrer gefälschten PEM-Datei, die in einem früheren Schritt generiert wurde.
2. Erstellen einer CSR

Führen Sie den folgenden Befehl aus, um mit der AWS CloudHSM OpenSSL Dynamic Engine eine Certificate Signing Request (CSR) zu erstellen. `<web_server_fake_pem.key>` Ersetzen Sie ihn durch den Namen der Datei, die Ihren gefälschten privaten PEM-Schlüssel enthält.

`<web_server.csr>` Ersetzen Sie es durch den Namen der Datei, die Ihre CSR enthält.

Der Befehl `req` ist interaktiv. Füllen Sie jedes Feld aus. Die Feldinformationen werden in Ihr SSL/TLS Zertifikat kopiert.

```
$ openssl req -engine cloudhsm -new -key <web_server_fake_pem.key> -  
out <web_server.csr>
```

3. Erstellen eines selbstsignierten Zertifikats

Führen Sie den folgenden Befehl aus, um die AWS CloudHSM OpenSSL Dynamic Engine zu verwenden, um Ihre CSR mit Ihrem privaten Schlüssel auf Ihrem HSM zu signieren. Dadurch wird ein selbstsigniertes Zertifikat erstellt. Ersetzen Sie die folgenden Werte in dem Befehl durch Ihre eigenen.

- `<web_server.csr>`— Name der Datei, die die CSR enthält.
- `<web_server_fake_pem.key>`— Name der Datei, die den gefälschten privaten PEM-Schlüssel enthält.
- `<web_server.crt>`— Name der Datei, die Ihr Webserver-Zertifikat enthalten wird.

```
$ openssl x509 -engine cloudhsm -req -days 365 -in <web_server.csr> -  
signkey <web_server_fake_pem.key> -out <web_server.crt>
```

Nachdem Sie diese Schritte abgeschlossen haben, fahren Sie mit [Schritt 3: Konfigurieren des Webservers](#) fort.

Schritt 3: Konfigurieren des Webservers

Aktualisieren Sie die Konfiguration Ihrer Webserver-Software, um das HTTPS-Zertifikat und den zugehörigen gefälschten privaten PEM-Schlüssel zu verwenden, die Sie im [vorherigen Schritt](#) erstellt haben. Denken Sie daran, Ihre vorhandenen Zertifikate und Schlüssel zu sichern, bevor Sie beginnen. Damit ist die Einrichtung Ihrer Linux-Webserver-Software für das SSL/TLS Offload mit abgeschlossen. AWS CloudHSM

Führen Sie die Schritte aus einem der folgenden Abschnitte aus.

Themen

- [Konfigurieren eines NGINX-Webservers](#)
- [Konfigurieren des Apache-Webservers](#)

Konfigurieren eines NGINX-Webservers

Verwenden Sie diesen Abschnitt, um NGINX auf unterstützten Plattformen zu konfigurieren.

So aktualisieren Sie die Webserverkonfiguration für NGINX

1. Stellen Sie eine Verbindung mit Ihrer Client-Instance her.
2. Führen Sie den folgenden Befehl aus, um die erforderlichen Verzeichnisse für das Webserverzertifikat und den gefälschten privaten PEM-Schlüssel zu erstellen.

```
$ sudo mkdir -p /etc/pki/nginx/private
```

3. Führen Sie den folgenden Befehl aus, um Ihr Webserverzertifikat an die erforderliche Stelle zu kopieren. `<web_server.crt>` Ersetzen Sie es durch den Namen Ihres Webserver-Zertifikats.

```
$ sudo cp <web_server.crt> /etc/pki/nginx/server.crt
```

4. Führen Sie den folgenden Befehl aus, um den gefälschten privaten PEM-Schlüssel an die erforderliche Stelle zu kopieren. `<web_server_fake_pem.key>` Ersetzen Sie es durch den Namen der Datei, die Ihren gefälschten privaten PEM-Schlüssel enthält.

```
$ sudo cp <web_server_fake_pem.key> /etc/pki/nginx/private/server.key
```

5. Führen Sie den folgenden Befehl aus, um die Eigentümerschaft für diese Dateien zu ändern, sodass der Benutzer mit dem Namen `nginx` diese lesen kann.

```
$ sudo chown nginx /etc/pki/nginx/server.crt /etc/pki/nginx/private/server.key
```

6. Führen Sie den folgenden Befehl aus, um die Datei `/etc/nginx/nginx.conf` zu sichern.

```
$ sudo cp /etc/nginx/nginx.conf /etc/nginx/nginx.conf.backup
```

7. Aktualisieren Sie die NGINX-Konfiguration.

Note

Jeder Cluster kann maximal 1000 NGINX-Worker-Prozesse auf allen NGINX-Webservern unterstützen.

Amazon Linux 2

Verwenden Sie einen Texteditor, um die Datei `/etc/nginx/nginx.conf` zu bearbeiten. Dies erfordert Linux-Root-Berechtigungen. Fügen Sie am Anfang der Datei die folgende Zeilen hinzu:

```
ssl_engine cloudhsm;  
env CLOUDHSM_PIN;
```

Fügen Sie dann dem Abschnitt TLS der Datei Folgendes hinzu:

```
# 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

Verwenden Sie einen Texteditor, um die Datei `/etc/nginx/nginx.conf` zu bearbeiten. Dies erfordert Linux-Root-Berechtigungen. Fügen Sie am Anfang der Datei die folgende Zeilen hinzu:

```

ssl_engine cloudhsm;
env CLOUDHSM_PIN;

```

Fügen Sie dann dem Abschnitt TLS der Datei Folgendes hinzu:

```

# 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

Verwenden Sie einen Texteditor, um die Datei `/etc/nginx/nginx.conf` zu bearbeiten. Dies erfordert Linux-Root-Berechtigungen. Fügen Sie am Anfang der Datei die folgende Zeilen hinzu:

```
ssl_engine cloudhsm;
```

```
env CLOUDHSM_PIN;
```

Fügen Sie dann dem Abschnitt TLS der Datei Folgendes hinzu:

```
# 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

Verwenden Sie einen Texteditor, um die Datei `/etc/nginx/nginx.conf` zu bearbeiten. Dies erfordert Linux-Root-Berechtigungen. Fügen Sie am Anfang der Datei die folgende Zeilen hinzu:

```
ssl_engine cloudhsm;  
env CLOUDHSM_PIN;
```

Fügen Sie dann dem Abschnitt TLS der Datei Folgendes hinzu:

```
# 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

Verwenden Sie einen Texteditor, um die Datei `/etc/nginx/nginx.conf` zu bearbeiten. Dies erfordert Linux-Root-Berechtigungen. Fügen Sie am Anfang der Datei die folgende Zeilen hinzu:

```

ssl_engine cloudhsm;
env CLOUDHSM_PIN;

```

Fügen Sie dann dem Abschnitt TLS der Datei Folgendes hinzu:

```

# 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

Verwenden Sie einen Texteditor, um die Datei `/etc/nginx/nginx.conf` zu bearbeiten. Dies erfordert Linux-Root-Berechtigungen. Fügen Sie am Anfang der Datei die folgende Zeilen hinzu:

```
ssl_engine cloudhsm;
env CLOUDHSM_PIN;
```

Fügen Sie dann dem Abschnitt TLS der Datei Folgendes hinzu:

```
# 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 {

}
}

```

Speichern Sie die Datei.

8. Sichern Sie die systemd-Konfigurationsdatei und legen Sie dann den EnvironmentFile-Pfad fest.

Amazon Linux 2

1. Sichern Sie die Datei `nginx.service`.

```
$ sudo cp /lib/systemd/system/nginx.service /lib/systemd/system/
nginx.service.backup
```

2. Öffnen Sie die Datei `/lib/systemd/system/nginx.service` in einem Texteditor. Fügen Sie dann im Abschnitt `[Service]` den folgenden Pfad hinzu:

```
EnvironmentFile=/etc/sysconfig/nginx
```

Amazon Linux 2023

1. Sichern Sie die Datei `nginx.service`.

```
$ sudo cp /lib/systemd/system/nginx.service /lib/systemd/system/  
nginx.service.backup
```

2. Öffnen Sie die `/lib/systemd/system/nginx.service` Datei in einem Texteditor und fügen Sie dann im Abschnitt `[Service]` den folgenden Pfad hinzu:

```
EnvironmentFile=/etc/sysconfig/nginx
```

Red Hat 8

1. Sichern Sie die Datei `nginx.service`.

```
$ sudo cp /lib/systemd/system/nginx.service /lib/systemd/system/  
nginx.service.backup
```

2. Öffnen Sie die Datei `/lib/systemd/system/nginx.service` in einem Texteditor. Fügen Sie dann im Abschnitt `[Service]` den folgenden Pfad hinzu:

```
EnvironmentFile=/etc/sysconfig/nginx
```

Red Hat 9

1. Sichern Sie die Datei `nginx.service`.

```
$ sudo cp /lib/systemd/system/nginx.service /lib/systemd/system/  
nginx.service.backup
```

2. Öffnen Sie die Datei `/lib/systemd/system/nginx.service` in einem Texteditor. Fügen Sie dann im Abschnitt `[Service]` den folgenden Pfad hinzu:

```
EnvironmentFile=/etc/sysconfig/nginx
```

Ubuntu 22.04 LTS

1. Sichern Sie die Datei `nginx.service`.

```
$ sudo cp /lib/systemd/system/nginx.service /lib/systemd/system/  
nginx.service.backup
```

2. Öffnen Sie die Datei `/lib/systemd/system/nginx.service` in einem Texteditor.
Fügen Sie dann im Abschnitt `[Service]` den folgenden Pfad hinzu:

```
EnvironmentFile=/etc/sysconfig/nginx
```

Ubuntu 24.04 LTS

1. Sichern Sie die Datei `nginx.service`.

```
$ sudo cp /lib/systemd/system/nginx.service /lib/systemd/system/  
nginx.service.backup
```

2. Öffnen Sie die Datei `/lib/systemd/system/nginx.service` in einem Texteditor.
Fügen Sie dann im Abschnitt `[Service]` den folgenden Pfad hinzu:

```
EnvironmentFile=/etc/sysconfig/nginx
```

9. Überprüfen Sie, ob die Datei `/etc/sysconfig/nginx` vorhanden ist, und führen Sie dann einen der folgenden Schritte aus:

- Wenn die Datei vorhanden ist, sichern Sie die Datei, indem Sie den folgenden Befehl ausführen:

```
$ sudo cp /etc/sysconfig/nginx /etc/sysconfig/nginx.backup
```

- Öffnen Sie einen Texteditor und erstellen Sie im Ordner `/etc/sysconfig/` eine Datei mit dem Namen `nginx`, wenn die Datei nicht vorhanden ist.

10. Konfigurieren Sie die NGINX-Umgebung.

 Note

Das Client-SDK 5 führt die CLOUDHSM_PIN-Umgebungsvariable zum Speichern der Anmeldeinformationen der CU ein.

Amazon Linux 2

Öffnen Sie die Datei `/etc/sysconfig/nginx` in einem Text-Editor. Dies erfordert Linux-Root-Berechtigungen. Fügen Sie die Anmeldeinformationen für den Crypto-Benutzer (CU) hinzu:

```
CLOUDHSM_PIN=<CU user name>:<password>
```

Ersetzen Sie *<CU user name>* und *<password>* durch die CU-Anmeldeinformationen.

Speichern Sie die Datei.

Amazon Linux 2023

Öffnen Sie die Datei `/etc/sysconfig/nginx` in einem Text-Editor. Dies erfordert Linux-Root-Berechtigungen. Fügen Sie die Anmeldeinformationen für den Crypto-Benutzer (CU) hinzu:

```
CLOUDHSM_PIN=<CU user name>:<password>
```

Ersetzen Sie *<CU user name>* und *<password>* durch die CU-Anmeldeinformationen.

Speichern Sie die Datei.

Red Hat 8

Öffnen Sie die Datei `/etc/sysconfig/nginx` in einem Text-Editor. Dies erfordert Linux-Root-Berechtigungen. Fügen Sie die Anmeldeinformationen für den Crypto-Benutzer (CU) hinzu:

```
CLOUDHSM_PIN=<CU user name>:<password>
```

Ersetzen Sie *<CU user name>* und *<password>* durch die CU-Anmeldeinformationen.

Speichern Sie die Datei.

Red Hat 9

Öffnen Sie die Datei `/etc/sysconfig/nginx` in einem Text-Editor. Dies erfordert Linux-Root-Berechtigungen. Fügen Sie die Anmeldeinformationen für den Crypto-Benutzer (CU) hinzu:

```
CLLOUDHSM_PIN=<CU user name>:<password>
```

Ersetzen Sie `<CU user name>` und `<password>` durch die CU-Anmeldeinformationen.

Speichern Sie die Datei.

Ubuntu 22.04 LTS

Öffnen Sie die Datei `/etc/sysconfig/nginx` in einem Text-Editor. Dies erfordert Linux-Root-Berechtigungen. Fügen Sie die Anmeldeinformationen für den Crypto-Benutzer (CU) hinzu:

```
CLLOUDHSM_PIN=<CU user name>:<password>
```

Ersetzen Sie `<CU user name>` und `<password>` durch die CU-Anmeldeinformationen.

Speichern Sie die Datei.

Ubuntu 24.04 LTS

Öffnen Sie die Datei `/etc/sysconfig/nginx` in einem Text-Editor. Dies erfordert Linux-Root-Berechtigungen. Fügen Sie die Anmeldeinformationen für den Crypto-Benutzer (CU) hinzu:

```
CLLOUDHSM_PIN=<CU user name>:<password>
```

Ersetzen Sie `<CU user name>` und `<password>` durch die CU-Anmeldeinformationen.

Speichern Sie die Datei.

11. Starten Sie den NGINX-Webserver.

Amazon Linux 2

Stoppen Sie alle laufenden NGINX-Prozesse

```
$ sudo systemctl stop nginx
```

Laden Sie die systemd-Konfiguration neu, um die neuesten Änderungen zu erhalten

```
$ sudo systemctl daemon-reload
```

Starten Sie den NGINX-Prozess

```
$ sudo systemctl start nginx
```

Amazon Linux 2023

Stoppen Sie alle NGINX-Prozesse

```
$ sudo systemctl stop nginx
```

Laden Sie die systemd-Konfiguration neu, um die neuesten Änderungen zu erhalten

```
$ sudo systemctl daemon-reload
```

Starten Sie NGINX

```
$ sudo systemctl start nginx
```

Red Hat 8

Stoppen Sie alle laufenden NGINX-Prozesse

```
$ sudo systemctl stop nginx
```

Laden Sie die systemd-Konfiguration neu, um die neuesten Änderungen zu erhalten

```
$ sudo systemctl daemon-reload
```

Starten Sie den NGINX-Prozess

```
$ sudo systemctl start nginx
```

Red Hat 9

Stoppen Sie alle laufenden NGINX-Prozesse

```
$ sudo systemctl stop nginx
```

Laden Sie die systemd-Konfiguration neu, um die neuesten Änderungen zu erhalten

```
$ sudo systemctl daemon-reload
```

Starten Sie den NGINX-Prozess

```
$ sudo systemctl start nginx
```

Ubuntu 22.04 LTS

Stoppen Sie alle laufenden NGINX-Prozesse

```
$ sudo systemctl stop nginx
```

Laden Sie die systemd-Konfiguration neu, um die neuesten Änderungen zu erhalten

```
$ sudo systemctl daemon-reload
```

Starten Sie den NGINX-Prozess

```
$ sudo systemctl start nginx
```

Ubuntu 24.04 LTS

Stoppen Sie alle laufenden NGINX-Prozesse

```
$ sudo systemctl stop nginx
```

Laden Sie die systemd-Konfiguration neu, um die neuesten Änderungen zu erhalten

```
$ sudo systemctl daemon-reload
```

Starten Sie den NGINX-Prozess

```
$ sudo systemctl start nginx
```

12. (Optional) Konfigurieren Sie Ihre Plattform so, dass NGINX beim Start gestartet wird.

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
```

Nachdem Sie Ihre Webserverkonfiguration aktualisiert haben, gehen Sie zu [Schritt 4: Aktivieren von HTTPS-Datenverkehr und Verifizieren des Zertifikats](#).

Konfigurieren des Apache-Webservers

Verwenden Sie diesen Abschnitt, um Apache auf unterstützten Plattformen zu konfigurieren.

So aktualisieren Sie die Webserverkonfiguration für Apache

1. Stellen Sie eine Verbindung zu Ihrer Amazon-EC2-Client-Instance her.
2. Definieren Sie Standardspeicherorte für Zertifikate und private Schlüssel für Ihre Plattform.

Amazon Linux 2

Stellen Sie sicher, dass in der `/etc/httpd/conf.d/ssl.conf`-Datei die folgenden Werte vorhanden sind:

```
SSLCertificateFile      /etc/pki/tls/certs/localhost.crt
SSLCertificateKeyFile   /etc/pki/tls/private/localhost.key
```

Amazon Linux 2023

In der `/etc/httpd/conf.d/ssl.conf` Datei. Stellen Sie sicher, dass diese Werte existieren:

```
SSLCertificateFile      /etc/pki/tls/certs/localhost.crt
SSLCertificateKeyFile   /etc/pki/tls/private/localhost.key
```

Red Hat 8

Stellen Sie sicher, dass in der `/etc/httpd/conf.d/ssl.conf`-Datei die folgenden Werte vorhanden sind:

```
SSLCertificateFile      /etc/pki/tls/certs/localhost.crt
SSLCertificateKeyFile   /etc/pki/tls/private/localhost.key
```

Red Hat 9

Stellen Sie sicher, dass in der `/etc/httpd/conf.d/ssl.conf`-Datei die folgenden Werte vorhanden sind:

```
SSLCertificateFile      /etc/pki/tls/certs/localhost.crt
SSLCertificateKeyFile   /etc/pki/tls/private/localhost.key
```

Ubuntu 22.04 LTS

Stellen Sie sicher, dass in der `/etc/apache2/sites-available/default-ssl.conf`-Datei die folgenden Werte vorhanden sind:

```
SSLCertificateFile      /etc/ssl/certs/localhost.crt
SSLCertificateKeyFile   /etc/ssl/private/localhost.key
```

Ubuntu 24.04 LTS

Stellen Sie sicher, dass in der `/etc/apache2/sites-available/default-ssl.conf`-Datei die folgenden Werte vorhanden sind:

```
SSLCertificateFile      /etc/ssl/certs/localhost.crt
SSLCertificateKeyFile   /etc/ssl/private/localhost.key
```

3. Kopieren Sie Ihr Webserver-Zertifikat an den für Ihre Plattform erforderlichen Speicherort.

Amazon Linux 2

```
$ sudo cp <web_server.crt> /etc/pki/tls/certs/localhost.crt
```

`<web_server.crt>` Ersetzen Sie es durch den Namen Ihres Webserver-Zertifikats.

Amazon Linux 2023

```
$ sudo cp <web_server.crt> /etc/pki/tls/certs/localhost.crt
```

`<web_server.crt>` Ersetzen Sie es durch den Namen Ihres Webserver-Zertifikats.

Red Hat 8

```
$ sudo cp <web_server.crt> /etc/pki/tls/certs/localhost.crt
```

`<web_server.crt>` Ersetzen Sie es durch den Namen Ihres Webserver-Zertifikats.

Red Hat 9

```
$ sudo cp <web_server.crt> /etc/pki/tls/certs/localhost.crt
```

<web_server.crt> Ersetzen Sie es durch den Namen Ihres Webserver-Zertifikats.

Ubuntu 22.04 LTS

```
$ sudo cp <web_server.crt> /etc/ssl/certs/localhost.crt
```

<web_server.crt> Ersetzen Sie es durch den Namen Ihres Webserver-Zertifikats.

Ubuntu 24.04 LTS

```
$ sudo cp <web_server.crt> /etc/ssl/certs/localhost.crt
```

<web_server.crt> Ersetzen Sie es durch den Namen Ihres Webserver-Zertifikats.

4. Kopieren Sie Ihren gefälschten privaten PEM-Schlüssel an den für Ihre Plattform erforderlichen Speicherort.

Amazon Linux 2

```
$ sudo cp <web_server_fake_pem.key> /etc/pki/tls/private/localhost.key
```

<web_server_fake_pem.key> Ersetzen Sie es durch den Namen der Datei, die Ihren gefälschten privaten PEM-Schlüssel enthält.

Amazon Linux 2023

```
$ sudo cp <web_server_fake_pem.key> /etc/pki/tls/private/localhost.key
```

<web_server_fake_pem.key> Ersetzen Sie es durch den Namen der Datei, die Ihren gefälschten privaten PEM-Schlüssel enthält.

Red Hat 8

```
$ sudo cp <web_server_fake_pem.key> /etc/pki/tls/private/localhost.key
```

<web_server_fake_pem.key> Ersetzen Sie es durch den Namen der Datei, die Ihren gefälschten privaten PEM-Schlüssel enthält.

Red Hat 9

```
$ sudo cp <web_server_fake_pem.key> /etc/pki/tls/private/localhost.key
```

`<web_server_fake_pem.key>` Ersetzen Sie es durch den Namen der Datei, die Ihren gefälschten privaten PEM-Schlüssel enthält.

Ubuntu 22.04 LTS

```
$ sudo cp <web_server_fake_pem.key> /etc/ssl/private/localhost.key
```

`<web_server_fake_pem.key>` Ersetzen Sie es durch den Namen der Datei, die Ihren gefälschten privaten PEM-Schlüssel enthält.

Ubuntu 24.04 LTS

```
$ sudo cp <web_server_fake_pem.key> /etc/ssl/private/localhost.key
```

`<web_server_fake_pem.key>` Ersetzen Sie es durch den Namen der Datei, die Ihren gefälschten privaten PEM-Schlüssel enthält.

5. Ändern Sie den Besitz dieser Dateien, falls Ihre Plattform dies erfordert.

Amazon Linux 2

```
$ sudo chown apache /etc/pki/tls/certs/localhost.crt /etc/pki/tls/private/localhost.key
```

Gewährt dem Benutzer mit dem Namen Apache Leseberechtigung.

Amazon Linux 2023

```
$ sudo chown apache /etc/pki/tls/certs/localhost.crt /etc/pki/tls/private/localhost.key
```

Gewährt dem Benutzer mit dem Namen Apache Leseberechtigung.

Red Hat 8

```
$ sudo chown apache /etc/pki/tls/certs/localhost.crt /etc/pki/tls/private/localhost.key
```

Gewährt dem Benutzer mit dem Namen Apache Leseberechtigung.

Red Hat 9

```
$ sudo chown apache /etc/pki/tls/certs/localhost.crt /etc/pki/tls/private/localhost.key
```

Gewährt dem Benutzer mit dem Namen Apache Leseberechtigung.

Ubuntu 22.04 LTS

Es ist keine Aktion erforderlich.

Ubuntu 24.04 LTS

Es ist keine Aktion erforderlich.

6. Konfigurieren Sie Apache-Direktiven für Ihre Plattform.

Amazon Linux 2

Suchen Sie die SSL-Datei für diese Plattform:

```
/etc/httpd/conf.d/ssl.conf
```

Diese Datei enthält Apache-Direktiven, die definieren, wie Ihr Server laufen soll. Direktiven erscheinen auf der linken Seite, gefolgt von einem Wert. Verwenden Sie einen Texteditor, um diese Datei zu bearbeiten. Dies erfordert Linux-Root-Berechtigungen.

Aktualisieren Sie die folgenden Direktiven oder geben Sie sie mit diesen Werten ein:

```
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
```

Speichern Sie die Datei.

Amazon Linux 2023

Suchen Sie die SSL-Datei für diese Plattform:


```
/etc/httpd/conf.d/ssl.conf
```

Diese Datei enthält Apache-Direktiven, die definieren, wie Ihr Server laufen soll. Direktiven erscheinen auf der linken Seite, gefolgt von einem Wert. Verwenden Sie einen Texteditor, um diese Datei zu bearbeiten. Dies erfordert Linux-Root-Berechtigungen.

Aktualisieren Sie die folgenden Direktiven oder geben Sie sie mit diesen Werten ein:

```
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
```

Speichern Sie die Datei.

Red Hat 8

Suchen Sie die SSL-Datei für diese Plattform:

```
/etc/httpd/conf.d/ssl.conf
```

Diese Datei enthält Apache-Direktiven, die definieren, wie Ihr Server laufen soll. Direktiven erscheinen auf der linken Seite, gefolgt von einem Wert. Verwenden Sie einen Texteditor, um diese Datei zu bearbeiten. Dies erfordert Linux-Root-Berechtigungen.

Aktualisieren Sie die folgenden Direktiven oder geben Sie sie mit diesen Werten ein:

```
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
```

Speichern Sie die Datei.

Red Hat 9

Suchen Sie die SSL-Datei für diese Plattform:

```
/etc/httpd/conf.d/ssl.conf
```

Diese Datei enthält Apache-Direktiven, die definieren, wie Ihr Server laufen soll. Direktiven erscheinen auf der linken Seite, gefolgt von einem Wert. Verwenden Sie einen Texteditor, um diese Datei zu bearbeiten. Dies erfordert Linux-Root-Berechtigungen.

Aktualisieren Sie die folgenden Direktiven oder geben Sie sie mit diesen Werten ein:

```
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
```

Speichern Sie die Datei.

Ubuntu 22.04 LTS

Suchen Sie die SSL-Datei für diese Plattform:

```
/etc/apache2/mods-available/ssl.conf
```

Diese Datei enthält Apache-Direktiven, die definieren, wie Ihr Server laufen soll. Direktiven erscheinen auf der linken Seite, gefolgt von einem Wert. Verwenden Sie einen Texteditor, um diese Datei zu bearbeiten. Dies erfordert Linux-Root-Berechtigungen.

Aktualisieren Sie die folgenden Direktiven oder geben Sie sie mit diesen Werten ein:

```
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
```

Speichern Sie die Datei.

Aktivieren Sie das SSL-Modul und die standardmäßige SSL-Site-Konfiguration:

```
$ sudo a2enmod ssl
$ sudo a2ensite default-ssl
```

Ubuntu 24.04 LTS

Suchen Sie die SSL-Datei für diese Plattform:

```
/etc/apache2/mods-available/ssl.conf
```

Diese Datei enthält Apache-Direktiven, die definieren, wie Ihr Server laufen soll. Direktiven erscheinen auf der linken Seite, gefolgt von einem Wert. Verwenden Sie einen Texteditor, um diese Datei zu bearbeiten. Dies erfordert Linux-Root-Berechtigungen.

Aktualisieren Sie die folgenden Direktiven oder geben Sie sie mit diesen Werten ein:

```
SSLCryptoDevice ccloudhsm
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
```

Speichern Sie die Datei.

Aktivieren Sie das SSL-Modul und die standardmäßige SSL-Site-Konfiguration:

```
$ sudo a2enmod ssl
$ sudo a2ensite default-ssl
```

7. Konfigurieren Sie eine Datei mit Umgebungswerten für Ihre Plattform.

Amazon Linux 2

Öffnen Sie die HTTPD-Servicedatei:

```
/lib/systemd/system/httpd.service
```

Fügen Sie unter dem Abschnitt [Service] Folgendes hinzu:

```
EnvironmentFile=/etc/sysconfig/httpd
```

Amazon Linux 2023

Öffnen Sie die HTTPD-Servicedatei:

```
/lib/systemd/system/httpd.service
```

Fügen Sie unter dem Abschnitt [Service] Folgendes hinzu:

```
EnvironmentFile=/etc/sysconfig/httpd
```

Red Hat 8

Öffnen Sie die HTTPD-Servicedatei:

```
/lib/systemd/system/httpd.service
```

Fügen Sie unter dem Abschnitt [Service] Folgendes hinzu:

```
EnvironmentFile=/etc/sysconfig/httpd
```

Red Hat 9

Öffnen Sie die HTTPD-Servicedatei:

```
/lib/systemd/system/httpd.service
```

```
EnvironmentFile=/etc/sysconfig/httpd
```

Ubuntu 22.04 LTS

Es ist keine Aktion erforderlich. Umgebungswerte werden gehen zu `/etc/sysconfig/httpd`

Ubuntu 24.04 LTS

Es ist keine Aktion erforderlich. Umgebungswerte werden gehen zu `/etc/sysconfig/httpd`

8. Legen Sie in der Datei, in der Umgebungsvariablen für Ihre Plattform gespeichert sind, eine Umgebungsvariable fest, die die Anmeldeinformationen für den Crypto-Benutzer (CU) enthält:

Amazon Linux 2

Verwenden Sie einen Texteditor, um `/etc/sysconfig/httpd` zu bearbeiten.

```
CLOUDHSM_PIN=<CU user name>:<password>
```

Ersetzen Sie `<CU user name>` und `<password>` durch die CU-Anmeldeinformationen.

Amazon Linux 2023

Verwenden Sie einen Texteditor, um `/etc/sysconfig/httpd` zu bearbeiten.

```
CLOUDHSM_PIN=<CU user name>:<password>
```

Ersetzen Sie `<CU user name>` und `<password>` durch die CU-Anmeldeinformationen.

Red Hat 8

Verwenden Sie einen Texteditor, um `/etc/sysconfig/httpd` zu bearbeiten.

```
CLOUDHSM_PIN=<CU user name>:<password>
```

Ersetzen Sie `<CU user name>` und `<password>` durch die CU-Anmeldeinformationen.

 Note

Das Client-SDK 5 führt die CLOUDHSM_PIN-Umgebungsvariable zum Speichern der Anmeldeinformationen der CU ein.

Red Hat 9

Verwenden Sie einen Texteditor, um `/etc/sysconfig/httpd` zu bearbeiten.

```
CLOUDHSM_PIN=<CU user name>:<password>
```

Ersetzen Sie `<CU user name>` und `<password>` durch die CU-Anmeldeinformationen.

 Note

Das Client-SDK 5 führt die CLOUDHSM_PIN-Umgebungsvariable zum Speichern der Anmeldeinformationen der CU ein.

Ubuntu 22.04 LTS

Verwenden Sie einen Texteditor, um `/etc/apache2/envvars` zu bearbeiten.

```
export CLOUDHSM_PIN=<CU user name>:<password>
```

Ersetzen Sie `<CU user name>` und `<password>` durch die CU-Anmeldeinformationen.

 Note

Das Client-SDK 5 führt die CLOUDHSM_PIN-Umgebungsvariable zum Speichern der Anmeldeinformationen der CU ein. Im Client-SDK 3 speichern Sie die CU-Anmeldeinformationen in der `n3fips_password`-Umgebungsvariable. Client-SDK 5 unterstützt beide Umgebungsvariablen, wir empfehlen jedoch, CLOUDHSM_PIN zu verwenden.

Ubuntu 24.04 LTS

Verwenden Sie einen Texteditor, um `/etc/apache2/envvars` zu bearbeiten.

```
export CLOUDHSM_PIN=<CU user name>:<password>
```

Ersetzen Sie `<CU user name>` und `<password>` durch die CU-Anmeldeinformationen.

Note

Das Client-SDK 5 führt die CLOUDHSM_PIN-Umgebungsvariable zum Speichern der Anmeldeinformationen der CU ein. Im Client-SDK 3 speichern Sie die CU-Anmeldeinformationen in der `n3fips_password`-Umgebungsvariable. Client-SDK 5 unterstützt beide Umgebungsvariablen, wir empfehlen jedoch, CLOUDHSM_PIN zu verwenden.

9. Starten Sie den Apache-Webserver.

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
```

10. (Optional) Konfigurieren Sie Ihre Plattform so, dass Apache beim Start gestartet wird.

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
```

Nachdem Sie Ihre Webserverkonfiguration aktualisiert haben, gehen Sie zu [Schritt 4: Aktivieren von HTTPS-Datenverkehr und Verifizieren des Zertifikats](#).

Schritt 4: Aktivieren von HTTPS-Datenverkehr und Verifizieren des Zertifikats

Nachdem Sie Ihren Webserver für SSL/TLS Offload mit konfiguriert haben AWS CloudHSM, fügen Sie Ihre Webserver-Instanz einer Sicherheitsgruppe hinzu, die eingehenden HTTPS-Verkehr zulässt. Dadurch können Clients, wie z. B. Webbrowser, eine HTTPS-Verbindung mit Ihrem Webserver herstellen. Stellen Sie dann eine HTTPS-Verbindung zu Ihrem Webserver her und stellen Sie sicher, dass er das Zertifikat verwendet, mit dem Sie das SSL/TLS Offload konfiguriert haben. AWS CloudHSM

Themen

- [Aktivieren von eingehenden HTTPS-Verbindungen](#)
- [Verifizieren, dass HTTPS das konfigurierte Zertifikat verwendet](#)

Aktivieren von eingehenden HTTPS-Verbindungen

Zum Herstellen einer Verbindung zu Ihrem Webserver von einem Client (z. B. ein Webbrowser) aus, erstellen Sie eine Sicherheitsgruppe, die eingehende HTTPS-Verbindungen zulässt. Insbesondere sollten eingehende TCP-Verbindungen auf Port 443 erlaubt werden. Weisen Sie diese Sicherheitsgruppe Ihrem Webserver zu.

So erstellen Sie eine Sicherheitsgruppe für HTTPS und weisen sie Ihrem Webserver zu

1. Öffnen Sie die Amazon-EC2-Konsole unter <https://console.aws.amazon.com/ec2/>.
2. Wählen Sie im Navigationsbereich Sicherheitsgruppen aus.
3. Wählen Sie Sicherheitsgruppe erstellen.
4. Führen Sie für Sicherheitsgruppe erstellen die folgenden Schritte aus:
 - a. Geben Sie in das Feld Sicherheitsgruppenname einen Namen für die Sicherheitsgruppe ein, die Sie erstellen.
 - b. (Optional) Geben Sie eine Beschreibung der Sicherheitsgruppe ein, die Sie erstellen.
 - c. Wählen Sie für VPC die VPC aus, die Ihre Amazon-EC2-Instance enthält.
 - d. Wählen Sie Regel hinzufügen aus.
 - e. Wählen Sie im Drop-down-Fenster für Typ die Option HTTPS aus.
 - f. Geben Sie für Quelle einen Quellspeicherort ein.
 - g. Wählen Sie Sicherheitsgruppe erstellen aus.
5. Wählen Sie im Navigationsbereich Instances aus.

6. Aktivieren Sie das Kontrollkästchen neben Ihrer Webserver-Instance.
7. Wählen Sie das Drop-down-Menü Aktionen oben auf der Seite. Wählen Sie Sicherheit und dann Sicherheitsgruppen ändern aus.
8. Wählen Sie unter Zugeordnete Sicherheitsgruppen das Suchfeld aus und wählen Sie die Sicherheitsgruppe, die Sie für HTTPS erstellt haben, aus. Wählen Sie dann Sicherheitsgruppen hinzufügen aus.
9. Wählen Sie Speichern.

Verifizieren, dass HTTPS das konfigurierte Zertifikat verwendet

Nachdem Sie den Webserver zu einer Sicherheitsgruppe hinzugefügt haben, können Sie überprüfen, ob für den SSL/TLS Offload Ihr selbstsigniertes Zertifikat verwendet wird. Sie können dazu einen Webbrowser oder ein Tool wie [OpenSSL s_client](#) nutzen.

Um den SSL/TLS Offload mit einem Webbrowser zu überprüfen

1. Verwenden Sie einen Web-Browser, um eine Verbindung zum Webserver unter Verwendung des öffentlichen DNS-Namen oder der IP-Adresse des Servers herzustellen. Stellen Sie sicher, dass die URL in die Adresszeile mit https:// beginnt. Beispiel, **`https://ec2-52-14-212-67.us-east-2.compute.amazonaws.com/`**.

 Tip

Sie können einen DNS-Dienst wie Amazon Route 53 verwenden, um den Domainnamen Ihrer Website (z. B. `https://www.example.com/`) an Ihren Webserver weiterzuleiten. Weitere Informationen finden Sie unter [Routing des Datenverkehrs zu einer Amazon-EC2-Instance](#) im Entwicklerleitfaden zu Amazon Route 53 oder in der Dokumentation für Ihren DNS-Service.

2. Zeigen Sie das Webserverzertifikat mit Ihrem Webbrowser an. Weitere Informationen finden Sie hier:
 - Wenn Sie Mozilla Firefox nutzen, sehen Sie sich die Informationen auf der Mozilla Support-Website unter [Zertifikat anzeigen](#) an.
 - Wenn Sie Google Chrome verwenden, sehen Sie sich die Informationen auf der „Google Tools für Web Developers“-Website unter [Sicherheitsprobleme verstehen](#) an.

Andere Webbrowser unterstützen möglicherweise ähnliche Funktionen, über die Sie das Webserverzertifikat anzeigen können.

3. Stellen Sie sicher, dass es sich bei dem SSL/TLS Zertifikat um das Zertifikat handelt, für dessen Verwendung Sie Ihren Webserver konfiguriert haben.

Um den SSL/TLS Offload mit OpenSSL s_client zu überprüfen

1. Führen Sie den folgenden OpenSSL-Befehl aus, um mittels HTTPS eine Verbindung zu Ihrem Webserver herzustellen. `<server name>` Ersetzen Sie es durch den öffentlichen DNS-Namen oder die IP-Adresse Ihres Webserver.

```
openssl s_client -connect <server name>:443
```

 Tip

Sie können einen DNS-Dienst wie Amazon Route 53 verwenden, um den Domainnamen Ihrer Website (z. B. `https://www.example.com/`) an Ihren Webserver weiterzuleiten. Weitere Informationen finden Sie unter [Routing des Datenverkehrs zu einer Amazon-EC2-Instance](#) im Entwicklerleitfaden zu Amazon Route 53 oder in der Dokumentation für Ihren DNS-Service.

2. Stellen Sie sicher, dass es sich bei dem SSL/TLS Zertifikat um das Zertifikat handelt, für dessen Verwendung Sie Ihren Webserver konfiguriert haben.

Sie verfügen jetzt über eine mit HTTPS gesicherte Website. Der private Schlüssel für den Webserver wird in einem HSM in Ihrem AWS CloudHSM Cluster gespeichert.

Informationen zum Hinzufügen eines Load Balancers finden Sie unter [Einen Load Balancer mit Elastic Load Balancing hinzufügen für AWS CloudHSM\(optional\)](#).

AWS CloudHSM SSL/TLS-Offload unter Linux mit NGINX oder mit OpenSSL Provider HAProxy

Dieses Thema enthält step-by-step Anweisungen zum Einrichten des SSL/TLS Server-Identity-Offloads AWS CloudHSM auf einem Linux-Webserver mit NGINX oder HAProxy mit dem OpenSSL Provider.

Themen

- [-Übersicht](#)
- [Schritt 1: Einrichten der Voraussetzungen](#)
- [Schritt 2: Generieren oder importieren Sie einen privaten Schlüssel und holen Sie sich ein Zertifikat](#)
- [Schritt 3: Konfigurieren des Webserver](#)
- [Schritt 4: Aktivieren von HTTPS-Datenverkehr und Verifizieren des Zertifikats](#)

-Übersicht

Unter Linux sind [NGINX](#) und die [HAProxy](#) Webserver-Software in [OpenSSL integriert, um HTTPS zu unterstützen](#). Der [AWS CloudHSM OpenSSL Provider](#) stellt eine Schnittstelle bereit, die es der Webserver-Software ermöglicht, die HSMs in Ihrem Cluster für kryptografisches Offloading und Schlüsselspeicherung zu verwenden. Der OpenSSL Provider ist die Brücke, die den Webserver mit Ihrem AWS CloudHSM Cluster verbindet.

Um dieses Tutorial abzuschließen, müssen Sie NGINX konfigurieren oder HAProxy den AWS CloudHSM OpenSSL Provider verwenden. Das Tutorial zeigt Ihnen, wie Sie Folgendes tun können:

- Installieren der Web-Server-Software auf einer Amazon EC2-Instance
- Konfigurieren Sie die Webserver-Software so, dass sie HTTPS mit einem privaten Schlüssel unterstützt, der in Ihrem AWS CloudHSM -Cluster gespeichert ist.
- (Optional) Verwenden Sie Amazon EC2, um eine zweite Webserver-Instance zu erstellen, und Elastic Load Balancing, um einen Load Balancer zu erstellen. Mit einem Load Balancer kann die Leistung durch Verteilung der Arbeitslast auf mehrere Server gesteigert werden. Sie kann auch für Redundanz und eine höhere Verfügbarkeit sorgen, falls ein oder mehrere Webserver ausfallen.

Wenn Sie bereit sind, sehen Sie sich [Schritt 1: Einrichten der Voraussetzungen](#) an.

Schritt 1: Einrichten der Voraussetzungen

Verschiedene Plattformen erfordern unterschiedliche Voraussetzungen. Verwenden Sie den Abschnitt mit den Voraussetzungen unten, der zu Ihrer Plattform passt.

Voraussetzungen für AWS CloudHSM OpenSSL Provider

Um das Offload von SSL/TLS Webserver-Serveridentitäten mit AWS CloudHSM OpenSSL Provider for Client SDK 5 einzurichten, benötigen Sie Folgendes:

- Ein aktiver AWS CloudHSM Cluster mit mindestens zwei Hardware-Sicherheitsmodulen (HSM)

Note

Sie können einen einzelnen HSM-Cluster verwenden, müssen aber zuerst die Haltbarkeit der Client-Schlüssel deaktivieren. Weitere Informationen finden Sie unter [Einstellungen für die Haltbarkeit von Client-Schlüsseln verwalten](#) und [Client-SDK 5 Configure Tool](#).

- Eine Amazon EC2-Instance, auf der ein Linux-Betriebssystem ausgeführt wird und die folgende Software installiert ist:
 - Ein Webserver (entweder NGINX oder) HAProxy
 - Der AWS CloudHSM OpenSSL-Anbieter für Client SDK 5
- Ein [Crypto-Benutzer](#) (CU), der den privaten Schlüssel des Webserver auf dem HSM besitzen und verwalten soll.

So richten Sie eine Linux-Webserver-Instance auf dem HSM ein und erstellen einen CU

Note

Viele der Befehle in diesem Verfahren erfordern erhöhte Rechte. Abhängig von Ihrer Systemkonfiguration müssen Sie möglicherweise Befehle mit sudo oder als Root-Benutzer ausführen.

1. Installieren und konfigurieren Sie den AWS CloudHSM OpenSSL Provider für Client SDK 5. Weitere Informationen zur Installation des OpenSSL Providers finden Sie unter [AWS CloudHSM OpenSSL Provider for Client SDK 5](#).

2. Installieren Sie auf einer EC2-Linux-Instance, die Zugriff auf Ihren Cluster hat, entweder NGINX oder einen Webserver: HAProxy

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 24.04

- NGINX

```
$ apt install nginx
```

- HAProxy

```
$ apt install haproxy
```

3. Verwenden Sie die CloudHSM-CLI, um einen [Crypto-Benutzer](#) zu erstellen. Weitere Informationen zur Verwaltung von HSM-Benutzern finden Sie unter [HSM-Benutzer mit der CloudHSM-CLI verwalten](#).

 Tip

Merken Sie sich den CU-Benutzernamen und das Passwort. Sie benötigen sie später beim Generieren oder Importieren des privaten HTTPS-Schlüssels und -Zertifikats für Ihren Webserver.

Nachdem Sie diese Schritte abgeschlossen haben, fahren Sie mit [Schritt 2: Generieren oder importieren Sie einen privaten Schlüssel und holen Sie sich ein Zertifikat](#) fort.

Hinweise

- Um Security-Enhanced Linux (SELinux) und Webserver zu verwenden, müssen Sie ausgehende TCP-Verbindungen auf Port 2223 zulassen, dem Port, den Client SDK 5 für die Kommunikation mit dem HSM verwendet.
- Um einen Cluster zu erstellen und zu aktivieren und einer EC2-Instance Zugriff auf den Cluster zu gewähren, führen Sie die Schritte unter [Erste Schritte mit AWS CloudHSM](#) aus. Die ersten Schritte bieten step-by-step Anweisungen zum Erstellen eines aktiven Clusters mit einem HSM und einer Amazon EC2 EC2-Client-Instance. Sie können diese Client-Instance als Ihren Webserver verwenden.
- Um zu vermeiden, dass die Haltbarkeit von Client-Schlüsseln deaktiviert wird, fügen Sie Ihrem Cluster mehr als ein HSM hinzu. Weitere Informationen finden Sie unter [Hinzufügen eines HSM zu einem Cluster AWS CloudHSM](#).
- Um sich mit Ihrer Client-Instance zu verbinden, können Sie SSH oder PuTTY verwenden. Weitere Informationen finden Sie unter [Herstellung einer Verbindung zu Ihrer Linux-Instance mit SSH](#) oder [Herstellung einer Verbindung zu Ihrer Linux-Instance von Windows mit PuTTY](#) in der Amazon EC2-Dokumentation.

Schritt 2: Generieren oder importieren Sie einen privaten Schlüssel und holen Sie sich ein Zertifikat

Um HTTPS zu aktivieren, benötigt Ihre Webserver-Anwendung (NGINX oder HAProxy) einen privaten Schlüssel und ein entsprechendes SSL/TLS Zertifikat. Um das Auslagern von SSL/TLS Webserver-Serveridentitäten mit verwenden zu können AWS CloudHSM, müssen Sie den privaten Schlüssel in einem HSM in Ihrem Cluster speichern. AWS CloudHSM Sie generieren zunächst einen privaten Schlüssel und verwenden diesen Schlüssel, um eine Certificate Signing Request (CSR) zu erstellen. Anschließend exportieren Sie einen gefälschten privaten PEM-Schlüssel aus dem HSM. Dabei handelt es sich um eine private Schlüsseldatei im PEM-Format, die einen Verweis auf den privaten Schlüssel enthält, der auf dem HSM gespeichert ist (es ist nicht der eigentliche private Schlüssel). Ihr Webserver verwendet die gefälschte private PEM-Schlüsseldatei, um den privaten Schlüssel auf dem HSM beim Auslagern der Serveridentität zu identifizieren. SSL/TLS

Generieren eines privaten Schlüssels

In diesem Abschnitt erfahren Sie, wie Sie mithilfe der [CloudHSM-CLI](#) ein Schlüsselpaar generieren. Sobald Sie ein key pair im HSM generiert haben, können Sie es als gefälschte PEM-Datei exportieren und das entsprechende Zertifikat generieren.

Installieren und konfigurieren Sie die CloudHSM-CLI

1. [Installieren und konfigurieren Sie](#) die CloudHSM-CLI.
2. Verwenden Sie den folgenden Befehl, um die CloudHSM-CLI zu starten.

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

3. Führen Sie den folgenden Befehl aus, um sich beim HSM anzumelden. Ersetzen Sie es *<user name>* durch den Benutzernamen Ihres Krypto-Benutzers

```
aws-cloudhsm>login --username <user name> --role crypto-user
```

Generieren eines privaten Schlüssels

Je nach Anwendungsfall können Sie entweder ein RSA- oder ein EC-Schlüsselpaar generieren. Führen Sie eine der folgenden Aktionen aus:

- So erstellen Sie einen privaten RSA-Schlüssel in einem HSM

Verwenden Sie den [key generate-asymmetric-pair rsa](#)-Befehl, um ein RSA-Schlüsselpaar zu erzeugen. In diesem Beispiel wird ein RSA-Schlüsselpaar mit einem Modul von 2048, einem öffentlichen Exponenten von 65537, der Bezeichnung des öffentlichen Schlüssels von und der Bezeichnung des privaten Schlüssels von *tls_rsa_pub* generiert. *tls_rsa_private*

```
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": "0x00000000000280cc8",
      "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": "0x00000000000280cc7",
  "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
    }
}
}
}

```

- So erstellen Sie einen privaten EC-Schlüssel in einem HSM

Verwenden Sie den [key generate-asymmetric-pair ec](#)-Befehl, um ein EC-Schlüsselpaar zu generieren. In diesem Beispiel wird ein EC-Schlüsselpaar mit der prime256v1 Kurve (die der NID_X9_62_prime256v1 Kurve entspricht), einem öffentlichen Schlüssellabel von *tls_ec_pub* und einem privaten Schlüssellabel von generiert *tls_ec_private*.

```

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": "0x0000000000012000b",
      "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"
  }
}
}
}

```

Exportieren einer gefälschten privaten PEM-Schlüsseldatei

Sobald Sie einen privaten Schlüssel auf dem HSM haben, müssen Sie eine gefälschte private PEM-Schlüsseldatei exportieren. Diese Datei enthält nicht die eigentlichen Schlüsseldaten, ermöglicht es der OpenSSL Dynamic Engine jedoch, den privaten Schlüssel auf dem HSM zu identifizieren. Sie können dann den privaten Schlüssel verwenden, um eine Zertifikatsignierungsanforderung (CSR) zu erstellen und die CSR zu signieren, um das Zertifikat zu erstellen.

Verwenden Sie den [key generate-file](#) Befehl, um den privaten Schlüssel im falschen PEM-Format zu exportieren und in einer Datei zu speichern. Ersetzen Sie die folgenden Werte durch Ihre eigenen.

- `<private_key_label>`— Bezeichnung des privaten Schlüssels, den Sie im vorherigen Schritt generiert haben.
- `<web_server_fake_pem.key>`— Name der Datei, in die Ihr gefälschter PEM-Schlüssel geschrieben wird.

```
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"  
  }  
}
```

Beenden Sie die CloudHSM-CLI

Führen Sie den folgenden Befehl aus, um die CloudHSM-CLI zu beenden.

```
aws-cloudhsm > quit
```

Sie sollten jetzt eine neue Datei auf Ihrem System haben, die sich unter dem `<web_server_fake_pem.key>` im vorherigen Befehl angegebenen Pfad befindet. Diese Datei ist die gefälschte private PEM-Schlüsseldatei.

Generieren eines selbstsignierten Zertifikats

Sobald Sie einen gefälschten privaten PEM-Schlüssel generiert haben, können Sie diese Datei verwenden, um eine Zertifikatssignieranforderung (CSR) und ein Zertifikat zu generieren.

In einer Produktionsumgebung verwenden Sie in der Regel eine Zertifikatsstelle (CA) zum Erstellen eines Zertifikats aus einer CSR. Für eine Testumgebung ist keine CA erforderlich. Wenn Sie eine Zertifizierungsstelle verwenden, senden Sie ihnen die CSR-Datei und verwenden Sie ein signiertes SSL/TLS Zertifikat, das sie Ihnen auf Ihrem Webserver für HTTPS zur Verfügung stellen.

Als Alternative zur Verwendung einer CA können Sie die AWS CloudHSM OpenSSL Dynamic Engine verwenden, um ein selbstsigniertes Zertifikat zu erstellen. Selbstsignierte Zertifikate sind nicht

vertrauenswürdig für Browser und sollten in Produktionsumgebungen nicht verwendet werden. Sie können in Testumgebungen verwendet werden.

 Warning

Selbstsignierte Zertifikate sollten nur in einer Testumgebung verwendet werden. Für eine Produktionsumgebung, verwenden Sie eine sicherere Methode, wie z. B. eine Zertifikatsstelle, um ein Zertifikat zu erstellen.

Installieren und konfigurieren Sie die OpenSSL Dynamic Engine

1. Stellen Sie eine Verbindung mit Ihrer Client-Instance her.
2. [the section called “Installieren”](#)

Generieren eines Zertifikats

1. Besorgen Sie sich eine Kopie Ihrer gefälschten PEM-Datei, die in einem früheren Schritt generiert wurde.
2. Erstellen einer CSR

Führen Sie den folgenden Befehl aus, um mit der AWS CloudHSM OpenSSL Dynamic Engine eine Certificate Signing Request (CSR) zu erstellen. `<web_server_fake_pem.key>` Ersetzen Sie ihn durch den Namen der Datei, die Ihren gefälschten privaten PEM-Schlüssel enthält. `<web_server.csr>` Ersetzen Sie es durch den Namen der Datei, die Ihre CSR enthält.

Der Befehl `req` ist interaktiv. Füllen Sie jedes Feld aus. Die Feldinformationen werden in Ihr SSL/TLS Zertifikat kopiert.

 Note

Die CSR-Erstellung wird derzeit vom OpenSSL Provider nicht unterstützt. Sie müssen für diesen Schritt die OpenSSL Engine verwenden, aber TLS-Verschlüsselungsoperationen funktionieren mit dem Anbieter.

```
$ openssl req -engine cloudhsm -new -key <web_server_fake_pem.key> -  
out <web_server.csr>
```

3. Erstellen eines selbstsignierten Zertifikats

Führen Sie den folgenden Befehl aus, um die AWS CloudHSM OpenSSL Dynamic Engine zu verwenden, um Ihre CSR mit Ihrem privaten Schlüssel auf Ihrem HSM zu signieren. Dadurch wird ein selbstsigniertes Zertifikat erstellt. Ersetzen Sie die folgenden Werte in dem Befehl durch Ihre eigenen.

- *<web_server.csr>*— Name der Datei, die die CSR enthält.
- *<web_server_fake_pem.key>*— Name der Datei, die den gefälschten privaten PEM-Schlüssel enthält.
- *<web_server.crt>*— Name der Datei, die Ihr Webserver-Zertifikat enthalten wird.

```
$ openssl x509 -engine cloudhsm -req -days 365 -in <web_server.csr> -  
signkey <web_server_fake_pem.key> -out <web_server.crt>
```

Nachdem Sie einen privaten Schlüssel und ein Zertifikat haben, gehen Sie zu [Schritt 3: Konfigurieren des Webserver](#)s.

Schritt 3: Konfigurieren des Webserver

Aktualisieren Sie die Konfiguration Ihrer Webserver-Software, um das HTTPS-Zertifikat und den zugehörigen gefälschten privaten PEM-Schlüssel zu verwenden, die Sie im [vorherigen Schritt](#) erstellt haben. Denken Sie daran, Ihre vorhandenen Zertifikate und Schlüssel zu sichern, bevor Sie beginnen. Damit ist die Einrichtung Ihrer Linux-Webserver-Software für das Auslagern von SSL/TLS Serveridentitäten mit abgeschlossen. AWS CloudHSM

Führen Sie die Schritte aus einem der folgenden Abschnitte aus.

Themen

- [Konfigurieren eines NGINX-Webserver](#)s
- [Webserver konfigurieren HAProxy](#)

Konfigurieren eines NGINX-Webservers

Verwenden Sie diesen Abschnitt, um NGINX mit dem OpenSSL Provider zu konfigurieren.

So konfigurieren Sie NGINX für OpenSSL Provider

1. Stellen Sie eine Verbindung mit Ihrer Client-Instance her.
2. Führen Sie den folgenden Befehl aus, um die erforderlichen Verzeichnisse für das Webserverzertifikat und den gefälschten privaten PEM-Schlüssel zu erstellen.

```
$ mkdir -p /etc/pki/nginx/private
```

3. Führen Sie den folgenden Befehl aus, um Ihr Webserverzertifikat an die erforderliche Stelle zu kopieren. Ersetzen Sie es `<web_server.crt>` durch den Namen Ihres Webserver-Zertifikats.

```
$ cp <web_server.crt> /etc/pki/nginx/server.crt
```

4. Führen Sie den folgenden Befehl aus, um den gefälschten privaten PEM-Schlüssel an die erforderliche Stelle zu kopieren. `<web_server_fake_pem.key>` Ersetzen Sie es durch den Namen der Datei, die Ihren gefälschten privaten PEM-Schlüssel enthält.

```
$ cp <web_server_fake_pem.key> /etc/pki/nginx/private/server.key
```

5. Führen Sie den folgenden Befehl aus, um die Eigentümerschaft für diese Dateien zu ändern, sodass der Benutzer mit dem Namen nginx diese lesen kann.

```
$ chown nginx /etc/pki/nginx/server.crt /etc/pki/nginx/private/server.key
```

6. Konfigurieren Sie OpenSSL für die Verwendung des AWS CloudHSM Anbieters. Weitere Informationen zur Konfiguration des OpenSSL Providers finden Sie unter [AWS CloudHSM OpenSSL Provider for Client SDK 5](#).

1. Suchen Sie Ihre OpenSSL-Konfigurationsdatei:

```
$ openssl version -d
```

Sie sollten eine Ausgabe sehen, die der folgenden ähnelt:

```
OPENSSLDIR: "/etc/pki/tls"
```

Die Konfigurationsdatei befindet sich `openssl.cnf` in diesem Verzeichnis.

2.

 Note

Ändern Sie die Standarddatei `openssl.cnf` Ihres Systems nicht direkt. Dadurch wird verhindert, dass systemweite OpenSSL-Operationen (SSH, TLS-Verbindungen und andere Dienste) unbeabsichtigt über den CloudHSM-Anbieter weitergeleitet werden. Durch die Verwendung einer separaten Konfigurationsdatei können Sie die Nutzung des CloudHSM Providers auf bestimmte Anwendungen beschränken, die HSM-gestützte kryptografische Operationen erfordern.

Erstellen Sie eine neue OpenSSL-Konfigurationsdatei mit dem folgenden Inhalt:

```
$ 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. Stellen Sie sicher, dass die `CLOUDHSM_PIN` Umgebungsvariable mit Ihren Crypto-Benutzeranmeldedaten (CU) gesetzt ist:

```
$ export CLOUDHSM_PIN=<username>:<password>
```

4. Stellen Sie die `OPENSSL_CONF` Umgebungsvariable so ein, dass sie auf Ihre aktualisierte Konfigurationsdatei verweist, und stellen Sie sicher, dass der Anbieter geladen ist:

```
$ OPENSSL_CONF=/path/to/example-cloudhsm-openssl.cnf openssl list -providers
```

Sie sollten sowohl den Standardanbieter als auch den CloudHSM-Anbieter aufgelistet sehen:

```
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. Führen Sie den folgenden Befehl aus, um die Datei `/etc/nginx/nginx.conf` zu sichern.

```
$ cp /etc/nginx/nginx.conf /etc/nginx/nginx.conf.backup
```

8. Aktualisieren Sie die NGINX-Konfiguration.

Note

Jeder Cluster kann maximal 1000 NGINX-Worker-Prozesse auf allen NGINX-Webservern unterstützen.

Amazon Linux 2023

Verwenden Sie einen Texteditor, um die Datei `/etc/nginx/nginx.conf` zu bearbeiten. Dafür sind möglicherweise Linux-Root-Rechte erforderlich. Fügen Sie am Anfang der Datei die folgende Zeilen hinzu:

```
env CLOUDHSM_PIN;
env OPENSSL_CONF;
```

Fügen Sie dann dem Abschnitt TLS der Datei Folgendes hinzu:

```
# 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+)

Verwenden Sie einen Texteditor, um die Datei `/etc/nginx/nginx.conf` zu bearbeiten. Dies erfordert möglicherweise Linux-Root-Rechte. Fügen Sie am Anfang der Datei die folgende Zeilen hinzu:

```
env CLOUDHSM_PIN;
env OPENSSL_CONF;
```

Fügen Sie dann dem Abschnitt TLS der Datei Folgendes hinzu:

```
# 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+)

Verwenden Sie einen Texteditor, um die Datei `/etc/nginx/nginx.conf` zu bearbeiten. Dies erfordert möglicherweise Linux-Root-Rechte. Fügen Sie am Anfang der Datei die folgende Zeilen hinzu:

```

env CLOUDHSM_PIN;
env OPENSSL_CONF;

```

Fügen Sie dann dem Abschnitt TLS der Datei Folgendes hinzu:

```

# 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 24.04

Verwenden Sie einen Texteditor, um die Datei `/etc/nginx/nginx.conf` zu bearbeiten. Dies erfordert möglicherweise Linux-Root-Rechte. Fügen Sie am Anfang der Datei die folgende Zeilen hinzu:

```

env CLOUDHSM_PIN;
env OPENSSL_CONF;

```

Fügen Sie dann dem Abschnitt TLS der Datei Folgendes hinzu:

```

# 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 {

}
```

Speichern Sie die Datei.

9. Sichern Sie die systemd-Konfigurationsdatei und legen Sie dann den EnvironmentFile-Pfad fest.

Amazon Linux 2023

1. Erstellen Sie eine Sicherungskopie der `nginx.service` Datei:

```
$ cp /lib/systemd/system/nginx.service /lib/systemd/system/nginx.service.backup
```


2. Öffnen Sie `/lib/systemd/system/nginx.service` in einem Texteditor. Fügen Sie im Abschnitt `[Service]` Folgendes hinzu:

```
EnvironmentFile=/etc/sysconfig/nginx
```

RHEL 9 (9.2+)

1. Erstellen Sie eine Sicherungskopie der `nginx.service` Datei:

```
$ cp /lib/systemd/system/nginx.service /lib/systemd/system/  
nginx.service.backup
```

2. Öffnen Sie `/lib/systemd/system/nginx.service` in einem Texteditor. Fügen Sie im Abschnitt `[Service]` Folgendes hinzu:

```
EnvironmentFile=/etc/sysconfig/nginx
```

RHEL 10 (10.0+)

1. Erstellen Sie eine Sicherungskopie der `nginx.service` Datei:

```
$ cp /lib/systemd/system/nginx.service /lib/systemd/system/  
nginx.service.backup
```

2. Öffnen Sie `/lib/systemd/system/nginx.service` in einem Texteditor. Fügen Sie im Abschnitt `[Service]` Folgendes hinzu:

```
EnvironmentFile=/etc/sysconfig/nginx
```

Ubuntu 24.04

1. Erstellen Sie eine Sicherungskopie der `nginx.service` Datei:

```
$ cp /lib/systemd/system/nginx.service /lib/systemd/system/  
nginx.service.backup
```

2. Öffnen Sie `/lib/systemd/system/nginx.service` in einem Texteditor. Fügen Sie im Abschnitt [Service] Folgendes hinzu:

```
EnvironmentFile=/etc/sysconfig/nginx
```

10. Überprüfen Sie, ob die Datei `/etc/sysconfig/nginx` vorhanden ist, und führen Sie dann einen der folgenden Schritte aus:

- Wenn die Datei vorhanden ist, sichern Sie die Datei, indem Sie den folgenden Befehl ausführen:

```
$ cp /etc/sysconfig/nginx /etc/sysconfig/nginx.backup
```

- Öffnen Sie einen Texteditor und erstellen Sie im Ordner `/etc/sysconfig/` eine Datei mit dem Namen `nginx`, wenn die Datei nicht vorhanden ist.

11. Konfigurieren Sie die NGINX-Umgebung.

Amazon Linux 2023

Öffnen Sie als Linux-Root-Benutzer die `/etc/sysconfig/nginx` Datei in einem Texteditor. Zum Beispiel

```
vi /etc/sysconfig/nginx
```

Fügen Sie die Cryptography User (CU) -Anmeldeinformationen und den Pfad zu Ihrer OpenSSL-Konfigurationsdatei hinzu:

```
CLOUDHSM_PIN=<CU user name>:<password>  
OPENSSL_CONF=<path to example-cloudhsm-openssl.cnf>
```

Ersetzen Sie `<CU user name>` und `<password>` durch die CU-Anmeldeinformationen. `<path to example-cloudhsm-openssl.cnf>` Ersetzen Sie durch den vollständigen Pfad zu der Konfigurationsdatei, in der Sie sie erstellt haben [So konfigurieren Sie NGINX für OpenSSL Provider](#).

Speichern Sie die Datei.

RHEL 9 (9.2+)

Öffnen Sie die Datei `/etc/sysconfig/nginx` in einem Text-Editor. Dies erfordert möglicherweise Linux-Root-Rechte. Fügen Sie die Cryptography User (CU) - Anmeldeinformationen und den Pfad zu Ihrer OpenSSL-Konfigurationsdatei hinzu:

```
CLOUDHSM_PIN=<CU user name>:<password>  
OPENSSL_CONF=<path to example-cloudhsm-openssl.cnf>
```

Ersetzen Sie *<CU user name>* und *<password>* durch die CU-Anmeldeinformationen. *<path to example-cloudhsm-openssl.cnf>* Ersetzen Sie durch den vollständigen Pfad zu der Konfigurationsdatei, in der Sie sie erstellt haben [So konfigurieren Sie NGINX für OpenSSL Provider](#).

Speichern Sie die Datei.

RHEL 10 (10.0+)

Öffnen Sie die Datei `/etc/sysconfig/nginx` in einem Text-Editor. Dies erfordert möglicherweise Linux-Root-Rechte. Fügen Sie die Cryptography User (CU) - Anmeldeinformationen und den Pfad zu Ihrer OpenSSL-Konfigurationsdatei hinzu:

```
CLOUDHSM_PIN=<CU user name>:<password>  
OPENSSL_CONF=<path to example-cloudhsm-openssl.cnf>
```

Ersetzen Sie *<CU user name>* und *<password>* durch die CU-Anmeldeinformationen. *<path to example-cloudhsm-openssl.cnf>* Ersetzen Sie durch den vollständigen Pfad zu der Konfigurationsdatei, in der Sie sie erstellt haben [So konfigurieren Sie NGINX für OpenSSL Provider](#).

Speichern Sie die Datei.

Ubuntu 24.04

Öffnen Sie die Datei `/etc/sysconfig/nginx` in einem Text-Editor. Dies erfordert möglicherweise Linux-Root-Rechte. Fügen Sie die Cryptography User (CU) - Anmeldeinformationen und den Pfad zu Ihrer OpenSSL-Konfigurationsdatei hinzu:

```
CLOUDHSM_PIN=<CU user name>:<password>  
OPENSSL_CONF=<path to example-cloudhsm-openssl.cnf>
```

Ersetzen Sie *<CU user name>* und *<password>* durch die CU-Anmeldeinformationen. *<path to example-cloudhsm-openssl.cnf>* Ersetzen Sie durch den vollständigen Pfad zu der Konfigurationsdatei, in der Sie sie erstellt haben [So konfigurieren Sie NGINX für OpenSSL Provider](#).

Speichern Sie die Datei.

12. Starten Sie den NGINX-Webserver.

Amazon Linux 2023

Stoppen Sie alle NGINX-Prozesse

```
$ systemctl stop nginx
```

Laden Sie die systemd-Konfiguration neu, um die neuesten Änderungen zu erhalten

```
$ systemctl daemon-reload
```

Starten Sie NGINX

```
$ systemctl start nginx
```

RHEL 9 (9.2+)

Stoppen Sie alle laufenden NGINX-Prozesse

```
$ systemctl stop nginx
```

Laden Sie die systemd-Konfiguration neu, um die neuesten Änderungen zu erhalten

```
$ systemctl daemon-reload
```

Starten Sie den NGINX-Prozess

```
$ systemctl start nginx
```

RHEL 10 (10.0+)

Stoppen Sie alle laufenden NGINX-Prozesse

```
$ systemctl stop nginx
```

Laden Sie die systemd-Konfiguration neu, um die neuesten Änderungen zu erhalten

```
$ systemctl daemon-reload
```

Starten Sie den NGINX-Prozess

```
$ systemctl start nginx
```

Ubuntu 24.04

Stoppen Sie alle laufenden NGINX-Prozesse

```
$ systemctl stop nginx
```

Laden Sie die systemd-Konfiguration neu, um die neuesten Änderungen zu erhalten

```
$ systemctl daemon-reload
```

Starten Sie den NGINX-Prozess

```
$ systemctl start nginx
```

Nachdem Sie NGINX konfiguriert haben, gehen Sie zu. [Verifizieren, dass HTTPS das konfigurierte Zertifikat verwendet](#)

Webserver konfigurieren HAProxy

Verwenden Sie diesen Abschnitt, um HAProxy mit dem OpenSSL Provider zu konfigurieren. Die folgenden Beispiele zeigen, wie Sie HAProxy mit Ihren CloudHSM-Zertifikaten und -Schlüsseln einrichten.

So konfigurieren Sie HAProxy für OpenSSL Provider

1. Erstellen Sie eine Sicherungskopie der vorhandenen kombinierten Zertifikatsdatei, falls vorhanden:

```
$ cp server-combined.pem server-combined.pem.backup
```

2. Erstellen Sie eine kombinierte Zertifikatsdatei für die HAProxy Verwendung Ihres Zertifikats und des gefälschten CloudHSM-PEM-Schlüssels:

```
$ cat server.crt server.key > server-combined.pem
```

3. Erstellen Sie eine Sicherungskopie der vorhandenen Konfiguration: HAProxy

```
$ cp /etc/haproxy/haproxy.cfg /etc/haproxy/haproxy.cfg.backup
```

4. Erstellen Sie eine neue CloudHSM-TLS-Offload-Konfiguration unter: `/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
```

Aktualisieren Sie den Zertifikatspfad so, dass er mit Ihrem Dateispeicherort übereinstimmt.

5. Konfigurieren Sie Systemd so, dass es eine Umgebungsdatei für HAProxy verwendet. Der Speicherort hängt von Ihrer Linux-Distribution ab.

Amazon Linux and RHEL

Sichern und ändern Sie die HAProxy Servicedatei:

```
$ cp /lib/systemd/system/haproxy.service /lib/systemd/system/
haproxy.service.backup
```

Bearbeiten Sie die folgende Zeile `/lib/systemd/system/haproxy.service` und fügen Sie sie im Abschnitt `[Service]` hinzu:

```
EnvironmentFile=/etc/sysconfig/haproxy
```

Ubuntu

Erstellen Sie eine Sicherungskopie und ändern Sie die HAProxy Servicedatei:

```
$ cp /lib/systemd/system/haproxy.service /lib/systemd/system/
haproxy.service.backup
```

Bearbeiten Sie die folgende Zeile `/lib/systemd/system/haproxy.service` und fügen Sie sie im Abschnitt `[Service]` hinzu:

```
EnvironmentFile=/etc/default/haproxy
```

6. Erstellen Sie die Umgebungsdatei am entsprechenden Speicherort für Ihr System.

Amazon Linux and RHEL

Sichern Sie die HAProxy Umgebungsdatei, falls sie existiert:

```
$ cp /etc/sysconfig/haproxy /etc/sysconfig/haproxy.backup
```

Erstellen Sie die HAProxy Umgebungsdatei `/etc/sysconfig/haproxy` mit dem folgenden Inhalt:

```
CLOUDHSM_PIN=<CU user name>:<password>
```

Ubuntu

Erstellen Sie eine HAProxy Sicherungskopie der Umgebungsdatei, falls sie existiert:

```
$ cp /etc/default/haproxy /etc/default/haproxy.backup
```

Erstellen Sie die HAProxy Umgebungsdatei `/etc/default/haproxy` mit dem folgenden Inhalt:

```
CLOUDHSM_PIN=<CU user name>:<password>
```

Ersetzen Sie `<CU user name>` und `<password>` durch Ihre CU-Anmeldeinformationen.

7. Laden Sie die Systemd-Konfiguration neu:

```
$ systemctl daemon-reload
```

8. Beginnen Sie HAProxy mit der CloudHSM TLS-Offload-Konfiguration:

```
$ systemctl start haproxy
```

Sie können den Vorgang auch HAProxy direkt mit einer benutzerdefinierten Konfigurationsdatei ausführen:

```
$ haproxy -f /path/to/haproxy-cloudhsm.cfg
```

Gehen Sie nach HAProxy der Konfiguration zu [Verifizieren, dass HTTPS das konfigurierte Zertifikat verwendet](#).

Schritt 4: Aktivieren von HTTPS-Datenverkehr und Verifizieren des Zertifikats

Nachdem Sie Ihren Webserver für SSL/TLS Offload mit konfiguriert haben AWS CloudHSM, fügen Sie Ihre Webserver-Instanz einer Sicherheitsgruppe hinzu, die eingehenden HTTPS-Verkehr zulässt. Dadurch können Clients, wie z. B. Webbrowser, eine HTTPS-Verbindung mit Ihrem Webserver herstellen. Stellen Sie dann eine HTTPS-Verbindung zu Ihrem Webserver her und stellen Sie sicher, dass er das Zertifikat verwendet, mit dem Sie das SSL/TLS Offload konfiguriert haben. AWS CloudHSM

Themen

- [Aktivieren von eingehenden HTTPS-Verbindungen](#)
- [Verifizieren, dass HTTPS das konfigurierte Zertifikat verwendet](#)

Aktivieren von eingehenden HTTPS-Verbindungen

Zum Herstellen einer Verbindung zu Ihrem Webserver von einem Client (z. B. ein Webbrowser) aus, erstellen Sie eine Sicherheitsgruppe, die eingehende HTTPS-Verbindungen zulässt. Insbesondere sollten eingehende TCP-Verbindungen auf Port 443 erlaubt werden. Weisen Sie diese Sicherheitsgruppe Ihrem Webserver zu.

So erstellen Sie eine Sicherheitsgruppe für HTTPS und weisen sie Ihrem Webserver zu

1. Öffnen Sie die Amazon-EC2-Konsole unter <https://console.aws.amazon.com/ec2/>.
2. Wählen Sie im Navigationsbereich Sicherheitsgruppen aus.
3. Wählen Sie Sicherheitsgruppe erstellen.
4. Führen Sie für Sicherheitsgruppe erstellen die folgenden Schritte aus:
 - a. Geben Sie in das Feld Sicherheitsgruppenname einen Namen für die Sicherheitsgruppe ein, die Sie erstellen.
 - b. (Optional) Geben Sie eine Beschreibung der Sicherheitsgruppe ein, die Sie erstellen.
 - c. Wählen Sie für VPC die VPC aus, die Ihre Amazon-EC2-Instance enthält.
 - d. Wählen Sie Regel hinzufügen aus.
 - e. Wählen Sie im Drop-down-Fenster für Typ die Option HTTPS aus.
 - f. Geben Sie für Quelle einen Quellspeicherort ein.
 - g. Wählen Sie Sicherheitsgruppe erstellen aus.
5. Wählen Sie im Navigationsbereich Instances aus.

6. Aktivieren Sie das Kontrollkästchen neben Ihrer Webserver-Instance.
7. Wählen Sie das Drop-down-Menü Aktionen oben auf der Seite. Wählen Sie Sicherheit und dann Sicherheitsgruppen ändern aus.
8. Wählen Sie unter Zugeordnete Sicherheitsgruppen das Suchfeld aus und wählen Sie die Sicherheitsgruppe, die Sie für HTTPS erstellt haben, aus. Wählen Sie dann Sicherheitsgruppen hinzufügen aus.
9. Wählen Sie Speichern.

Verifizieren, dass HTTPS das konfigurierte Zertifikat verwendet

Nachdem Sie den Webserver zu einer Sicherheitsgruppe hinzugefügt haben, können Sie überprüfen, ob für den SSL/TLS Offload Ihr selbstsigniertes Zertifikat verwendet wird. Sie können dazu einen Webbrowser oder ein Tool wie [OpenSSL s_client](#) nutzen.

Um den SSL/TLS Offload mit einem Webbrowser zu überprüfen

1. Verwenden Sie einen Web-Browser, um eine Verbindung zum Webserver unter Verwendung des öffentlichen DNS-Namen oder der IP-Adresse des Servers herzustellen. Stellen Sie sicher, dass die URL in die Adresszeile mit https:// beginnt. Beispiel, **`https://ec2-52-14-212-67.us-east-2.compute.amazonaws.com/`**.

 Tip

Sie können einen DNS-Dienst wie Amazon Route 53 verwenden, um den Domainnamen Ihrer Website (z. B. `https://www.example.com/`) an Ihren Webserver weiterzuleiten. Weitere Informationen finden Sie unter [Routing des Datenverkehrs zu einer Amazon-EC2-Instance](#) im Entwicklerleitfaden zu Amazon Route 53 oder in der Dokumentation für Ihren DNS-Service.

2. Zeigen Sie das Webserverzertifikat mit Ihrem Webbrowser an. Weitere Informationen finden Sie hier:
 - Wenn Sie Mozilla Firefox nutzen, sehen Sie sich die Informationen auf der Mozilla Support-Website unter [Zertifikat anzeigen](#) an.
 - Wenn Sie Google Chrome verwenden, sehen Sie sich die Informationen auf der „Google Tools für Web Developers“-Website unter [Sicherheitsprobleme verstehen](#) an.

Andere Webbrowser unterstützen möglicherweise ähnliche Funktionen, über die Sie das Webserverzertifikat anzeigen können.

3. Stellen Sie sicher, dass es sich bei dem SSL/TLS Zertifikat um das Zertifikat handelt, für dessen Verwendung Sie Ihren Webserver konfiguriert haben.

Um den SSL/TLS Offload mit OpenSSL s_client zu überprüfen

1. Führen Sie den folgenden OpenSSL-Befehl aus, um mittels HTTPS eine Verbindung zu Ihrem Webserver herzustellen. `<server name>` Ersetzen Sie es durch den öffentlichen DNS-Namen oder die IP-Adresse Ihres Webserver.

```
openssl s_client -connect <server name>:443
```

 Tip

Sie können einen DNS-Dienst wie Amazon Route 53 verwenden, um den Domainnamen Ihrer Website (z. B. `https://www.example.com/`) an Ihren Webserver weiterzuleiten. Weitere Informationen finden Sie unter [Routing des Datenverkehrs zu einer Amazon-EC2-Instance](#) im Entwicklerleitfaden zu Amazon Route 53 oder in der Dokumentation für Ihren DNS-Service.

2. Stellen Sie sicher, dass es sich bei dem SSL/TLS Zertifikat um das Zertifikat handelt, für dessen Verwendung Sie Ihren Webserver konfiguriert haben.

Sie verfügen jetzt über eine mit HTTPS gesicherte Website. Der private Schlüssel für den Webserver wird in einem HSM in Ihrem AWS CloudHSM Cluster gespeichert.

Informationen zum Hinzufügen eines Load Balancers finden Sie unter [Einen Load Balancer mit Elastic Load Balancing hinzufügen für AWS CloudHSM\(optional\)](#).

AWS CloudHSM SSL/TLS-Offload unter Linux mit Tomcat mit JSSE

Dieses Thema enthält step-by-step Anweisungen zum Einrichten von SSL/TLS Offloads mithilfe der Java Secure Socket Extension (JSSE) mit dem JCE SDK. AWS CloudHSM

Themen

- [-Übersicht](#)
- [Schritt 1: Einrichten der Voraussetzungen](#)
- [Schritt 2: Generieren oder importieren Sie einen privaten Schlüssel und ein Zertifikat SSL/TLS](#)
- [Schritt 3: Konfigurieren des Tomcat-Webserver](#)
- [Schritt 4: Aktivieren von HTTPS-Datenverkehr und Verifizieren des Zertifikats](#)

-Übersicht

In AWS CloudHSM: Tomcat-Webserver funktionieren unter Linux und unterstützen HTTPS. Das AWS CloudHSM JCE SDK bietet eine Schnittstelle, die mit JSSE (Java Secure Socket Extension) verwendet werden kann, um die Verwendung von HSMs für solche Webserver zu ermöglichen. AWS CloudHSM JCE ist die Brücke, die JSSE mit Ihrem AWS CloudHSM CloudHSM-Cluster verbindet. JSSE ist eine Java-API für Secure Sockets Layer (SSL) und Transport Layer Security (TLS).

Schritt 1: Einrichten der Voraussetzungen

Erfüllen Sie diese Voraussetzungen, um einen Tomcat-Webserver AWS CloudHSM für SSL/TLS Offload unter Linux zu verwenden. Diese Voraussetzungen müssen erfüllt sein, um den SSL/TLS Webserver-Offload mit dem Client SDK 5 und einem Tomcat-Webserver einzurichten.

Note

Verschiedene Plattformen erfordern unterschiedliche Voraussetzungen. Folgen Sie immer den richtigen Installationsschritten für Ihre Plattform.

Voraussetzungen

- Eine Amazon EC2-Instance, auf der ein Linux-Betriebssystem mit einem installierten Tomcat-Webserver ausgeführt wird.
- Ein [Crypto-Benutzer](#) (CU), der den privaten Schlüssel des Webserver auf dem HSM besitzen und verwalten soll.
- Ein aktiver AWS CloudHSM Cluster mit mindestens zwei Hardware-Sicherheitsmodulen (HSMs), auf denen [JCE for Client SDK 5](#) installiert und konfiguriert ist.

 Note

Sie können einen einzelnen HSM-Cluster verwenden, müssen aber zuerst die Haltbarkeit der Client-Schlüssel deaktivieren. Weitere Informationen finden Sie unter [Einstellungen für die Haltbarkeit von Client-Schlüsseln verwalten](#) und [Client-SDK 5 Configure Tool](#).

Wie erfüllt man die Voraussetzungen

1. Installieren und konfigurieren Sie das JCE für AWS CloudHSM auf einem aktiven AWS CloudHSM Cluster mit mindestens zwei Hardware-Sicherheitsmodulen (HSMs). Weitere Informationen zur Installation finden Sie unter [JCE für Client-SDK 5](#).
2. Folgen Sie auf einer EC2-Linux-Instance, die Zugriff auf Ihren AWS CloudHSM Cluster hat, den [Apache Tomcat-Anweisungen](#), um den Tomcat-Webserver herunterzuladen und zu installieren.
3. Verwenden Sie die [CloudHSM-CLI](#), um einen Crypto-Benutzer (CU) zu erstellen. Weitere Informationen zur Verwaltung von HSM-Benutzern finden Sie unter [HSM-Benutzer mit der CloudHSM-CLI verwalten](#).

 Tip

Merken Sie sich den CU-Benutzernamen und das Passwort. Sie benötigen sie später beim Generieren oder Importieren des privaten HTTPS-Schlüssels und -Zertifikats für Ihren Webserver.

4. Folgen Sie den Anweisungen unter [Verwenden Sie Client SDK 5 zur Integration AWS CloudHSM mit Java Keytool und Jarsigner](#), um JCE mit Java Keytool einzurichten.

Nachdem Sie diese Schritte abgeschlossen haben, fahren Sie mit [Schritt 2: Generieren oder importieren Sie einen privaten Schlüssel und ein Zertifikat SSL/TLS](#) fort.

Hinweise

- Um Security-Enhanced Linux (SELinux) und Webserver zu verwenden, müssen Sie ausgehende TCP-Verbindungen an Port 2223 zulassen, dem Port, den das Client SDK 5 für die Kommunikation mit dem HSM verwendet.

- Um einen Cluster zu erstellen und zu aktivieren und einer EC2-Instance Zugriff auf den Cluster zu gewähren, führen Sie die Schritte unter [Erste Schritte mit AWS CloudHSM](#) aus. Dieser Abschnitt enthält step-by-step Anweisungen zum Erstellen eines aktiven Clusters mit einem HSM und einer Amazon EC2 EC2-Client-Instance. Sie können diese Client-Instance als Ihren Webserver verwenden.
- Um zu vermeiden, dass die Haltbarkeit von Client-Schlüsseln deaktiviert wird, fügen Sie Ihrem Cluster mehr als ein HSM hinzu. Weitere Informationen finden Sie unter [Hinzufügen eines HSM zu einem Cluster AWS CloudHSM](#).
- Um sich mit Ihrer Client-Instance zu verbinden, können Sie SSH oder PuTTY verwenden. Weitere Informationen finden Sie unter [Herstellung einer Verbindung zu Ihrer Linux-Instance mit SSH](#) oder [Herstellung einer Verbindung zu Ihrer Linux-Instance von Windows mit PuTTY](#) in der Amazon EC2-Dokumentation.

Schritt 2: Generieren oder importieren Sie einen privaten Schlüssel und ein Zertifikat SSL/TLS

Um HTTPS zu aktivieren, benötigt Ihre Tomcat-Webserver-Anwendung einen privaten Schlüssel und ein entsprechendes Zertifikat. SSL/TLS Um das SSL/TLS Webserver-Offload mit verwenden zu können AWS CloudHSM, müssen Sie den privaten Schlüssel in einem HSM in Ihrem Cluster speichern. AWS CloudHSM

Note

Wenn Sie noch nicht über einen privaten Schlüssel und ein entsprechendes Zertifikat verfügen, generieren Sie einen privaten Schlüssel in einem HSM. Sie verwenden den privaten Schlüssel, um eine Certificate Signing Request (CSR) zu erstellen, mit der Sie das Zertifikat erstellen. SSL/TLS

Sie erstellen eine lokale AWS CloudHSM KeyStore Datei, die einen Verweis auf Ihren privaten Schlüssel auf dem HSM und das zugehörige Zertifikat enthält. Ihr Webserver verwendet die AWS CloudHSM KeyStore Datei, um den privaten Schlüssel auf dem HSM während SSL/TLS des Offloads zu identifizieren.

Themen

- [Generieren eines privaten Schlüssels](#)

- [Generieren eines selbstsignierten Zertifikats](#)

Generieren eines privaten Schlüssels

In diesem Abschnitt erfahren Sie, wie Sie mit dem KeyTool From JDK ein key pair generieren. Sobald Sie ein key pair im HSM generiert haben, können Sie es als KeyStore Datei exportieren und das entsprechende Zertifikat generieren.

Je nach Anwendungsfall können Sie entweder ein RSA- oder ein EC-Schlüsselpaar generieren. Im Folgenden wird beschrieben, wie Sie ein RSA-Schlüsselpaar generieren.

Verwenden Sie den **genkeypair** Befehl in KeyTool , um ein RSA-Schlüsselpaar zu generieren

1. Nachdem Sie die **<VARIABLES>** folgenden Daten durch Ihre spezifischen Daten ersetzt haben, verwenden Sie den folgenden Befehl, um eine Keystore-Datei mit dem Namen zu generieren `jsse_keystore.keystore`, die eine Referenz Ihres privaten Schlüssels auf dem HSM enthält.

```
$ 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>**: Der Pfad, in dem Sie Ihre Keystore-Datei generieren möchten.
- **<UNIQUE ALIAS FOR KEYS>**: Dies wird verwendet, um Ihren Schlüssel auf dem HSM eindeutig zu identifizieren. Dieser Alias wird als LABEL-Attribut für den Schlüssel festgelegt.
- **<KEY PASSWORD>**: Wir speichern den Verweis auf Ihren Schlüssel in der lokalen Keystore-Datei, und dieses Passwort schützt diese lokale Referenz.
- **<KEYSTORE PASSWORD>**: Dies ist das Passwort für Ihre lokale Keystore-Datei.
- **<JSSE KEYSTORE NAME>**: Name der Keystore-Datei.
- **<CERT DOMAIN NAME>**: X.500 Eindeutiger Name.
- **<KEY ALGORITHM>**: Schlüsselalgorithmus zur Generierung key pair (z. B. RSA und EC).
- **<KEY SIZE>**: Schlüsselgröße zur Generierung des key pair (z. B. 2048, 3072 und 4096).

- **<SIGN ALGORITHM>**: Schlüsselgröße zur Generierung des key pair (z. B. RSA, SHA1with RSA, SHA224with RSA, SHA256with RSA und SHA384with RSA). SHA512with
2. Um zu bestätigen, dass der Befehl erfolgreich war, geben Sie den folgenden Befehl ein und überprüfen Sie, ob Sie erfolgreich ein RSA-Schlüsselpaar generiert haben.

```
$ ls <PATH>/<JSSE KEYSTORE NAME>.keystore
```

Generieren eines selbstsignierten Zertifikats

Nachdem Sie zusammen mit der Keystore-Datei einen privaten Schlüssel generiert haben, können Sie diese Datei verwenden, um eine Certificate Signing Request (CSR) und ein Zertifikat zu generieren.

In einer Produktionsumgebung verwenden Sie in der Regel eine Zertifikatsstelle (CA) zum Erstellen eines Zertifikats aus einer CSR. Für eine Testumgebung ist keine CA erforderlich. Wenn Sie eine Zertifizierungsstelle verwenden, senden Sie ihnen die CSR-Datei und verwenden Sie ein signiertes SSL/TLS Zertifikat, das sie Ihnen auf Ihrem Webserver für HTTPS zur Verfügung stellen.

Als Alternative zur Verwendung einer Zertifizierungsstelle können Sie die verwenden, KeyTool um ein selbstsigniertes Zertifikat zu erstellen. Selbstsignierte Zertifikate sind nicht vertrauenswürdig für Browser und sollten in Produktionsumgebungen nicht verwendet werden. Sie können in Testumgebungen verwendet werden.

Warning

Selbstsignierte Zertifikate sollten nur in einer Testumgebung verwendet werden. Für eine Produktionsumgebung, verwenden Sie eine sicherere Methode, wie z. B. eine Zertifikatsstelle, um ein Zertifikat zu erstellen.

Generieren eines Zertifikats

1. Besorgen Sie sich eine Kopie Ihrer Keystore-Datei, die in einem früheren Schritt generiert wurde.
2. Führen Sie den folgenden Befehl aus, um mit dem eine Zertifikatsignieranforderung (CSR) KeyTool zu erstellen.

```
$ 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

Die Ausgabedatei der Zertifikatssignierungsanforderung ist `certreq.csr`.

Ein Zertifikat signieren

- Nachdem **<VARIABLES>** Sie die folgenden Daten durch Ihre spezifischen Daten ersetzt haben, führen Sie den folgenden Befehl aus, um Ihre CSR mit Ihrem privaten Schlüssel auf Ihrem HSM zu signieren. Dadurch wird ein selbstsigniertes Zertifikat erstellt.

```
$ 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` ist das signierte Zertifikat, das den privaten Schlüssel des Alias verwendet.

Ein Zertifikat in Keystore importieren

- Nachdem **<VARIABLES>** Sie das Folgende durch Ihre spezifischen Daten ersetzt haben, führen Sie den folgenden Befehl aus, um ein signiertes Zertifikat als vertrauenswürdiges Zertifikat zu importieren. In diesem Schritt wird das Zertifikat in dem durch den Alias identifizierten Keystore-Eintrag gespeichert.

```
$ 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>
```

Konvertiert ein Zertifikat in ein PEM

- Führen Sie den folgenden Befehl aus, um die signierte Zertifikatsdatei (.crt) in eine PEM-Datei zu konvertieren. Die PEM-Datei wird verwendet, um die Anfrage vom HTTP-Client zu senden.

```
$ openssl x509 -inform der -in certificate.crt -out certificate.pem
```

Nachdem Sie diese Schritte ausgeführt haben, fahren Sie mit [Schritt 3: Konfigurieren des Webservers](#) fort.

Schritt 3: Konfigurieren des Tomcat-Webservers

Aktualisieren Sie die Konfiguration Ihrer Webserver-Software, um das HTTPS-Zertifikat und die entsprechende PEM-Datei zu verwenden, die Sie im vorherigen Schritt erstellt haben. Denken Sie daran, Ihre vorhandenen Zertifikate und Schlüssel zu sichern, bevor Sie beginnen. Damit ist die Einrichtung Ihrer Linux-Webserver-Software für das SSL/TLS Offload mit abgeschlossen. AWS CloudHSM Weitere Informationen finden Sie in der [Konfigurationsreferenz für Apache Tomcat 9](#).

Den Server beenden

- Nachdem **<VARIABLES>** Sie die folgenden Daten durch Ihre spezifischen Daten ersetzt haben, führen Sie den folgenden Befehl aus, um Tomcat Server zu beenden, bevor Sie die Konfiguration aktualisieren

```
$ /<TOMCAT DIRECTORY>/bin/shutdown.sh
```

- <TOMCAT DIRECTORY>**: Ihr Tomcat-Installationsverzeichnis.

Aktualisieren Sie den Tomcat-Klassenpfad

- Stellen Sie eine Verbindung mit Ihrer Client-Instance her.
- Suchen Sie den Tomcat-Installationsordner.
- Nachdem **<VARIABLES>** Sie das Folgende durch Ihre spezifischen Daten ersetzt haben, verwenden Sie den folgenden Befehl, um die Java-Bibliothek und AWS CloudHSM den Java-Pfad in Tomcat hinzuzufügenclasspath, die sich in der SH-Datei befinden. Tomcat/bin/catalina

```
$ sed -i 's@CLASSPATH="$CLASSPATH""$CATALINA_HOME"/bin/\nbootstrap.jar@CLASSPATH="$CLASSPATH""$CATALINA_HOME"/bin/bootstrap.jar:'
```

```
<JAVA LIBRARY>'"/*:~/opt/cloudhsm/java/*:./@' <TOMCAT PATH> /bin/
catalina.sh
```

- **<JAVA LIBRARY>**: Speicherort der Java JRE-Bibliothek.
- **<TOMCAT PATH>**: Tomcat-Installationsordner.

Fügen Sie der Serverkonfiguration einen HTTPS-Connector hinzu.

1. Gehen Sie zum Tomcat-Installationsordner.
2. Nachdem **<VARIABLES>** Sie die folgenden Daten durch Ihre spezifischen Daten ersetzt haben, verwenden Sie den folgenden Befehl, um einen HTTPS-Connector hinzuzufügen, um Zertifikate zu verwenden, die in den Voraussetzungen generiert wurden:

```
$ 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>**: Verzeichnis, in dem sich die Keystore-Datei befindet.
- **<JSSE KEYSTORE NAME>**: Name der Keystore-Datei.
- **<KEYSTORE PASSWORD>**: Dies ist das Passwort für Ihre lokale Keystore-Datei.
- **<KEY PASSWORD>**: Wir speichern den Verweis auf Ihren Schlüssel in der lokalen Keystore-Datei, und dieses Passwort schützt diese lokale Referenz.
- **<UNIQUE ALIAS FOR KEYS>**: Dies wird verwendet, um Ihren Schlüssel auf dem HSM eindeutig zu identifizieren. Dieser Alias wird als LABEL-Attribut für den Schlüssel festgelegt.
- **<TOMCAT PATH>**: Der Pfad zu Ihrem Tomcat-Ordner.

Starten des Servers

- Nachdem **<VARIABLES>** Sie das Folgende durch Ihre spezifischen Daten ersetzt haben, verwenden Sie den folgenden Befehl, um Tomcat Server zu starten:

```
$ /<TOMCAT DIRECTORY>/bin/startup.sh
```

 Note

<TOMCAT DIRECTORY> ist der Name Ihres Tomcat-Installationsverzeichnis.

Nachdem Sie Ihre Webserverkonfiguration aktualisiert haben, gehen Sie zu [Schritt 4: Aktivieren von HTTPS-Datenverkehr und Verifizieren des Zertifikats](#).

Schritt 4: Aktivieren von HTTPS-Datenverkehr und Verifizieren des Zertifikats

Nachdem Sie Ihren Webserver für SSL/TLS Offload mit konfiguriert haben AWS CloudHSM, fügen Sie Ihre Webserver-Instanz einer Sicherheitsgruppe hinzu, die eingehenden HTTPS-Verkehr zulässt. Dadurch können Clients, wie z. B. Webbrowser, eine HTTPS-Verbindung mit Ihrem Webserver herstellen. Stellen Sie dann eine HTTPS-Verbindung zu Ihrem Webserver her und stellen Sie sicher, dass er das Zertifikat verwendet, mit dem Sie das SSL/TLS Offload konfiguriert haben. AWS CloudHSM

Themen

- [Aktivieren von eingehenden HTTPS-Verbindungen](#)
- [Verifizieren, dass HTTPS das konfigurierte Zertifikat verwendet](#)

Aktivieren von eingehenden HTTPS-Verbindungen

Zum Herstellen einer Verbindung zu Ihrem Webserver von einem Client (z. B. ein Webbrowser) aus, erstellen Sie eine Sicherheitsgruppe, die eingehende HTTPS-Verbindungen zulässt. Insbesondere sollten eingehende TCP-Verbindungen auf Port 443 erlaubt werden. Weisen Sie diese Sicherheitsgruppe Ihrem Webserver zu.

So erstellen Sie eine Sicherheitsgruppe für HTTPS und weisen sie Ihrem Webserver zu

1. Öffnen Sie die Amazon-EC2-Konsole unter <https://console.aws.amazon.com/ec2/>.
2. Wählen Sie im Navigationsbereich Sicherheitsgruppen aus.
3. Wählen Sie Sicherheitsgruppe erstellen.
4. Führen Sie für Sicherheitsgruppe erstellen die folgenden Schritte aus:
 - a. Geben Sie in das Feld Sicherheitsgruppenname einen Namen für die Sicherheitsgruppe ein, die Sie erstellen.

- b. (Optional) Geben Sie eine Beschreibung der Sicherheitsgruppe ein, die Sie erstellen.
 - c. Wählen Sie für VPC die VPC aus, die Ihre Amazon-EC2-Instance enthält.
 - d. Wählen Sie Regel hinzufügen aus.
 - e. Wählen Sie im Drop-down-Fenster für Typ die Option HTTPS aus.
 - f. Geben Sie für Quelle einen Quellspeicherort ein.
 - g. Wählen Sie Sicherheitsgruppe erstellen aus.
5. Wählen Sie im Navigationsbereich Instances aus.
 6. Aktivieren Sie das Kontrollkästchen neben Ihrer Webserver-Instance.
 7. Wählen Sie das Drop-down-Menü Aktionen oben auf der Seite. Wählen Sie Sicherheit und dann Sicherheitsgruppen ändern aus.
 8. Wählen Sie unter Zugeordnete Sicherheitsgruppen das Suchfeld aus und wählen Sie die Sicherheitsgruppe, die Sie für HTTPS erstellt haben, aus. Wählen Sie dann Sicherheitsgruppen hinzufügen aus.
 9. Wählen Sie Speichern.

Verifizieren, dass HTTPS das konfigurierte Zertifikat verwendet

Nachdem Sie den Webserver zu einer Sicherheitsgruppe hinzugefügt haben, können Sie überprüfen, ob der SSL/TLS Offload Ihr selbstsigniertes Zertifikat verwendet. Sie können dazu einen Webbrowser oder ein Tool wie [OpenSSL s_client](#) nutzen.

Um den SSL/TLS Offload mit einem Webbrowser zu überprüfen

1. Verwenden Sie einen Web-Browser, um eine Verbindung zum Webserver unter Verwendung des öffentlichen DNS-Namen oder der IP-Adresse des Servers herzustellen. Stellen Sie sicher, dass die URL in die Adresszeile mit https:// beginnt. Beispiel, **https://ec2-52-14-212-67.us-east-2.compute.amazonaws.com/**.

 Tip

Sie können einen DNS-Dienst wie Amazon Route 53 verwenden, um den Domainnamen Ihrer Website (z. B. https://www.example.com/) an Ihren Webserver weiterzuleiten. Weitere Informationen finden Sie unter [Routing des Datenverkehrs zu einer Amazon-EC2-Instance](#) im Entwicklerleitfaden zu Amazon Route 53 oder in der Dokumentation für Ihren DNS-Service.

2. Zeigen Sie das Webserverzertifikat mit Ihrem Webbrowser an. Weitere Informationen finden Sie hier:
 - Wenn Sie Mozilla Firefox nutzen, sehen Sie sich die Informationen auf der Mozilla Support-Website unter [Zertifikat anzeigen](#) an.
 - Wenn Sie Google Chrome verwenden, sehen Sie sich die Informationen auf der „Google Tools für Web Developers“-Website unter [Sicherheitsprobleme verstehen](#) an.

Andere Webbrowser unterstützen möglicherweise ähnliche Funktionen, über die Sie das Webserverzertifikat anzeigen können.

3. Stellen Sie sicher, dass es sich bei dem SSL/TLS Zertifikat um das Zertifikat handelt, für dessen Verwendung Sie Ihren Webserver konfiguriert haben.

Um den SSL/TLS Offload mit OpenSSL `s_client` zu überprüfen

1. Führen Sie den folgenden OpenSSL-Befehl aus, um mittels HTTPS eine Verbindung zu Ihrem Webserver herzustellen. `<server name>` Ersetzen Sie es durch den öffentlichen DNS-Namen oder die IP-Adresse Ihres Webserver.

```
openssl s_client -connect <server name>:443
```

 Tip

Sie können einen DNS-Dienst wie Amazon Route 53 verwenden, um den Domainnamen Ihrer Website (z. B. `https://www.example.com/`) an Ihren Webserver weiterzuleiten. Weitere Informationen finden Sie unter [Routing des Datenverkehrs zu einer Amazon-EC2-Instance](#) im Entwicklerleitfaden zu Amazon Route 53 oder in der Dokumentation für Ihren DNS-Service.

2. Stellen Sie sicher, dass es sich bei dem SSL/TLS Zertifikat um das Zertifikat handelt, für dessen Verwendung Sie Ihren Webserver konfiguriert haben.

Sie verfügen jetzt über eine mit HTTPS gesicherte Website. Der private Schlüssel für den Webserver wird in einem HSM in Ihrem AWS CloudHSM Cluster gespeichert.

Informationen zum Hinzufügen eines Load Balancers finden Sie unter [Einen Load Balancer mit Elastic Load Balancing hinzufügen für AWS CloudHSM\(optional\)](#).

AWS CloudHSM SSL/TLS-Offload unter Windows mithilfe von IIS mit KSP

Dieses Tutorial enthält step-by-step Anweisungen zum Einrichten von SSL/TLS Offload mit AWS CloudHSM auf einem Windows-Webserver.

Themen

- [-Übersicht](#)
- [Schritt 1: Einrichten der Voraussetzungen](#)
- [Schritt 2: Erstellen einer Zertifikatsignierungsanforderung \(CSR\) und eines Zertifikats](#)
- [Schritt 3: Konfigurieren des Webserver](#)
- [Schritt 4: Aktivieren von HTTPS-Datenverkehr und Verifizieren des Zertifikats](#)

-Übersicht

Unter Windows bietet die Webserveranwendung [Internet Information Services \(IIS\) für Windows Server](#) native Unterstützung für HTTPS. Der [AWS CloudHSM Key Storage Provider \(KSP\) für Microsofts Cryptography API: Next Generation \(CNG\)](#) stellt die Schnittstelle bereit, über die IIS den HSMs in Ihrem Cluster für kryptografisches Offloading und Schlüsselspeicher verwenden kann. Der AWS CloudHSM KSP ist die Brücke, die IIS mit Ihrem Cluster verbindet. AWS CloudHSM

In diesem Tutorial erfahren Sie, wie Sie folgende Aufgaben ausführen:

- Installieren der Web-Server-Software auf einer Amazon EC2-Instance
- Konfigurieren Sie die Webserver-Software so, dass sie HTTPS mit einem privaten Schlüssel unterstützt, der in Ihrem AWS CloudHSM -Cluster gespeichert ist.
- (Optional) Verwenden Sie Amazon EC2, um eine zweite Webserver-Instance zu erstellen, und Elastic Load Balancing, um einen Load Balancer zu erstellen. Mit einem Load Balancer kann die Leistung durch Verteilung der Arbeitslast auf mehrere Server gesteigert werden. Sie kann auch für Redundanz und eine höhere Verfügbarkeit sorgen, falls ein oder mehrere Webserver ausfallen.

Wenn Sie bereit sind, sehen Sie sich [Schritt 1: Einrichten der Voraussetzungen](#) an.

Schritt 1: Einrichten der Voraussetzungen

Verschiedene Plattformen erfordern unterschiedliche Voraussetzungen. Verwenden Sie den Abschnitt mit den Voraussetzungen unten, der zu Ihrer Plattform passt.

Themen

- [Voraussetzungen für das Client-SDK 5](#)
- [Voraussetzungen für das Client-SDK 3](#)

Voraussetzungen für das Client-SDK 5

Um das SSL/TLS Webserver-Offload mit einzurichten AWS CloudHSM, benötigen Sie Folgendes:

- Ein aktiver AWS CloudHSM Cluster mit mindestens einem HSM.
- Eine Amazon EC2-Instance, auf der ein Windows-Betriebssystem ausgeführt wird und die folgende Software installiert ist:
 - Die AWS CloudHSM Client-Software für Windows.
 - Internet Information Services (IIS) für Windows Server
- Ein [Crypto-Benutzer](#) (CU), der den privaten Schlüssel des Webserver auf dem HSM besitzen und verwalten soll.

Note

Dieses Tutorial verwendet Microsoft Windows Server 2019. Microsoft Windows Server 2016 und 2022 werden ebenfalls unterstützt.

So richten Sie eine Windows-Webserver-Instance auf dem HSM ein und erstellen einen CU

1. Führen Sie die Schritte unter [Erste Schritte](#) aus. Wenn Sie den Amazon EC2 Client starten, wählen Sie ein Windows Server 2019-AMI. Wenn Sie diese Schritte ausgeführt haben, verfügen Sie über einen aktiven Cluster mit mindestens einem HSM. Sie haben auch eine Amazon EC2 Instance, auf der Windows Server ausgeführt wird und auf der die AWS CloudHSM Client-Software für Windows installiert ist.
2. (Optional) Fügen Sie Ihrem Cluster weitere HSMs hinzu. Weitere Informationen finden Sie unter [Hinzufügen eines HSM zu einem Cluster AWS CloudHSM](#).

3. Stellen Sie eine Verbindung zu Ihrem Windows-Server her. Weitere Informationen finden Sie unter [Connect to Your Instance](#) im Amazon EC2 EC2-Benutzerhandbuch.
4. Verwenden Sie die CloudHSM-CLI, um einen Crypto-Benutzer (CU) zu erstellen. Merken Sie sich den CU-Benutzernamen und das Passwort. Sie benötigen diese im nächsten Schritt.

 Note

Informationen zum Erstellen eines Benutzers finden Sie unter [HSM-Benutzer mit der CloudHSM-CLI verwalten](#).

5. [Legen Sie die Anmeldeinformationen für das HSM fest](#). Verwenden Sie hierfür den im vorherigen Schritt erstellten CU-Benutzernamen und das entsprechende Passwort.
6. Wenn Sie in Schritt 5 den Windows Credentials Manager zum Einrichten [psexec.exe](#) von HSM-Anmeldeinformationen verwendet haben, laden Sie den folgenden Befehl als NT Authority\SYSTEM herunter, SysInternals um ihn auszuführen:

```
psexec.exe -s "C:\Program Files\Amazon\CloudHsm\tools\set_cloudhsm_credentials.exe"  
--username <USERNAME> --password <PASSWORD>
```

Ersetzen Sie **<USERNAME>** und **<PASSWORD>** durch die HSM-Anmeldeinformationen.

So installieren Sie IIS auf Ihrem Windows-Server

1. Stellen Sie eine Verbindung mit Ihrem Windows-Server her, sofern noch nicht geschehen. Weitere Informationen finden Sie unter [Connect to Your Instance](#) im Amazon EC2 EC2-Benutzerhandbuch.
2. Starten Sie auf Ihrem Windows-Server Server Manager.
3. Wählen Sie im Dashboard Server-Manager die Option Rollen und Features hinzufügen.
4. Lesen Sie die Informationen unter Before you begin (Zur Vorbereitung) und klicken Sie dann auf Weiter.
5. Wählen Sie unter Installationsart die Rollenbasierte oder funktionsbasierte Installation. Wählen Sie anschließend Weiter.
6. Wählen Sie unter Serverauswahl die Option Einen Server aus dem Serverpool auswählen. Klicken Sie anschließend auf Weiter.
7. Gehen Sie für Serverrollen wie folgt vor:

- a. Wählen Sie Web Server (IIS) (Webserver (IIS)) aus.
 - b. Klicken Sie für Add features that are required für Web Server (IIS) (Für Webserver (IIS) erforderliche Funktionen hinzufügen) auf Add Features (Funktionen hinzufügen).
 - c. Klicken Sie auf Weiter, um das Auswählen von Serverrollen abzuschließen.
8. Für Features (Funktionen), übernehmen Sie die Standardeinstellungen. Klicken Sie anschließend auf Weiter.
 9. Lesen Sie die Informationen unter Web Server Role (IIS) (Webserverrolle (IIS)). Klicken Sie anschließend auf Weiter.
 10. Übernehmen Sie für Select role services (Rollenservices auswählen) die Standardwerte oder ändern Sie die Einstellungen. Klicken Sie anschließend auf Weiter.
 11. Lesen Sie unter Confirmation (Bestätigung) die Informationen zur Bestätigung. Wählen Sie anschließend Installieren.
 12. Nach abgeschlossener Installation klicken Sie auf Close (Schließen).

Nachdem Sie diese Schritte abgeschlossen haben, fahren Sie mit [Schritt 2: Erstellen einer Zertifikatsignierungsanforderung \(CSR\) und eines Zertifikats](#) fort.

Voraussetzungen für das Client-SDK 3

Um das SSL/TLS Webserver-Offload mit einzurichten AWS CloudHSM, benötigen Sie Folgendes:

- Ein aktiver AWS CloudHSM Cluster mit mindestens einem HSM.
- Eine Amazon EC2-Instance, auf der ein Windows-Betriebssystem ausgeführt wird und die folgende Software installiert ist:
 - Die AWS CloudHSM Client-Software für Windows.
 - Internet Information Services (IIS) für Windows Server
- Ein [Crypto-Benutzer](#) (CU), der den privaten Schlüssel des Webserver auf dem HSM besitzen und verwalten soll.

Note

In diesem Tutorial wird Microsoft Windows Server 2016 verwendet. Microsoft Windows Server 2012 wird ebenfalls unterstützt, Microsoft Windows Server 2012 R2 jedoch nicht.

So richten Sie eine Windows-Webserver-Instance auf dem HSM ein und erstellen einen CU

1. Führen Sie die Schritte unter [Erste Schritte](#) aus. Beim Starten des Amazon EC2-Clients wählen Sie ein Windows Server 2016- oder Windows Server 2012-AMI aus. Wenn Sie diese Schritte ausgeführt haben, verfügen Sie über einen aktiven Cluster mit mindestens einem HSM. Sie haben auch eine Amazon EC2 EC2-Client-Instance, auf der Windows Server ausgeführt wird und auf der die AWS CloudHSM Client-Software für Windows installiert ist.
2. (Optional) Fügen Sie Ihrem Cluster weitere HSMs hinzu. Weitere Informationen finden Sie unter [Hinzufügen eines HSM zu einem Cluster AWS CloudHSM](#).
3. Stellen Sie eine Verbindung zu Ihrem Windows-Server her. Weitere Informationen finden Sie unter [Connect to Your Instance](#) im Amazon EC2 EC2-Benutzerhandbuch.
4. Verwenden Sie die CloudHSM-CLI, um einen Crypto-Benutzer (CU) zu erstellen. Merken Sie sich den CU-Benutzernamen und das Passwort. Sie benötigen diese im nächsten Schritt.

 Note

Informationen zum Erstellen eines Benutzers finden Sie unter [HSM-Benutzer mit der CloudHSM-CLI verwalten](#).

5. [Legen Sie die Anmeldeinformationen für das HSM fest](#). Verwenden Sie hierfür den im vorherigen Schritt erstellten CU-Benutzernamen und das entsprechende Passwort.
6. Wenn Sie in Schritt 5 den Windows Credentials Manager zum Einrichten [psexec.exe](#) von HSM-Anmeldeinformationen verwendet haben, laden Sie den folgenden Befehl als NT Authority\SYSTEM herunter, SysInternals um ihn auszuführen:

```
psexec.exe -s "C:\Program Files\Amazon\CloudHsm\tools\set_cloudhsm_credentials.exe"  
--username <USERNAME> --password <PASSWORD>
```

Ersetzen Sie **<USERNAME>** und **<PASSWORD>** durch die HSM-Anmeldeinformationen.

So installieren Sie IIS auf Ihrem Windows-Server

1. Stellen Sie eine Verbindung mit Ihrem Windows-Server her, sofern noch nicht geschehen. Weitere Informationen finden Sie unter [Connect to Your Instance](#) im Amazon EC2 EC2-Benutzerhandbuch.
2. Starten Sie auf Ihrem Windows-Server Server Manager.

3. Wählen Sie im Dashboard Server-Manager die Option Rollen und Features hinzufügen.
4. Lesen Sie die Informationen unter Before you begin (Zur Vorbereitung) und klicken Sie dann auf Weiter.
5. Wählen Sie unter Installationsart die Rollenbasierte oder funktionsbasierte Installation. Wählen Sie anschließend Weiter.
6. Wählen Sie unter Serverauswahl die Option Einen Server aus dem Serverpool auswählen. Klicken Sie anschließend auf Weiter.
7. Gehen Sie für Serverrollen wie folgt vor:
 - a. Wählen Sie Web Server (IIS) (Webserver (IIS)) aus.
 - b. Klicken Sie für Add features that are required für Web Server (IIS) (Für Webserver (IIS)) erforderliche Funktionen hinzufügen) auf Add Features (Funktionen hinzufügen).
 - c. Klicken Sie auf Weiter, um das Auswählen von Serverrollen abzuschließen.
8. Für Features (Funktionen), übernehmen Sie die Standardeinstellungen. Klicken Sie anschließend auf Weiter.
9. Lesen Sie die Informationen unter Web Server Role (IIS) (Webserverrolle (IIS)). Klicken Sie anschließend auf Weiter.
10. Übernehmen Sie für Select role services (Rollenservices auswählen) die Standardwerte oder ändern Sie die Einstellungen. Klicken Sie anschließend auf Weiter.
11. Lesen Sie unter Confirmation (Bestätigung) die Informationen zur Bestätigung. Wählen Sie anschließend Installieren.
12. Nach abgeschlossener Installation klicken Sie auf Close (Schließen).

Nachdem Sie diese Schritte abgeschlossen haben, fahren Sie mit [Schritt 2: Erstellen einer Zertifikatsignierungsanforderung \(CSR\) und eines Zertifikats](#) fort.

Schritt 2: Erstellen einer Zertifikatsignierungsanforderung (CSR) und eines Zertifikats

Um HTTPS zu aktivieren, benötigt Ihr Webserver ein SSL/TLS Zertifikat und einen entsprechenden privaten Schlüssel. Um SSL/TLS Offload mit zu verwenden AWS CloudHSM, speichern Sie den privaten Schlüssel im HSM in Ihrem AWS CloudHSM Cluster. Verwenden Sie dazu den [AWS CloudHSM -Schlüsselspeicher-Anbieter \(Key Storage Provider, KSP\) für die Microsoft Cryptography API: Next Generation \(CNG\)](#), um eine Zertifikatsignierungsanforderung (CSR) zu erstellen. Senden Sie die CSR dann an eine Zertifizierungsstelle (Certificate Authority, CA), um die CSR signieren zu lassen und ein Zertifikat zu erhalten.

Themen

- [Erstellen Sie eine CSR mit Client SDK 5](#)
- [Erstellen Sie eine CSR mit dem Client SDK 3](#)
- [Erhalten und Importieren eines signierten Zertifikats](#)

Erstellen Sie eine CSR mit Client SDK 5

1. Verwenden Sie auf Ihrem Windows-Server einen Texteditor, um eine Zertifikatanforderungsdatei namens `IISCertRequest.inf` zu erstellen. Den Inhalt einer Beispieldatei `IISCertRequest.inf` sehen Sie nachfolgend. Weitere Informationen zu den Abschnitten, Schlüsseln und Werten in dieser Datei finden Sie in der [Microsoft-Dokumentation](#). Ändern Sie den Wert `ProviderName` nicht.

```
[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. Verwenden Sie den [certreqWindows-Befehl](#), um eine CSR aus der `IISCertRequest.inf` Datei zu erstellen, die Sie im vorherigen Schritt erstellt haben. Im folgenden Beispiel wird die CSR in einer Datei namens `IISCertRequest.csr` gespeichert. Wenn Sie für Ihre Zertifikatsanforderungsdatei einen anderen Dateinamen verwendet haben, ***IISCertRequest.inf*** ersetzen Sie ihn durch den entsprechenden Dateinamen. Sie können Ihre CSR-Datei optional durch einen anderen Dateinamen ersetzen ***IISCertRequest.csr***.

```
C:\>certreq -new IISCertRequest.inf IISCertRequest.csr
```

```
CertReq: Request Created
```

Die Datei `IISCertRequest.csr` enthält Ihre CSR. Sie benötigen diese CSR, um ein signiertes Zertifikat zu erhalten.

Erstellen Sie eine CSR mit dem Client SDK 3

1. Stellen Sie eine Verbindung mit Ihrem Windows-Server her, sofern noch nicht geschehen. Weitere Informationen finden Sie unter [Connect to Your Instance](#) im Amazon EC2 EC2-Benutzerhandbuch.
2. Verwenden Sie den folgenden Befehl, um den AWS CloudHSM Client-Daemon zu starten.

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

- Für Windows-Client 1.1.2 und höher:

```
C:\Program Files\Amazon\CloudHSM>net.exe start AWSCloudHSMClient
```

- Für Windows-Clients 1.1.1 und früher:

```
C:\Program Files\Amazon\CloudHSM>start "cloudhsm_client" cloudhsm_client.exe  
C:\ProgramData\Amazon\CloudHSM\data\cloudhsm_client.cfg
```

3. Verwenden Sie auf Ihrem Windows-Server einen Texteditor, um eine Zertifikatanforderungsdatei namens `IISCertRequest.inf` zu erstellen. Den Inhalt einer Beispieldatei `IISCertRequest.inf` sehen Sie nachfolgend. Weitere Informationen zu den Abschnitten, Schlüsseln und Werten in dieser Datei finden Sie in der [Microsoft-Dokumentation](#). Ändern Sie den Wert `ProviderName` nicht.

```
[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. Verwenden Sie den [certreqWindows-Befehl](#), um eine CSR aus der `IISCertRequest.inf` Datei zu erstellen, die Sie im vorherigen Schritt erstellt haben. Im folgenden Beispiel

wird die CSR in einer Datei namens `IISCertRequest.csr` gespeichert. Wenn Sie für Ihre Zertifikatsanforderungsdatei einen anderen Dateinamen verwendet haben, *IISCertRequest.inf* ersetzen Sie ihn durch den entsprechenden Dateinamen. Sie können Ihre CSR-Datei optional durch einen anderen Dateinamen ersetzen *IISCertRequest.csr*.

```
C:\>certreq -new IISCertRequest.inf IISCertRequest.csr
      SDK Version: 2.03

CertReq: Request Created
```

Die Datei `IISCertRequest.csr` enthält Ihre CSR. Sie benötigen diese CSR, um ein signiertes Zertifikat zu erhalten.

Erhalten und Importieren eines signierten Zertifikats

In einer Produktionsumgebung verwenden Sie in der Regel eine Zertifikatsstelle (CA) zum Erstellen eines Zertifikats aus einer CSR. Für eine Testumgebung ist keine CA erforderlich. Wenn Sie eine Zertifizierungsstelle verwenden, senden Sie die CSR-Datei (`IISCertRequest.csr`) an diese und verwenden Sie die Zertifizierungsstelle, um ein signiertes SSL/TLS Zertifikat zu erstellen.

Als Alternative zur Verwendung einer Zertifizierungsstelle können Sie ein Tool wie [OpenSSL](#) verwenden, um ein selbstsigniertes Zertifikat zu erstellen.

Warning

Selbstsignierte Zertifikate sind nicht vertrauenswürdig für Browser und sollten in Produktionsumgebungen nicht verwendet werden. Sie können in Testumgebungen verwendet werden.

Nachfolgend wird beschrieben, wie Sie ein selbstsigniertes Zertifikat erstellen und dieses für die Signierung der CSR Ihres Webservers verwenden.

So erstellen Sie ein selbstsigniertes Zertifikat

1. Verwenden Sie den folgenden OpenSSL-Befehl, um einen privaten Schlüssel zu erstellen. Sie können es optional durch den Dateinamen *SelfSignedCA.key* ersetzen, der Ihren privaten Schlüssel enthält.


```

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. Verwenden Sie den folgenden OpenSSL-Befehl, um ein selbstsigniertes Zertifikat mit dem privaten Schlüssel zu erstellen, den Sie im vorherigen Schritt erstellt haben. Hierbei handelt es sich um einen interaktiven Befehl. Lesen Sie die Anweisungen auf dem Bildschirm und befolgen Sie die Eingabeaufforderungen. *SelfSignedCA.key* Ersetzen Sie ihn durch den Namen der Datei, die Ihren privaten Schlüssel enthält (falls anders). Sie können es optional durch den Dateinamen *SelfSignedCA.crt* ersetzen, um Ihr selbstsigniertes Zertifikat zu enthalten.

```

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 []:

```

So verwenden Sie ein selbstsigniertes Zertifikat, um die CSR Ihres Webservers zu signieren

- Verwenden Sie den folgenden OpenSSL-Befehl, um mit Ihrem privaten Schlüssel und Ihrem selbstsignierten Zertifikat die CSR zu signieren. Ersetzen Sie die folgenden Namen durch die Namen der Dateien, die die jeweiligen Daten enthalten, sofern diese abweichen.
 - *IISCertRequest.csr*— Der Name der Datei, die den CSR Ihres Webservers enthält

- *SelfSignedCA.crt*— Der Name der Datei, die Ihr selbstsigniertes Zertifikat enthält
- *SelfSignedCA.key*— Der Name der Datei, die Ihren privaten Schlüssel enthält
- *IISCert.crt*— Der Name der Datei, die das signierte Zertifikat Ihres Webserverns enthalten soll

```
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:
```

Nachdem Sie den vorherigen Schritt abgeschlossen haben, haben Sie ein signiertes Zertifikat für Ihren Webserver (*IISCert.crt*) sowie ein selbstsigniertes Zertifikat (*SelfSignedCA.crt*). Wenn Sie diese Dateien haben, gehen Sie zu [Schritt 3: Konfigurieren des Webserverns](#).

Schritt 3: Konfigurieren des Webserverns

Aktualisieren Sie die Konfiguration Ihrer IIS-Website, um das HTTPS-Zertifikat zu verwenden, das Sie am Ende des [vorherigen Schritts](#) erstellt haben. Damit ist die Einrichtung Ihrer Windows-Webserver-Software (IIS) für das SSL/TLS Auslagern mit abgeschlossen. AWS CloudHSM

Wenn Sie ein selbstsigniertes Zertifikat zum Signieren Ihrer CSR verwendet haben, müssen Sie zunächst das selbstsignierte Zertifikat in Windows unter „Vertrauenswürdige Stammzertifizierungsstellen“ importieren.

So importieren Sie Ihr selbstsigniertes Zertifikat in Windows unter „Vertrauenswürdige Stammzertifizierungsstellen“

1. Stellen Sie eine Verbindung mit Ihrem Windows-Server her, sofern noch nicht geschehen. Weitere Informationen finden Sie unter [Connect to Your Instance](#) im Amazon EC2 EC2-Benutzerhandbuch.
2. Kopieren Sie Ihr selbstsigniertes Zertifikat auf Ihren Windows-Server.
3. Öffnen Sie auf Ihrem Windows-Server die Systemsteuerung.

4. Geben Sie für Search Control Panel (Systemsteuerung durchsuchen) **certificates** ein. Klicken Sie dann auf Computerzertifikate verwalten.
5. Doppelklicken Sie im Fenster Zertifikate — Lokaler Computer auf Vertrauenswürdige Stammzertifizierungsstellen.
6. Klicken Sie mit der rechten Maustaste auf Zertifikate, Alle Aufgaben und Importieren.
7. Klicken Sie im Zertifikatimport-Assistent auf Weiter.
8. Klicken Sie auf Durchsuchen und wählen Sie Ihr selbstsigniertes Zertifikat aus. Wenn Sie Ihr selbstsigniertes Zertifikat nach der Anleitung im [vorherigen Schritt dieser Anleitung](#) erstellt haben, hat Ihr selbstsigniertes Zertifikat den Namen `SelfSignedCA.crt`. Klicken Sie auf Open.
9. Wählen Sie Weiter.
10. Klicken Sie für Zertifikatspeicher auf Alle Zertifikate in folgendem Speicher speichern. Stellen Sie dann sicher, dass für Zertifikatspeicher Vertrauenswürdige Stammzertifizierungsstellen ausgewählt ist.
11. Wählen Sie Weiter und anschließend Beenden aus.

So aktualisieren Sie die Konfiguration Ihrer IIS-Website

1. Stellen Sie eine Verbindung mit Ihrem Windows-Server her, sofern noch nicht geschehen. Weitere Informationen finden Sie unter [Connect to Your Instance](#) im Amazon EC2 EC2-Benutzerhandbuch.
2. Starten Sie den AWS CloudHSM Client-Daemon.
3. Kopieren Sie das signierte Zertifikat Ihres Webservers – dasjenige, das Sie am Ende [des vorherigen Schritts dieses Tutorials erstellt haben](#) – auf Ihren Windows-Server.
4. Verwenden Sie auf Ihrem Windows Server den [certreqWindows-Befehl](#), um das signierte Zertifikat zu akzeptieren, wie im folgenden Beispiel. **IISCert.crt** Ersetzen Sie es durch den Namen der Datei, die das signierte Zertifikat Ihres Webservers enthält.

```
C:\>certreq -accept IISCert.crt  
SDK Version: 2.03
```

5. Starten Sie auf Ihrem Windows-Server Server Manager.
6. Klicken Sie rechts oben im Dashboard Server-Manager auf Extras, Internet Information Services (IIS)-Manager.
7. Doppelklicken Sie im Fenster Internet Information Services (IIS)-Manager auf den Servernamen. Doppelklicken Sie dann auf Websites. Wählen Sie Ihre Website aus.

8. Klicken Sie auf SSL-Einstellungen. Klicken Sie dann rechts im Fenster auf Bindungen.
9. Klicken Sie im Fenster Websitebindungen auf Hinzufügen.
10. Wählen Sie für Typ die Option HTTPS aus. Wählen Sie für SSL-Zertifikat das HTTPS-Zertifikat aus, das Sie am Ende [des vorherigen Schritts dieser Anleitung](#) erstellt haben.

 Note

Wenn während dieser Zertifikat-Bindung ein Fehler auftritt, starten Sie Ihren Server neu und wiederholen Sie diesen Schritt.

11. Wählen Sie OK aus.

Nachdem Sie die Konfiguration Ihrer Website aktualisiert haben, gehen Sie zu [Schritt 4: Aktivieren von HTTPS-Datenverkehr und Verifizieren des Zertifikats](#).

Schritt 4: Aktivieren von HTTPS-Datenverkehr und Verifizieren des Zertifikats

Nachdem Sie Ihren Webserver für SSL/TLS Offload mit konfiguriert haben AWS CloudHSM, fügen Sie Ihre Webserver-Instanz einer Sicherheitsgruppe hinzu, die eingehenden HTTPS-Verkehr zulässt. Dadurch können Clients, wie z. B. Webbrowser, eine HTTPS-Verbindung mit Ihrem Webserver herstellen. Stellen Sie dann eine HTTPS-Verbindung zu Ihrem Webserver her und stellen Sie sicher, dass er das Zertifikat verwendet, mit dem Sie das SSL/TLS Offload konfiguriert haben. AWS CloudHSM

Themen

- [Aktivieren von eingehenden HTTPS-Verbindungen](#)
- [Verifizieren, dass HTTPS das konfigurierte Zertifikat verwendet](#)

Aktivieren von eingehenden HTTPS-Verbindungen

Zum Herstellen einer Verbindung zu Ihrem Webserver von einem Client (z. B. ein Webbrowser) aus, erstellen Sie eine Sicherheitsgruppe, die eingehende HTTPS-Verbindungen zulässt. Insbesondere sollten eingehende TCP-Verbindungen auf Port 443 erlaubt werden. Weisen Sie diese Sicherheitsgruppe Ihrem Webserver zu.

So erstellen Sie eine Sicherheitsgruppe für HTTPS und weisen sie Ihrem Webserver zu

1. Öffnen Sie die Amazon-EC2-Konsole unter <https://console.aws.amazon.com/ec2/>.

2. Wählen Sie im Navigationsbereich Sicherheitsgruppen aus.
3. Wählen Sie Sicherheitsgruppe erstellen.
4. Führen Sie für Sicherheitsgruppe erstellen die folgenden Schritte aus:
 - a. Geben Sie in das Feld Sicherheitsgruppenname einen Namen für die Sicherheitsgruppe ein, die Sie erstellen.
 - b. (Optional) Geben Sie eine Beschreibung der Sicherheitsgruppe ein, die Sie erstellen.
 - c. Wählen Sie für VPC die VPC aus, die Ihre Amazon-EC2-Instance enthält.
 - d. Wählen Sie Regel hinzufügen aus.
 - e. Wählen Sie im Drop-down-Fenster für Typ die Option HTTPS aus.
 - f. Geben Sie für Quelle einen Quellspeicherort ein.
 - g. Wählen Sie Sicherheitsgruppe erstellen aus.
5. Wählen Sie im Navigationsbereich Instances aus.
6. Aktivieren Sie das Kontrollkästchen neben Ihrer Webserver-Instance.
7. Wählen Sie das Drop-down-Menü Aktionen oben auf der Seite. Wählen Sie Sicherheit und dann Sicherheitsgruppen ändern aus.
8. Wählen Sie unter Zugeordnete Sicherheitsgruppen das Suchfeld aus und wählen Sie die Sicherheitsgruppe, die Sie für HTTPS erstellt haben, aus. Wählen Sie dann Sicherheitsgruppen hinzufügen aus.
9. Wählen Sie Speichern.

Verifizieren, dass HTTPS das konfigurierte Zertifikat verwendet

Nachdem Sie den Webserver zu einer Sicherheitsgruppe hinzugefügt haben, können Sie überprüfen, ob der SSL/TLS Offload Ihr selbstsigniertes Zertifikat verwendet. Sie können dazu einen Webbrowser oder ein Tool wie [OpenSSL s_client](#) nutzen.

Um den SSL/TLS Offload mit einem Webbrowser zu überprüfen

1. Verwenden Sie einen Web-Browser, um eine Verbindung zum Webserver unter Verwendung des öffentlichen DNS-Namen oder der IP-Adresse des Servers herzustellen. Stellen Sie sicher, dass die URL in die Adresszeile mit https:// beginnt. Beispiel, **https://ec2-52-14-212-67.us-east-2.compute.amazonaws.com/**.

 Tip

Sie können einen DNS-Dienst wie Amazon Route 53 verwenden, um den Domainnamen Ihrer Website (z. B. <https://www.example.com/>) an Ihren Webserver weiterzuleiten. Weitere Informationen finden Sie unter [Routing des Datenverkehrs zu einer Amazon-EC2-Instance](#) im Entwicklerleitfaden zu Amazon Route 53 oder in der Dokumentation für Ihren DNS-Service.

2. Zeigen Sie das Webserverzertifikat mit Ihrem Webbrowser an. Weitere Informationen finden Sie hier:
 - Wenn Sie Mozilla Firefox nutzen, sehen Sie sich die Informationen auf der Mozilla Support-Website unter [Zertifikat anzeigen](#) an.
 - Wenn Sie Google Chrome verwenden, sehen Sie sich die Informationen auf der „Google Tools für Web Developers“-Website unter [Sicherheitsprobleme verstehen](#) an.

Andere Webbrowser unterstützen möglicherweise ähnliche Funktionen, über die Sie das Webserverzertifikat anzeigen können.

3. Stellen Sie sicher, dass es sich bei dem SSL/TLS Zertifikat um das Zertifikat handelt, für dessen Verwendung Sie Ihren Webserver konfiguriert haben.

Um den SSL/TLS Offload mit OpenSSL `s_client` zu überprüfen

1. Führen Sie den folgenden OpenSSL-Befehl aus, um mittels HTTPS eine Verbindung zu Ihrem Webserver herzustellen. `<server name>` Ersetzen Sie es durch den öffentlichen DNS-Namen oder die IP-Adresse Ihres Webserver.

```
openssl s_client -connect <server name>:443
```

 Tip

Sie können einen DNS-Dienst wie Amazon Route 53 verwenden, um den Domainnamen Ihrer Website (z. B. <https://www.example.com/>) an Ihren Webserver weiterzuleiten. Weitere Informationen finden Sie unter [Routing des Datenverkehrs zu einer Amazon-](#)

[EC2-Instance](#) im Entwicklerleitfaden zu Amazon Route 53 oder in der Dokumentation für Ihren DNS-Service.

2. Stellen Sie sicher, dass es sich bei dem SSL/TLS Zertifikat um das Zertifikat handelt, für dessen Verwendung Sie Ihren Webserver konfiguriert haben.

Sie verfügen jetzt über eine mit HTTPS gesicherte Website. Der private Schlüssel für den Webserver wird in einem HSM in Ihrem AWS CloudHSM Cluster gespeichert.

Informationen zum Hinzufügen eines Load Balancers finden Sie unter [Einen Load Balancer mit Elastic Load Balancing hinzufügen für AWS CloudHSM\(optional\)](#).

Einen Load Balancer mit Elastic Load Balancing hinzufügen für AWS CloudHSM(optional)

Nachdem Sie SSL/TLS Offload mit einem Webserver eingerichtet haben, können Sie weitere Webserver und einen Elastic Load Balancing Load Balancer erstellen, der HTTPS-Verkehr an die Webserver weiterleitet. Ein Load Balancer kann die Auslastung Ihrer einzelnen Webserver reduzieren, indem der Verkehr auf zwei oder mehr Webserver verteilt wird. Er kann zudem die Verfügbarkeit Ihrer Website erhöhen, da er den Zustand Ihrer Webserver überwacht und den Datenverkehr nur an stabile Server weiterleitet. Wenn ein Webserver ausfällt, beendet der Load Balancer automatisch die Weiterleitung des Datenverkehrs an diesen.

Themen

- [Schritt 1. Erstellen eines Subnetzes für einen zweiten Webserver](#)
- [Schritt 2. Erstellen des zweiten Webserver](#)
- [Schritt 3. Erstellen Sie den Load Balancer](#)

Schritt 1. Erstellen eines Subnetzes für einen zweiten Webserver

Bevor Sie einen weiteren Webserver erstellen können, müssen Sie ein neues Subnetz in derselben VPC erstellen, die Ihren vorhandenen Webserver und AWS CloudHSM Cluster enthält.

So erstellen Sie ein neues Subnetz

1. Öffnen Sie den [Abschnitt Subnetze der Amazon VPC-Konsole](#).
2. Wählen Sie Create Subnet aus.

3. Führen Sie im Dialogfeld Create Subnet Folgendes aus:
 - a. Geben Sie unter Name tag(Namens-Tag) einen Namen für Ihr Subnetz ein.
 - b. Wählen Sie für VPC die AWS CloudHSM VPC aus, die Ihren vorhandenen Webserver und AWS CloudHSM Cluster enthält.
 - c. Wählen Sie als Availability Zone eine Availability Zone aus, die sich von der unterscheidet, die Ihren vorhandenen Webserver enthält.
 - d. Geben Sie unter IPv4 CIDR block den CIDR-Block für das Subnetz ein. Geben Sie beispielsweise **10.0.10.0/24** ein.
 - e. Wählen Sie Ja, erstellen aus.
4. Aktivieren Sie das Kontrollkästchen neben dem öffentlichen Subnetz, das Ihren vorhandenen Webserver enthält. Dieses unterscheidet sich von dem öffentlichen Subnetz, das Sie im vorherigen Schritt erstellt haben.
5. Klicken Sie im Inhaltsbereich auf die Registerkarte Routing-Tabelle. Wählen Sie dann den Link für die Routing-Tabelle aus.

subnet-1f358d78 | CloudHSM Public subnet

Destination	Target
10.0.0.0/16	local
0.0.0.0/0	igw-68ee440c

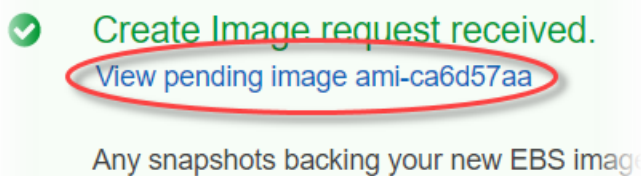
6. Aktivieren Sie das Kontrollkästchen neben der Routing-Tabelle.
7. Wählen Sie die Registerkarte Subnetz-Verknüpfungen aus. Wählen Sie dann Bearbeiten aus.
8. Wählen Sie das Kontrollkästchen neben dem öffentlichen Subnetz aus, das Sie zuvor erstellt haben. Wählen Sie dann Speichern.

Schritt 2. Erstellen des zweiten Webserver

Führen Sie die folgenden Schritte aus, um einen zweiten Webserver mit der gleichen Konfiguration wie der des vorhandenen Webserver zu erstellen.

So erstellen Sie einen zweiten Webserver

1. Öffnen Sie den Bereich [Instances](#) in der Amazon-EC2-Konsole unter.
2. Aktivieren Sie das Kontrollkästchen neben Ihrer vorhandenen Webserver-Instance.
3. Wählen Sie Aktionen, Image und dann Image erstellen.
4. Führen Sie im Dialogfeld Create Image (Abbild erstellen) die folgenden Schritte aus:
 - a. Geben Sie für Image name (Image-Name) einen eindeutigen Namen für das Image ein.
 - b. Geben Sie unter Image description (Image-Beschreibung) eine Beschreibung für das Image ein.
 - c. Wählen Sie Image erstellen aus. Mit dieser Aktion wird Ihr vorhandener Webserver neu gestartet.
 - d. Wählen Sie den Link „Ausstehendes Bild anzeigen“. **<AMI ID>**



Beachten Sie den Abbildstatus in der Status-Spalte. Lautet der Abbildstatus available (verfügbar), dies kann einige Minuten dauern, fahren Sie mit dem nächsten Schritt fort.

5. Wählen Sie im Navigationsbereich Instances aus.
6. Aktivieren Sie das Kontrollkästchen neben Ihrem vorhandenen Webserver.
7. Wählen Sie Aktionen und dann Weitere solche starten aus.
8. Wählen Sie AMI bearbeiten aus.

AMI Details



amzn-ami-hvm-2017.09.1.20171120-x86_64-gp2 - ami-a51f27c5

Amazon Linux AMI 2017.09.1.20171120 x86_64 HVM GP2

Root Device Type: ebs Virtualization type: hvm

[Edit AMI](#)

9. Wählen Sie im linken Navigationsbereich Meine aus AMIs. Löschen Sie anschließend den Text im Suchfeld.
10. Wählen Sie neben Ihrem Webserverabbild Auswählen aus.
11. Wählen Sie Ja, ich möchte mit diesem AMI fortfahren (**<image name>**- ami-**<AMI ID>**).
12. Wählen Sie Weiter aus.

13. Wählen Sie einen Instance-Typen und danach Weiter: Instance-Details konfigurieren aus.
14. Führen Sie für Step 3: Configure Instance Details (Schritt 3: Instance-Details konfigurieren) Folgendes aus:
 - a. Wählen Sie für Netzwerk die VPC mit Ihrem vorhandenen Webserver aus.
 - b. Wählen Sie für Subnetz das öffentliche Subnetz aus, das Sie für den zweiten Webserver erstellt haben.
 - c. Wählen Sie für Öffentliche IP automatisch zuweisen Aktivieren aus.
 - d. Ändern Sie die verbleibenden Instance-Details wie gewünscht. Wählen Sie dann Weiter: Speicher hinzufügen aus.
15. Ändern Sie die Speichereinstellungen wie gewünscht. Wählen Sie dann Weiter: Tags hinzufügen aus.
16. Bearbeiten Sie Tags nach Bedarf oder fügen Sie diese hinzu. Wählen Sie dann Als Nächstes: Sicherheitsgruppe konfigurieren aus.
17. Führen Sie für Schritt 6: Sicherheitsgruppe konfigurieren die folgenden Schritte aus:
 - a. Wählen Sie für Sicherheitsgruppe zuweisen die Option Existierende Sicherheitsgruppe auswählen aus.
 - b. Aktivieren Sie das Kontrollkästchen neben der Sicherheitsgruppe mit dem Namen cloudhsm-**<cluster ID>** -sg. AWS CloudHSM hat diese Sicherheitsgruppe in Ihrem Namen erstellt, als Sie den Cluster [erstellt](#) haben. Sie müssen diese Sicherheitsgruppe auswählen, damit die Webserver-Instanz eine Verbindung mit der HSMs im Cluster herstellen kann.
 - c. Aktivieren Sie das Kontrollkästchen neben der Sicherheitsgruppe, die eingehenden HTTPS-Datenverkehr zulässt. Sie [haben diese Sicherheitsgruppe zuvor erstellt](#).
 - d. (Optional) Aktivieren Sie das Kontrollkästchen neben einer Sicherheitsgruppe, die eingehenden SSH- (für Linux) oder RDP-Datenverkehr (für Windows) aus Ihrem Netzwerk zulässt. Das heißt, die Sicherheitsgruppe muss ein- und ausgehenden TCP-Datenverkehr auf Port 22 (für SSH unter Linux) bzw. Port 3389 (für RDP unter Windows) zulassen. Andernfalls können Sie keine Verbindung zu Ihrer Client-Instanz herstellen. Wenn Sie keine solche Sicherheitsgruppe haben, müssen Sie sie erstellen und Ihrer Client-Instanz zu einem späteren Zeitpunkt hinzufügen.

Klicken Sie auf Review and Launch.

18. Überprüfen Sie die Instance-Details. Wählen Sie anschließend Start aus.

19. Legen Sie fest, ob Ihre Instance mit einem vorhandenen Schlüsselpaar oder ohne Schlüsselpaar gestartet werden soll, oder ob ein neues Schlüsselpaar erstellt werden soll.

- Um ein vorhandenes Schlüsselpaar zu verwenden, führen Sie die folgenden Schritte aus:
 1. Wählen Sie Vorhandenes Schlüsselpaar auswählen aus.
 2. Wählen Sie für Schlüsselpaar auswählen das zu verwendende Schlüsselpaar aus.
 3. Aktivieren Sie das Kontrollkästchen neben Ich bestätige, dass ich Zugriff auf die ausgewählte private Schlüsseldatei (**<private key file name>.pem**) habe und dass ich mich ohne diese Datei nicht bei meiner Instance anmelden kann.
- Wenn Sie ein neues Schlüsselpaar erstellen möchten, führen Sie die folgenden Schritte aus:
 1. Wählen Sie Ein neues Schlüsselpaar erstellen aus.
 2. Geben Sie für Key pair name (Schlüsselpaarname) einen Schlüsselpaarnamen ein.
 3. Wählen Sie Schlüsselpaar herunterladen aus und speichern Sie die Datei mit dem privaten Schlüssel an einem sicheren, zugänglichen Ort.

 Warning

Nach diesem Zeitpunkt können Sie die Datei mit dem privaten Schlüssel nicht erneut herunterladen. Wenn Sie die Datei mit dem privaten Schlüssel jetzt nicht herunterzuladen, können Sie sich auf die Client-Instance zugreifen.

- Zum Starten Ihrer Instance ohne Schlüsselpaar führen Sie die folgenden Schritte aus:
 1. Wählen Sie Ohne Schlüsselpaar fortfahren aus.
 2. Aktivieren Sie das Kontrollkästchen neben Ich bestätige, dass ich mich nicht bei dieser Instance anmelden kann, wenn ich das in dieses AMI integrierte Passwort nicht kenne.

Wählen Sie Instances starten aus.

Schritt 3. Erstellen Sie den Load Balancer

Führen Sie die folgenden Schritte aus, um einen Elastic Load Balancing Load Balancer zu erstellen, der den HTTPS-Datenverkehr an Ihre Webserver weiterleitet.

So erstellen Sie einen Load Balancer

1. Öffnen Sie den Bereich [Load Balancers](#) in der Amazon EC2-Konsole.

Load Balancer hinzufügen (optional)

2. Klicken Sie auf Load Balancer erstellen.
3. Klicken Sie im Bereich Network Load Balancer auf Create.
4. Führen Sie für Step 1: Configure Load Balancer (Schritt 1; Konfigurieren von Load Balancer) die folgenden Schritte aus:
 - a. Geben Sie als Name einen Namen für den Load Balancer ein, den Sie erstellen.
 - b. Ändern Sie im Bereich Listeners den Wert in **443** für Load Balancer Port.
 - c. Wählen Sie im Bereich Availability Zones als VPC die VPC aus, die Ihre Webserver enthält.
 - d. Wählen Sie im Bereich Availability Zones die Subnetze aus, die Ihre Webserver enthalten.
 - e. Wählen Sie Weiter: Routing konfigurieren aus.
5. Führen Sie für Step 2: Configure Routing (Schritt 2: Routing konfigurieren) die folgenden Schritte aus:
 - a. Geben Sie als Name einen Namen für die Zielgruppe ein, die Sie erstellen.
 - b. Ändern Sie für den Port den Wert in **443**.
 - c. Klicken Sie auf Weiter: Ziele registrieren.
6. Führen Sie für Step 3: Register Targets (Schritt 3: Ziele registrieren) die folgenden Schritte aus:
 - a. Aktivieren Sie im Bereich Instances die Kontrollkästchen neben Ihren Webserver-Instances. Wählen Sie dann Zu registrierten hinzufügen aus.
 - b. Wählen Sie Weiter: Prüfen aus.
7. Überprüfen Sie die Load Balancer-Details und wählen Sie dann Erstellen aus.
8. Wenn der Load Balancer erfolgreich erstellt wurde, klicken Sie auf Close (Schließen).

Nachdem Sie die vorangegangenen Schritte durchgeführt haben, zeigt die Amazon EC2-Konsole Ihren Elastic Load-Balancing Load Balancer an.

Wenn der Status Ihres Load Balancers „aktiv“ ist, können Sie verifizieren, dass der Load Balancer funktioniert. Das heißt, Sie können mit SSL/TLS Offload with überprüfen, ob HTTPS-Verkehr an Ihre Webserver gesendet wird. AWS CloudHSM Sie können dazu einen Webbrowser oder ein Tool wie [OpenSSL s_client](#) verwenden.

So stellen Sie sicher, dass der Load Balancer mit einem Webbrowser arbeitet

1. Suchen Sie in der Amazon EC2-Konsole den DNS-Namen für den Load Balancer, den Sie gerade erstellt haben. Wählen Sie anschließend den DNS-Namen aus und kopieren Sie ihn.

2. Verwenden Sie einen Webbrowser wie Mozilla Firefox oder Google Chrome für die Verbindung Ihres Load Balancers über dessen DNS-Namen. Stellen Sie sicher, dass die URL in die Adresszeile mit `https://` beginnt.

 Tip

Sie können einen DNS-Dienst wie Amazon Route 53 verwenden, um den Domainnamen Ihrer Website (z. B. `https://www.example.com/`) an Ihren Webserver weiterzuleiten. Weitere Informationen finden Sie unter [Routing des Datenverkehrs zu einer Amazon-EC2-Instance](#) im Entwicklerleitfaden zu Amazon Route 53 oder in der Dokumentation für Ihren DNS-Service.

3. Zeigen Sie das Webserverzertifikat mit Ihrem Webbrowser an. Weitere Informationen finden Sie hier:
 - Wenn Sie Mozilla Firefox nutzen, sehen Sie sich die Informationen auf der Mozilla Support-Website unter [Zertifikat anzeigen](#) an.
 - Wenn Sie Google Chrome verwenden, sehen Sie sich die Informationen auf der „Google Tools für Web Developers“-Website unter [Sicherheitsprobleme verstehen](#) an.

Andere Webbrowser unterstützen möglicherweise ähnliche Funktionen, über die Sie das Webserverzertifikat anzeigen können.

4. Stellen Sie sicher, dass das Zertifikat dasjenige ist, das Sie für die Nutzung im Webserver konfiguriert haben.

So stellen Sie sicher, dass der Load Balancer mit OpenSSL `s_client` arbeitet

1. Verbinden Sie mit dem folgenden OpenSSL-Befehl Ihrer Load Balancer mittels HTTPS. *<DNS name>* Ersetzen Sie ihn durch den DNS-Namen Ihres Load Balancers.

```
openssl s_client -connect <DNS name>:443
```

 Tip

Sie können einen DNS-Dienst wie Amazon Route 53 verwenden, um den Domainnamen Ihrer Website (z. B. `https://www.example.com/`) an Ihren Webserver weiterzuleiten. Weitere Informationen finden Sie unter [Routing des Datenverkehrs zu einer Amazon-](#)

[EC2-Instance](#) im Entwicklerleitfaden zu Amazon Route 53 oder in der Dokumentation für Ihren DNS-Service.

2. Stellen Sie sicher, dass das Zertifikat dasjenige ist, das Sie für die Nutzung im Webserver konfiguriert haben.

Sie haben jetzt eine Website, die mit HTTPS gesichert ist, wobei der private Schlüssel des Webserver in einem HSM in Ihrem AWS CloudHSM Cluster gespeichert ist. Ihre Website verfügt über zwei Webserver und einen Load Balancer, um Effizienz und Verfügbarkeit zu verbessern.

Konfigurieren Sie Windows Server als Zertifizierungsstelle (CA) mit AWS CloudHSM

AWS CloudHSM bietet Unterstützung für die Konfiguration von Windows Server als Zertifizierungsstelle (CA) über Client SDK 3 und Client SDK 5. Die Schritte zur Verwendung dieser Tools hängen von der Version des Client-SDK ab, die Sie derzeit heruntergeladen haben. Die folgenden Abschnitte enthalten Informationen zu den einzelnen SDKs.

Themen

- [Konfigurieren Sie Windows Server als Zertifizierungsstelle \(CA\) mit Client SDK 5](#)
- [Konfigurieren Sie Windows Server als Zertifizierungsstelle \(CA\) mit Client SDK 3](#)

Konfigurieren Sie Windows Server als Zertifizierungsstelle (CA) mit Client SDK 5

In einer Public Key-Infrastruktur (PKI) ist eine Zertifizierungsstelle (CA) eine vertrauenswürdige Entität, die digitale Zertifikate ausstellt. Diese digitalen Zertifikate binden einen öffentlichen Schlüssel unter Verwendung von Kryptografie öffentlicher Schlüssel und von digitalen Signaturen an eine Identität (eine Person oder Organisation). Um eine CA zu betreiben, müssen Sie das Vertrauen aufrechterhalten, indem Sie den privaten Schlüssel schützen, der die von Ihrer CA ausgestellten Zertifikate signiert. Sie können den privaten Schlüssel im HSM in Ihrem AWS CloudHSM -Cluster speichern und das HSM zur Durchführung der kryptografischen Signiervorgänge verwenden.

In diesem Tutorial verwenden Sie Windows Server und AWS CloudHSM konfigurieren eine Zertifizierungsstelle. Nach der Installation der AWS CloudHSM -Client-Software für Windows auf Ihrem Windows Server fügen Sie Ihrem Windows Server die Active Directory-Zertifikatsdienste

(AD CS)-Rolle hinzu. Wenn Sie diese Rolle konfigurieren, verwenden Sie einen AWS CloudHSM Key Storage Provider (KSP), um den privaten Schlüssel der CA zu erstellen und auf Ihrem AWS CloudHSM Cluster zu speichern. Der KSP ist die Brücke, die Ihren Windows-Server mit Ihrem Cluster verbindet. AWS CloudHSM Im letzten Schritt signieren Sie mit Ihrer Windows Server-CA eine Zertifikatssignierungsanforderung (Certificate Signing Request, CSR).

Weitere Informationen finden Sie unter den folgenden Themen:

Themen

- [Schritt 1: Einrichten der Voraussetzungen](#)
- [Schritt 2: Erstellen Sie eine Windows Server-Zertifizierungsstelle mit AWS CloudHSM](#)
- [Schritt 3: Signieren Sie eine Zertifikatssignierungsanforderung \(CSR\) mit Ihrer Windows Server-Zertifizierungsstelle mit AWS CloudHSM](#)

Schritt 1: Einrichten der Voraussetzungen

Um Windows Server als Zertifizierungsstelle (CA) mit einzurichten AWS CloudHSM, benötigen Sie Folgendes:

- Ein aktiver AWS CloudHSM Cluster mit mindestens einem HSM.
- Eine Amazon EC2 EC2-Instance, auf der ein Windows Server-Betriebssystem mit installierter AWS CloudHSM Client-Software für Windows ausgeführt wird. In diesem Tutorial wird Microsoft Windows Server 2016 verwendet.
- Einen Crypto-Benutzer (CU), der den privaten Schlüssel der CA auf dem HSM besitzen und verwalten soll.

Um die Voraussetzungen für eine Windows Server-CA einzurichten mit AWS CloudHSM

1. Führen Sie die Schritte unter [Erste Schritte](#) aus. Beim Starten der Amazon EC2-Clients wählen Sie ein Windows Server-AMI aus. In diesem Tutorial wird Microsoft Windows Server 2016 verwendet. Wenn Sie diese Schritte ausgeführt haben, verfügen Sie über einen aktiven Cluster mit mindestens einem HSM. Sie haben auch eine Amazon EC2 EC2-Client-Instance, auf der Windows Server ausgeführt wird und auf der die AWS CloudHSM Client-Software für Windows installiert ist.
2. (Optional) Fügen Sie Ihrem Cluster weitere HSMs hinzu. Weitere Informationen finden Sie unter [Hinzufügen eines HSM zu einem Cluster AWS CloudHSM](#).

3. Stellen Sie eine Verbindung mit Ihrer Client-Instance her. Weitere Informationen finden Sie unter [Connect to Your Instance](#) im Amazon EC2 EC2-Benutzerhandbuch.
4. Erstellen Sie einen Crypto-Benutzer (CU) mithilfe [von Managing HSM users with CloudHSM CLI oder Managing HSM users with CloudHSM](#) Management Utility (CMU). Merken Sie sich den CU-Benutzernamen und das Passwort. Sie benötigen diese im nächsten Schritt.
5. [Legen Sie die Anmeldeinformationen für das HSM fest](#). Verwenden Sie hierfür den im vorherigen Schritt erstellten CU-Benutzernamen und das entsprechende Passwort.
6. Wenn Sie in Schritt 5 den Windows Credentials Manager zum Einrichten von HSM-Anmeldeinformationen verwendet haben, laden Sie den folgenden Befehl als NT Authority\ [psexec.exe](#)SYSTEM herunter, SysInternals um ihn auszuführen:

```
psexec.exe -s "C:\Program Files\Amazon\CloudHsm\tools\set_cloudhsm_credentials.exe"  
--username <USERNAME> --password <PASSWORD>
```

Ersetzen Sie **<USERNAME>** und **<PASSWORD>** durch die HSM-Anmeldeinformationen.

Um eine Windows Server-Zertifizierungsstelle mit zu erstellen AWS CloudHSM, gehen Sie zu [Erstellen einer Windows Server-CA](#).

Schritt 2: Erstellen Sie eine Windows Server-Zertifizierungsstelle mit AWS CloudHSM

Um eine Windows Server-CA zu erstellen, fügen Sie Ihrem Windows Server die Active Directory-Zertifikatsdienste (Active Directory Certificate Services, AD CS)-Rolle hinzu. Wenn Sie diese Rolle hinzufügen, verwenden Sie einen AWS CloudHSM Schlüsselspeicheranbieter (KSP), um den privaten Schlüssel der Zertifizierungsstelle zu erstellen und auf Ihrem AWS CloudHSM Cluster zu speichern.

Note

Wenn Sie Ihre Windows Server-CA erstellen, können Sie sich für das Erstellen einer Stamm-Zertifizierungsstelle oder einer untergeordneten Zertifizierungsstelle entscheiden. Sie treffen diese Entscheidung in der Regel basierend auf dem Entwurf Ihrer Public Key-Infrastruktur und der Sicherheitsrichtlinien Ihrer Organisation. In diesem Tutorial wird der Einfachheit halber erklärt, wie eine Stamm-CA erstellt wird.

So fügen Sie Ihrem Windows Server die AD-CS-Rolle hinzu und erstellen den privaten Schlüssel der CA

1. Stellen Sie eine Verbindung mit Ihrem Windows-Server her, sofern noch nicht geschehen. Weitere Informationen finden Sie unter [Connect to Your Instance](#) im Amazon EC2 EC2-Benutzerhandbuch.
2. Starten Sie auf Ihrem Windows-Server Server Manager.
3. Wählen Sie im Dashboard Server-Manager die Option Rollen und Features hinzufügen.
4. Lesen Sie die Informationen unter Before you begin (Zur Vorbereitung) und klicken Sie dann auf Weiter.
5. Wählen Sie unter Installationsart die Rollenbasierte oder funktionsbasierte Installation. Wählen Sie anschließend Weiter.
6. Wählen Sie unter Serverauswahl die Option Einen Server aus dem Serverpool auswählen. Klicken Sie anschließend auf Weiter.
7. Gehen Sie für Serverrollen wie folgt vor:
 - a. Wählen Sie Active Directory-Zertifikatsdienste.
 - b. Wählen Sie für Sollen für Active Directory-Zertifikatsdienste erforderliche Features hinzugefügt werden? die Option Features hinzufügen.
 - c. Klicken Sie auf Weiter, um das Auswählen von Serverrollen abzuschließen.
8. Übernehmen Sie für Features die Standardwerte und klicken Sie dann auf Weiter.
9. Verfahren Sie für AD CS wie folgt:
 - a. Wählen Sie Weiter aus.
 - b. Wählen Sie Zertifizierungsstelle und klicken Sie dann auf Weiter.
10. Lesen Sie unter Bestätigung die Informationen zur Bestätigung und klicken Sie dann auf Installieren. Schließen Sie das Fenster nicht.
11. Wählen Sie den hervorgehobenen Link Active Directory-Zertifikatsdienste auf dem Zielserver konfigurieren.
12. Prüfen oder ändern Sie die Anmeldeinformationen unter Anmeldeinformationen. Klicken Sie anschließend auf Weiter.
13. Wählen Sie für Rollendienste die Option Zertifizierungsstelle. Klicken Sie anschließend auf Weiter.

14. Wählen Sie für Installationsart die Option Eigenständige Zertifizierungsstelle. Klicken Sie anschließend auf Weiter.
15. Wählen Sie für Zertifizierungsstellentyp die Option Stammzertifizierungsstelle. Klicken Sie anschließend auf Weiter.

 Note

Sie können basierend auf dem Entwurf Ihrer Public Key-Infrastruktur und der Sicherheitsrichtlinien Ihrer Organisation entscheiden, ob eine Stammzertifizierungsstelle oder eine untergeordnete Zertifizierungsstelle erstellt werden soll. In diesem Tutorial wird der Einfachheit halber erklärt, wie eine Stamm-CA erstellt wird.

16. Wählen Sie für Privater Schlüssel die Option Neuen privaten Schlüssel erstellen. Klicken Sie anschließend auf Weiter.
17. Führen Sie für Kryptografie einen der folgenden Schritte aus:
 - a. Wählen Sie für Select a cryptographic provider eine der CloudHSM Key Storage Provider-Optionen aus dem Menü aus. Dies sind die AWS CloudHSM -Schlüsselspeicher-Provider. Sie können beispielsweise RSA #CloudHSM Key Storage Provider wählen.
 - b. Wählen Sie für Schlüssellänge eine der verfügbaren Schlüssellängen aus.
 - c. Wählen Sie für Select the hash algorithm für signing certificates issued by this CA (Wählen Sie den Hashalgorithmus zum Signieren von Zertifikaten aus, die von dieser Zertifizierungsstelle ausgestellt wurden) einen der verfügbaren Hashalgorithmen aus.

Wählen Sie Weiter.

18. Führen Sie für Zertifizierungsstelle einen der folgenden Schritte aus:
 - a. (Optional) Bearbeiten Sie den allgemeinen Namen.
 - b. (Optional) Geben Sie ein spezifisches Namenssuffix ein.

Wählen Sie Weiter.

19. Geben Sie für Gültigkeitsdauer einen Zeitraum in Jahren, Monaten, Wochen oder Tagen ein. Klicken Sie anschließend auf Weiter.

20. Für Zertifikatdatenbank können Sie die Standardwerte übernehmen. Sie können optional den Speicherort für die Datenbank und das Datenbankprotokoll ändern. Klicken Sie anschließend auf Weiter.
21. Überprüfen Sie die Informationen zur Zertifizierungsstelle unter Bestätigung. Klicken Sie dann auf Konfigurieren.
22. Klicken Sie auf Schließen und dann erneut auf Schließen.

Sie haben jetzt eine Windows Server-Zertifizierungsstelle mit AWS CloudHSM. Weitere Informationen zum Signieren einer Zertifikatssignierungsanforderung (Certificate Signing Request, CSR) mit Ihrer CA finden Sie unter [Signieren einer CSR](#).

Schritt 3: Signieren Sie eine Zertifikatssignieranforderung (CSR) mit Ihrer Windows Server-Zertifizierungsstelle mit AWS CloudHSM

Sie können Ihre Windows Server-Zertifizierungsstelle verwenden, AWS CloudHSM um eine Zertifikatssignieranforderung (CSR) zu signieren. Um diese Schritte ausführen zu können, benötigen Sie eine gültige CSR. Es gibt mehrere Möglichkeiten, eine CSR zu erstellen, einschließlich der Folgenden:

- Verwenden von OpenSSL
- Verwenden des Windows Server Internet Information Services (IIS) Manager
- Verwenden des Zertifikate-Snap-In in der Microsoft Management Console
- Verwenden des Befehlszeilendienstprogramms certreq unter Windows

Die Schritte zum Erstellen einer CSR werden in dieser Anleitung allerdings nicht behandelt. Wenn Sie über eine CSR verfügen, können Sie sie mit Ihrer Windows Server-Zertifizierungsstelle signieren.

So signieren Sie eine CSR mit Ihrer Windows Server-CA

1. Stellen Sie eine Verbindung mit Ihrem Windows-Server her, sofern noch nicht geschehen. Weitere Informationen finden Sie unter [Connect to Your Instance](#) im Amazon EC2 EC2-Benutzerhandbuch.
2. Starten Sie auf Ihrem Windows-Server Server Manager.
3. Klicken Sie rechts oben im Dashboard Server-Manager auf Extras, Zertifizierungsstelle.
4. Wählen Sie im Fenster Zertifizierungsstelle den Namen Ihres Computers aus.

5. Wählen Sie im Menü Aktion die Option Alle Aufgaben, Neue Anforderung einreichen.
6. Wählen Sie Ihre CSR-Datei aus und klicken Sie auf Öffnen.
7. Doppelklicken Sie im Fenster Zertifizierungsstelle auf Ausstehende Anforderungen.
8. Wählen Sie die ausstehende Anforderung aus. Wählen Sie dann im Menü Aktion die Option Alle Aufgaben, Ausstellen.
9. Doppelklicken Sie im Fenster Zertifizierungsstelle auf Ausgestellte Anforderungen, um das signierte Zertifikat anzuzeigen.
10. (Optional) Um das signierte Zertifikat in eine Datei zu exportieren, führen Sie die folgenden Schritte aus:
 - a. Doppelklicken Sie im Fenster Zertifizierungsstelle auf das Zertifikat.
 - b. Wählen Sie die Registerkarte Details und anschließend In Datei kopieren.
 - c. Befolgen Sie die Anweisungen im Zertifikatexport-Assistent.

Sie haben jetzt eine Windows Server-Zertifizierungsstelle mit und ein gültiges Zertifikat AWS CloudHSM, das von der Windows Server-Zertifizierungsstelle signiert wurde.

Konfigurieren Sie Windows Server als Zertifizierungsstelle (CA) mit Client SDK 3

In einer Public Key-Infrastruktur (PKI) ist eine Zertifizierungsstelle (CA) eine vertrauenswürdige Entität, die digitale Zertifikate ausstellt. Diese digitalen Zertifikate binden einen öffentlichen Schlüssel unter Verwendung von Kryptografie öffentlicher Schlüssel und von digitalen Signaturen an eine Identität (eine Person oder Organisation). Um eine CA zu betreiben, müssen Sie das Vertrauen aufrechterhalten, indem Sie den privaten Schlüssel schützen, der die von Ihrer CA ausgestellten Zertifikate signiert. Sie können den privaten Schlüssel im HSM in Ihrem AWS CloudHSM -Cluster speichern und das HSM zur Durchführung der kryptografischen Signiervorgänge verwenden.

In diesem Tutorial verwenden Sie Windows Server und AWS CloudHSM konfigurieren eine Zertifizierungsstelle. Nach der Installation der AWS CloudHSM -Client-Software für Windows auf Ihrem Windows Server fügen Sie Ihrem Windows Server die Active Directory-Zertifikatsdienste (AD CS)-Rolle hinzu. Wenn Sie diese Rolle konfigurieren, verwenden Sie einen AWS CloudHSM Key Storage Provider (KSP), um den privaten Schlüssel der CA zu erstellen und auf Ihrem AWS CloudHSM Cluster zu speichern. Der KSP ist die Brücke, die Ihren Windows-Server mit Ihrem Cluster verbindet. AWS CloudHSM Im letzten Schritt signieren Sie mit Ihrer Windows Server-CA eine Zertifikatssignierungsanforderung (Certificate Signing Request, CSR).

Weitere Informationen finden Sie unter den folgenden Themen:

Themen

- [Schritt 1: Einrichten der Voraussetzungen](#)
- [Schritt 2: Erstellen Sie eine Windows Server-Zertifizierungsstelle mit AWS CloudHSM](#)
- [Schritt 3: Signieren Sie eine Zertifikatssignieranforderung \(CSR\) mit Ihrer Windows Server-Zertifizierungsstelle mit AWS CloudHSM](#)

Schritt 1: Einrichten der Voraussetzungen

Um Windows Server als Zertifizierungsstelle (CA) mit einzurichten AWS CloudHSM, benötigen Sie Folgendes:

- Ein aktiver AWS CloudHSM Cluster mit mindestens einem HSM.
- Eine Amazon EC2 EC2-Instance, auf der ein Windows Server-Betriebssystem mit installierter AWS CloudHSM Client-Software für Windows ausgeführt wird. In diesem Tutorial wird Microsoft Windows Server 2016 verwendet.
- Einen Crypto-Benutzer (CU), der den privaten Schlüssel der CA auf dem HSM besitzen und verwalten soll.

Um die Voraussetzungen für eine Windows Server-CA einzurichten mit AWS CloudHSM

1. Führen Sie die Schritte unter [Erste Schritte](#) aus. Beim Starten der Amazon EC2-Clients wählen Sie ein Windows Server-AMI aus. In diesem Tutorial wird Microsoft Windows Server 2016 verwendet. Wenn Sie diese Schritte ausgeführt haben, verfügen Sie über einen aktiven Cluster mit mindestens einem HSM. Sie haben auch eine Amazon EC2 EC2-Client-Instance, auf der Windows Server ausgeführt wird und auf der die AWS CloudHSM Client-Software für Windows installiert ist.
2. (Optional) Fügen Sie Ihrem Cluster weitere HSMs hinzu. Weitere Informationen finden Sie unter [Hinzufügen eines HSM zu einem Cluster AWS CloudHSM](#).
3. Stellen Sie eine Verbindung mit Ihrer Client-Instance her. Weitere Informationen finden Sie unter [Connect to Your Instance](#) im Amazon EC2 EC2-Benutzerhandbuch.
4. Erstellen Sie einen Crypto-Benutzer (CU) mithilfe [von Managing HSM users with CloudHSM CLI oder Managing HSM users with CloudHSM](#) Management Utility (CMU). Merken Sie sich den CU-Benutzernamen und das Passwort. Sie benötigen diese im nächsten Schritt.

5. [Legen Sie die Anmeldeinformationen für das HSM fest](#). Verwenden Sie hierfür den im vorherigen Schritt erstellten CU-Benutzernamen und das entsprechende Passwort.
6. Wenn Sie in Schritt 5 den Windows Credentials Manager zum Einrichten von HSM-Anmeldeinformationen verwendet haben, laden Sie den folgenden Befehl als NT Authority\ [psexec.exe](#)SYSTEM herunter, SysInternals um ihn auszuführen:

```
psexec.exe -s "C:\Program Files\Amazon\CloudHsm\tools  
\set_cloudhsm_credentials.exe" --username <USERNAME> --password <PASSWORD>
```

Ersetzen Sie **<USERNAME>** und **<PASSWORD>** durch die HSM-Anmeldeinformationen.

Um eine Windows Server-Zertifizierungsstelle mit zu erstellen AWS CloudHSM, gehen Sie zu [Erstellen einer Windows Server-CA](#).

Schritt 2: Erstellen Sie eine Windows Server-Zertifizierungsstelle mit AWS CloudHSM

Um eine Windows Server-CA zu erstellen, fügen Sie Ihrem Windows Server die Active Directory-Zertifikatsdienste (Active Directory Certificate Services, AD CS)-Rolle hinzu. Wenn Sie diese Rolle hinzufügen, verwenden Sie einen AWS CloudHSM Schlüsselspeicheranbieter (KSP), um den privaten Schlüssel der Zertifizierungsstelle zu erstellen und auf Ihrem AWS CloudHSM Cluster zu speichern.

Note

Wenn Sie Ihre Windows Server-CA erstellen, können Sie sich für das Erstellen einer Stamm-Zertifizierungsstelle oder einer untergeordneten Zertifizierungsstelle entscheiden. Sie treffen diese Entscheidung in der Regel basierend auf dem Entwurf Ihrer Public Key-Infrastruktur und der Sicherheitsrichtlinien Ihrer Organisation. In diesem Tutorial wird der Einfachheit halber erklärt, wie eine Stamm-CA erstellt wird.

So fügen Sie Ihrem Windows Server die AD-CS-Rolle hinzu und erstellen den privaten Schlüssel der CA

1. Stellen Sie eine Verbindung mit Ihrem Windows-Server her, sofern noch nicht geschehen. Weitere Informationen finden Sie unter [Connect to Your Instance](#) im Amazon EC2 EC2-Benutzerhandbuch.

2. Starten Sie auf Ihrem Windows-Server Server Manager.
3. Wählen Sie im Dashboard Server-Manager die Option Rollen und Features hinzufügen.
4. Lesen Sie die Informationen unter Before you begin (Zur Vorbereitung) und klicken Sie dann auf Weiter.
5. Wählen Sie unter Installationsart die Rollenbasierte oder funktionsbasierte Installation. Wählen Sie anschließend Weiter.
6. Wählen Sie unter Serverauswahl die Option Einen Server aus dem Serverpool auswählen. Klicken Sie anschließend auf Weiter.
7. Gehen Sie für Serverrollen wie folgt vor:
 - a. Wählen Sie Active Directory-Zertifikatdienste.
 - b. Wählen Sie für Sollen für Active Directory-Zertifikatsdienste erforderliche Features hinzugefügt werden? die Option Features hinzufügen.
 - c. Klicken Sie auf Weiter, um das Auswählen von Serverrollen abzuschließen.
8. Übernehmen Sie für Features die Standardwerte und klicken Sie dann auf Weiter.
9. Verfahren Sie für AD CS wie folgt:
 - a. Wählen Sie Weiter aus.
 - b. Wählen Sie Zertifizierungsstelle und klicken Sie dann auf Weiter.
10. Lesen Sie unter Bestätigung die Informationen zur Bestätigung und klicken Sie dann auf Installieren. Schließen Sie das Fenster nicht.
11. Wählen Sie den hervorgehobenen Link Active Directory-Zertifikatdienste auf dem Zielservers konfigurieren.
12. Prüfen oder ändern Sie die Anmeldeinformationen unter Anmeldeinformationen. Klicken Sie anschließend auf Weiter.
13. Wählen Sie für Rollendienste die Option Zertifizierungsstelle. Klicken Sie anschließend auf Weiter.
14. Wählen Sie für Installationsart die Option Eigenständige Zertifizierungsstelle. Klicken Sie anschließend auf Weiter.
15. Wählen Sie für Zertifizierungsstellentyp die Option Stammzertifizierungsstelle. Klicken Sie anschließend auf Weiter.

 Note

Sie können basierend auf dem Entwurf Ihrer Public Key-Infrastruktur und der Sicherheitsrichtlinien Ihrer Organisation entscheiden, ob eine Stammzertifizierungsstelle oder eine untergeordnete Zertifizierungsstelle erstellt werden soll. In diesem Tutorial wird der Einfachheit halber erklärt, wie eine Stamm-CA erstellt wird.

16. Wählen Sie für Privater Schlüssel die Option Neuen privaten Schlüssel erstellen. Klicken Sie anschließend auf Weiter.
17. Führen Sie für Kryptografie einen der folgenden Schritte aus:
 - a. Wählen Sie für Wählen Sie einen Kryptografie-Provider aus eine der Optionen für Cavium-Schlüsselspeicher-Provider im Menü aus. Dies sind die AWS CloudHSM - Schlüsselspeicher-Provider. Beispielsweise können Sie RSA#Cavium Key Storage Provider (RSA#Cavium-Schlüsselspeicher-Provider) wählen.
 - b. Wählen Sie für Schlüssellänge eine der verfügbaren Schlüssellängen aus.
 - c. Wählen Sie für Select the hash algorithm für signing certificates issued by this CA (Wählen Sie den Hashalgorithmus zum Signieren von Zertifikaten aus, die von dieser Zertifizierungsstelle ausgestellt wurden) einen der verfügbaren Hashalgorithmen aus.

Wählen Sie Weiter.

18. Führen Sie für Zertifizierungsstelle einen der folgenden Schritte aus:
 - a. (Optional) Bearbeiten Sie den allgemeinen Namen.
 - b. (Optional) Geben Sie ein spezifisches Namenssuffix ein.

Wählen Sie Weiter.

19. Geben Sie für Gültigkeitsdauer einen Zeitraum in Jahren, Monaten, Wochen oder Tagen ein. Klicken Sie anschließend auf Weiter.
20. Für Zertifikatdatenbank können Sie die Standardwerte übernehmen. Sie können optional den Speicherort für die Datenbank und das Datenbankprotokoll ändern. Klicken Sie anschließend auf Weiter.
21. Überprüfen Sie die Informationen zur Zertifizierungsstelle unter Bestätigung. Klicken Sie dann auf Konfigurieren.

22. Klicken Sie auf Schließen und dann erneut auf Schließen.

Sie haben jetzt eine Windows Server-CA mit AWS CloudHSM. Weitere Informationen zum Signieren einer Zertifikatssignierungsanforderung (Certificate Signing Request, CSR) mit Ihrer CA finden Sie unter [Signieren einer CSR](#).

Schritt 3: Signieren Sie eine Zertifikatssignieranforderung (CSR) mit Ihrer Windows Server-Zertifizierungsstelle mit AWS CloudHSM

Sie können Ihre Windows Server-Zertifizierungsstelle verwenden, AWS CloudHSM um eine Zertifikatssignieranforderung (CSR) zu signieren. Um diese Schritte ausführen zu können, benötigen Sie eine gültige CSR. Es gibt mehrere Möglichkeiten, eine CSR zu erstellen, einschließlich der Folgenden:

- Verwenden von OpenSSL
- Verwenden des Windows Server Internet Information Services (IIS) Manager
- Verwenden des Zertifikate-Snap-In in der Microsoft Management Console
- Verwenden des Befehlszeilendienstprogramms certreq unter Windows

Die Schritte zum Erstellen einer CSR werden in dieser Anleitung allerdings nicht behandelt. Wenn Sie über eine CSR verfügen, können Sie sie mit Ihrer Windows Server-Zertifizierungsstelle signieren.

So signieren Sie eine CSR mit Ihrer Windows Server-CA

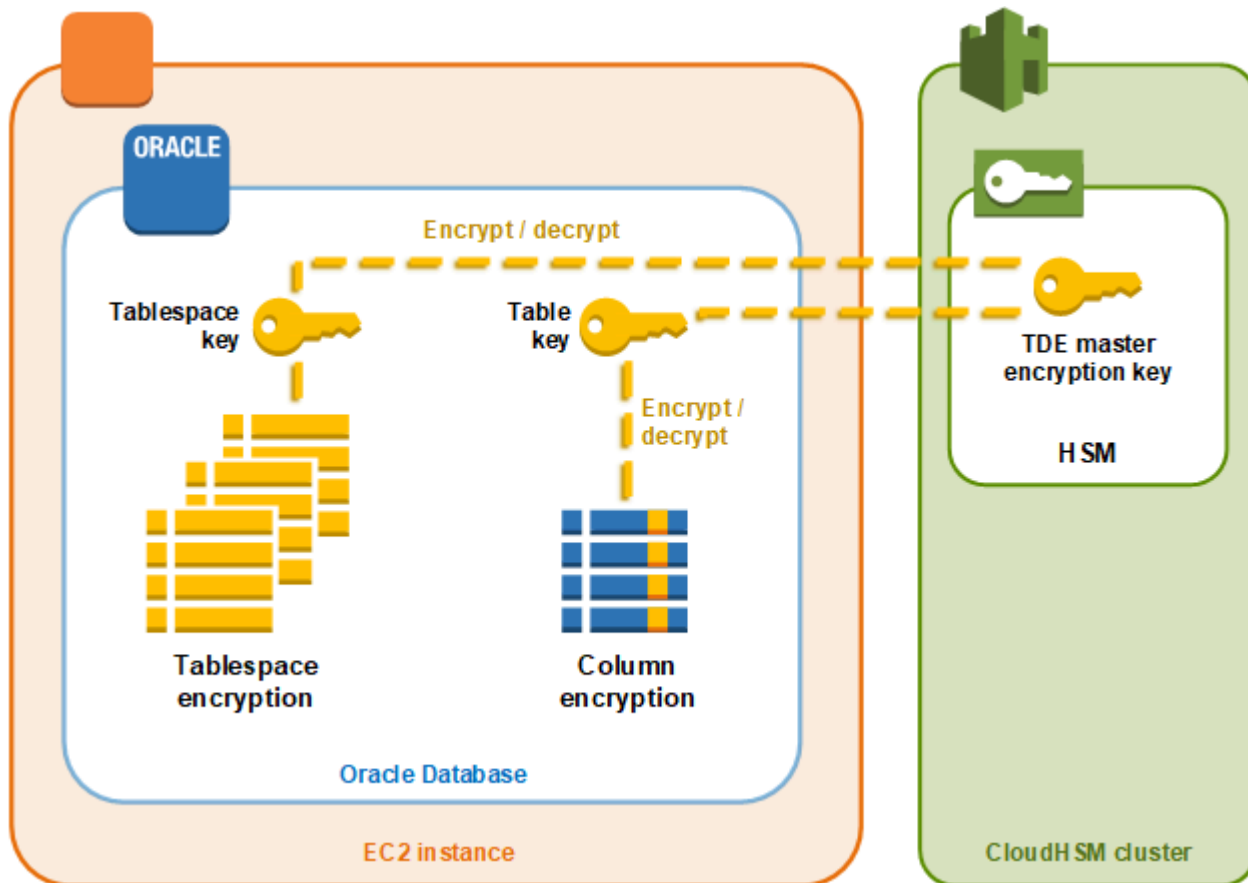
1. Stellen Sie eine Verbindung mit Ihrem Windows-Server her, sofern noch nicht geschehen. Weitere Informationen finden Sie unter [Connect to Your Instance](#) im Amazon EC2 EC2-Benutzerhandbuch.
2. Starten Sie auf Ihrem Windows-Server Server Manager.
3. Klicken Sie rechts oben im Dashboard Server-Manager auf Extras, Zertifizierungsstelle.
4. Wählen Sie im Fenster Zertifizierungsstelle den Namen Ihres Computers aus.
5. Wählen Sie im Menü Aktion die Option Alle Aufgaben, Neue Anforderung einreichen.
6. Wählen Sie Ihre CSR-Datei aus und klicken Sie auf Öffnen.
7. Doppelklicken Sie im Fenster Zertifizierungsstelle auf Ausstehende Anforderungen.
8. Wählen Sie die ausstehende Anforderung aus. Wählen Sie dann im Menü Aktion die Option Alle Aufgaben, Ausstellen.

9. Doppelklicken Sie im Fenster Zertifizierungsstelle auf Ausgestellte Anforderungen, um das signierte Zertifikat anzuzeigen.
10. (Optional) Um das signierte Zertifikat in eine Datei zu exportieren, führen Sie die folgenden Schritte aus:
 - a. Doppelklicken Sie im Fenster Zertifizierungsstelle auf das Zertifikat.
 - b. Wählen Sie die Registerkarte Details und anschließend In Datei kopieren.
 - c. Befolgen Sie die Anweisungen im Zertifikatexport-Assistent.

Sie haben jetzt eine Windows Server-Zertifizierungsstelle mit und ein gültiges Zertifikat AWS CloudHSM, das von der Windows Server-Zertifizierungsstelle signiert wurde.

Transparente Datenverschlüsselung (TDE) für Oracle-Datenbanken mit AWS CloudHSM

Transparent Data Encryption (TDE) wird für die Verschlüsselung von Datenbankdateien eingesetzt. Mit TDE verschlüsselt die Datenbanksoftware die Daten, bevor sie auf der Festplatte gespeichert werden. Die Daten in den Tabellenspalten oder Tablespaces der Datenbank werden mit einem Tabellenschlüssel oder einem Tablespace-Schlüssel verschlüsselt. Einige Versionen der Datenbanksoftware von Oracle bieten TDE. In Oracle TDE werden diese Schlüssel mit einem TDE-Master-Schlüssel verschlüsselt. Sie können mehr Sicherheit erreichen, indem Sie den TDE-Master-Verschlüsselungsschlüssel HSMs in Ihrem AWS CloudHSM Cluster speichern.



In dieser Lösung verwenden Sie Oracle Database, das in einer Amazon EC2-Instance installiert ist. Oracle Database ist in die [AWS CloudHSM Softwarebibliothek für PKCS #11](#) integriert, um den TDE-Hauptschlüssel in Ihrem Cluster zu speichern. HSMs

⚠ Important

- Wir empfehlen die Installation der Oracle-Datenbank auf einer Amazon-EC2-Instance.

Führen Sie die folgenden Schritte aus, um die Oracle TDE-Integration mit AWS CloudHSM zu vollenden.

Themen

- [Schritt 1: Richten Sie die Voraussetzungen ein](#)
- [Schritt 2: Aktualisieren Sie die Oracle-Datenbankkonfiguration](#)
- [Schritt 3: Generieren Sie den Oracle TDE-Master-Verschlüsselungsschlüssel](#)

Schritt 1. Richten Sie die Voraussetzungen ein

Um die Oracle TDE-Integration mit durchzuführen AWS CloudHSM, benötigen Sie Folgendes:

- Ein aktiver AWS CloudHSM Cluster mit mindestens einem HSM.
- Eine Amazon EC2-Instance, auf der das Amazon Linux-Betriebssystem mit der folgenden Software installiert ist:
 - Der AWS CloudHSM Client und die Befehlszeilentools.
 - Die AWS CloudHSM Softwarebibliothek für PKCS #11.
 - Oracle-Datenbank. AWS CloudHSM unterstützt die Oracle TDE-Integration. Client-SDK 5.6 und höher unterstützen Oracle-Datenbank 19c. Das Client-SDK 3 unterstützt Oracle TDE für die Oracle-Datenbankversionen 11g und 12c.
- Ein kryptografischer Benutzer (CU), der den TDE-Master-Verschlüsselungsschlüssel HSMs in Ihrem Cluster besitzt und verwaltet.

Führen Sie zum Einrichten aller Voraussetzungen die folgenden Schritte aus.

Um die Voraussetzungen für die Oracle TDE-Integration mit einzurichten AWS CloudHSM

1. Führen Sie die Schritte unter [Erste Schritte](#) aus. Anschließend verfügen Sie über einen aktiven Cluster mit einem HSM. Außerdem verfügen Sie über eine Amazon-EC2-Instance, auf der das Amazon-Linux-Betriebssystem läuft. Die AWS CloudHSM Client- und Befehlszeilentools werden ebenfalls installiert und konfiguriert.
2. (Optional) Fügen Sie Ihrem Cluster weitere HSMs hinzu. Weitere Informationen finden Sie unter [Hinzufügen eines HSM zu einem Cluster AWS CloudHSM](#).
3. Stellen Sie eine Verbindung zur Ihrer Amazon EC2-Client-Instance her und führen Sie die folgenden Schritte aus:
 - a. [Installieren Sie die AWS CloudHSM Softwarebibliothek für PKCS #11](#).
 - b. Installieren Sie Oracle Database. Weitere Informationen finden Sie in der [Oracle Database-Dokumentation](#). Client-SDK 5.6 und höher unterstützen Oracle-Datenbank 19c. Das Client-SDK 3 unterstützt Oracle TDE für die Oracle-Datenbankversionen 11g und 12c.
 - c. Verwenden Sie zum Erstellen eines Crypto-Benutzers (CU) in Ihrem Cluster das Befehlszeilen-Tool cloudhsm_mgmt_util. Weitere Informationen zum Erstellen eines CU finden Sie unter [So verwalten Sie HSM-Benutzer mit CMU](#) und [HSM-Benutzer](#).

Schritt 2: Aktualisieren Sie die Oracle-Datenbankkonfiguration

Um die Oracle Database-Konfiguration zu aktualisieren, um ein HSM in Ihrem Cluster als externes Sicherheitsmodul zu verwenden, führen Sie die folgenden Schritte aus. Weitere Informationen zu externen Sicherheitsmodulen finden Sie unter [Introduction to Transparent Data Encryption](#) im Oracle Database Advanced Security Guide.

So aktualisieren Sie die Oracle-Konfiguration

1. Stellen Sie eine Verbindung zu Ihrer Amazon-EC2-Client-Instance her. Dies ist die Instance, in der Sie Oracle Database installiert haben.
2. Erstellen Sie eine Sicherungskopie der Datei `sqlnet.ora`. Informationen zum Speicherort dieser Datei finden Sie in der Oracle-Dokumentation.
3. Verwenden Sie einen Texteditor, um die Daten namens `sqlnet.ora` zu bearbeiten. Fügen Sie die folgende Zeile zu. Wenn eine Zeile in der Datei mit `encryption_wallet_location` beginnt, ersetzen Sie diese durch die folgende.

```
encryption_wallet_location=(source=(method=hsm))
```

Speichern Sie die Datei.

4. Führen Sie den folgenden Befehl aus, um das Verzeichnis zu erstellen, in dem Oracle Database erwartet, die Bibliotheksdatei für die AWS CloudHSM PKCS #11 -Softwarebibliothek zu finden.

```
sudo mkdir -p /opt/oracle/extapi/64/hsm
```

5. Führen Sie den folgenden Befehl aus, um die AWS CloudHSM Softwarebibliothek für die PKCS #11 -Datei in das Verzeichnis zu kopieren, das Sie im vorherigen Schritt erstellt haben.

```
sudo cp /opt/cloudhsm/lib/libcloudhsm_pkcs11.so /opt/oracle/extapi/64/hsm/
```

Note

Das Verzeichnis `/opt/oracle/extapi/64/hsm` darf nur eine Bibliotheksdatei enthalten. Entfernen Sie alle anderen Dateien, die in diesem Verzeichnis vorhanden sind.

6. Führen Sie den folgenden Befehl aus, um den Eigentümer des Verzeichnisses `/opt/oracle` und alle darin enthaltenen Daten zu ändern.

```
sudo chown -R oracle:dba /opt/oracle
```

7. Starten Sie Oracle Database.

Schritt 3: Generieren Sie den Oracle TDE-Master-Verschlüsselungsschlüssel

Um den Oracle TDE-Masterschlüssel auf dem HSMs in Ihrem Cluster zu generieren, führen Sie die Schritte im folgenden Verfahren aus.

So generieren Sie den Masterschlüssel

1. Verwenden Sie den folgenden Befehl, um Oracle SQL*Plus zu öffnen. Geben Sie, wenn Sie dazu aufgefordert werden, das Systempasswort ein, das Sie beim Installieren von Oracle Database festgelegt haben.

```
sqlplus / as sysdba
```

Note

Für Client-SDK 3 müssen Sie die `CLOUDHSM_IGNORE_CKA_MODIFIABLE_FALSE`-Umgebungsvariable jedes Mal setzen, wenn Sie einen Hauptschlüssel erzeugen. Diese Variable wird nur für die Generierung des Masterschlüssels benötigt. Weitere Informationen finden Sie unter „Problem: Oracle legt das PCS #11-Attribut `CKA_MODIFIABLE` bei der Generierung des Masterschlüssels fest, aber das HSM unterstützt es nicht“ unter [Bekannte Probleme für die Integration von Drittanbieteranwendungen](#).

2. Führen Sie die SQL-Anweisung aus, die den Master-Verschlüsselungsschlüssel erstellt, wie in den folgenden Beispielen gezeigt. Verwenden Sie die Anweisung, die Ihrer Version von Oracle Database entspricht. `<CU user name>` Ersetzen Sie ihn durch den Benutzernamen des kryptografischen Benutzers (CU). Ersetzen Sie es `<password>` durch das CU-Passwort.

⚠ Important

Führen Sie einmalig den folgenden Befehl aus. Jedes Mal, wenn der Befehl ausgeführt wird, wird ein neuer Master-Verschlüsselungsschlüssel erstellt.

- Führen Sie für Oracle Database, Version 11, die folgende SQL-Anweisung aus.

```
SQL> alter system set encryption key identified by "<CU user name>:<password>";
```

- Für Oracle Database Version 12 und Version 19c führen Sie die folgende SQL-Anweisung aus.

```
SQL> administer key management set key identified by "<CU user name>:<password>";
```

Wenn die Antwort `System altered` oder `keystore altered` ist, haben Sie den Masterschlüssel für Oracle TDE erfolgreich generiert und festgelegt.

- (Optional) Führen Sie den folgenden Befehl aus, um den Status von Oracle Wallet zu verifizieren.

```
SQL> select * from v$encryption_wallet;
```

Ist das Wallet nicht geöffnet, öffnen Sie es mit einem der folgenden Befehle. `<CU user name>` Ersetzen Sie es durch den Namen des kryptografischen Benutzers (CU). Ersetzen Sie es `<password>` durch das CU-Passwort.

- Führen Sie für Oracle 11 zum Öffnen des Wallet den folgenden Befehl aus.

```
SQL> alter system set encryption wallet open identified by "<CU user name>:<password>";
```

Führen Sie zum manuellen Schließen des Wallet den folgenden Befehl aus.

```
SQL> alter system set encryption wallet close identified by "<CU user name>:<password>";
```

- Bei Oracle 12 und Oracle 19c führen Sie den folgenden Befehl aus, um die Brieftasche zu öffnen.

```
SQL> administer key management set keystore open identified by "<CU user  
name>:<password>";
```

Führen Sie zum manuellen Schließen des Wallet den folgenden Befehl aus.

```
SQL> administer key management set keystore close identified by "<CU user  
name>:<password>";
```

Verwenden Sie Microsoft SignTool mit AWS CloudHSM , um Dateien zu signieren

AWS CloudHSM bietet Unterstützung für die Verwendung von Microsoft Signtool zum Signieren von Dateien über Client SDK 3 und Client SDK 5. Die Schritte zur Verwendung dieser Tools hängen von der Version des Client-SDK ab, die Sie derzeit heruntergeladen haben. Die folgenden Abschnitte enthalten Informationen zu den einzelnen SDKs.

Themen

- [Verwenden Sie Microsoft SignTool mit Client SDK 5 zum Signieren von Dateien](#)
- [Verwenden Sie Microsoft SignTool mit Client SDK 3 zum Signieren von Dateien](#)

Verwenden Sie Microsoft SignTool mit Client SDK 5 zum Signieren von Dateien

In der Kryptographie und bei PKIs (Public Key Infrastructure) werden digitale Signaturen verwendet, um zu bestätigen, dass Daten von einer vertrauenswürdigen Einheit gesendet wurden. Signaturen zeigen außerdem, dass die Daten während der Übertragung nicht manipuliert wurden. Eine Signatur ist ein verschlüsselter Hash, der mit dem privaten Schlüssel des Absenders erzeugt wird. Der Empfänger kann die Datenintegrität überprüfen, indem er seine Hashsignatur mit dem öffentlichen Schlüssel des Absenders entschlüsselt. Im Gegenzug liegt es in der Verantwortung des Absenders, ein digitales Zertifikat zu pflegen. Das digitale Zertifikat belegt den Besitz des privaten Schlüssels durch den Absender und stellt dem Empfänger den öffentlichen Schlüssel zur Verfügung, der für die Entschlüsselung benötigt wird. Solange der private Schlüssel dem Absender gehört, kann

der Signatur vertraut werden. AWS CloudHSM bietet sichere, nach FIPS 140-2 Level 3 validierte Hardware, mit der Sie diese Schlüssel mit exklusivem Einzelmandantenzugriff sichern können.

Viele Organisationen verwenden Microsoft SignTool, ein Befehlszeilentool, das Dateien signiert, verifiziert und mit einem Zeitstempel versehen kann, um den Codesignaturprozess zu vereinfachen. Sie können AWS CloudHSM es verwenden, um Ihre Schlüsselpaare sicher zu speichern, bis sie von Ihnen benötigt werden SignTool, und so einen leicht automatisierbaren Arbeitsablauf für das Signieren von Daten schaffen.

Die folgenden Themen bieten einen Überblick über die Verwendung SignTool von. AWS CloudHSM

Themen

- [Schritt 1: Einrichten der Voraussetzungen](#)
- [Schritt 2: Erstellen Sie ein Signaturzertifikat](#)
- [Schritt 3: Signieren Sie eine Datei](#)

Schritt 1: Einrichten der Voraussetzungen

Um Microsoft SignTool mit verwenden zu können AWS CloudHSM, benötigen Sie Folgendes:

- Eine Amazon-EC2-Client-Instance, auf der ein Windows-Betriebssystem läuft.
- Eine Zertifizierungsstelle (CA), die entweder selbst unterhalten oder von einem Drittanbieter eingerichtet wurde.
- Ein aktiver AWS CloudHSM Cluster in derselben virtuellen Public Cloud (VPC) wie Ihre EC2-Instance. Der Cluster muss mindestens ein HSM enthalten.
- Ein Krypto-Benutzer (CU), der Schlüssel im Cluster besitzt und verwaltet. AWS CloudHSM
- Eine unsignierte Datei oder ausführbare Datei.
- Das Microsoft Windows Software Development Kit (SDK).

Um die Voraussetzungen für die Verwendung AWS CloudHSM mit Windows einzurichten SignTool

1. Befolgen Sie die Anweisungen im Abschnitt [Erste Schritte](#) dieses Handbuchs, um eine Windows EC2-Instance und einen AWS CloudHSM -Cluster zu starten.
2. Wenn Sie Ihre eigene Windows Server-Zertifizierungsstelle hosten möchten, folgen Sie den Schritten 1 und 2 [unter Windows Server als Zertifizierungsstelle konfigurieren mit](#)

[AWS CloudHSM](#). Verwenden Sie andernfalls weiterhin Ihre öffentlich vertrauenswürdige Zertifizierungsstelle eines Drittanbieters.

3. Laden Sie eine der folgenden Versionen des Microsoft Windows-SDKs herunter und installieren Sie es in Ihrer Windows EC2-Instance:

- [Microsoft Windows SDK 10](#)
- [Microsoft Windows SDK 8.1](#)
- [Microsoft Windows SDK 7](#)

Die ausführbare Datei `SignTool` ist Teil der Installationsfeatures von Windows SDK Signing Tools für Desktop Apps. Sie können die anderen zu installierenden Features weglassen, wenn Sie sie nicht benötigen. Der Standardinstallationspfad ist:

```
C:\Program Files (x86)\Windows Kits\<SDK version>\bin\<version number>\<CPU architecture>\signtool.exe
```

Sie können jetzt das Microsoft Windows SDK, Ihren AWS CloudHSM Cluster und Ihre CA verwenden, um [ein Signaturzertifikat zu erstellen](#).

Schritt 2: Erstellen Sie ein Signaturzertifikat

Nachdem Sie das Windows-SDK auf Ihre EC2-Instance heruntergeladen haben, können Sie damit eine Certificate Signing Request (CSR) generieren. Die CSR ist ein unsigniertes Zertifikat, das möglicherweise zur Signatur an Ihre CA weitergeleitet wird. In diesem Beispiel verwenden wir die ausführbare Datei `certreq`, die im Windows-SDK enthalten ist, um die CSR zu generieren.

So erstellen Sie eine CSR mit der ausführbaren Datei **certreq**:

1. Wenn Sie dies noch nicht getan haben, verbinden Sie sich mit Ihrer Windows EC2-Instance. Weitere Informationen finden Sie unter [Connect to Your Instance](#) im Amazon EC2 EC2-Benutzerhandbuch.
2. Erstellen Sie eine Datei namens `request.inf`, die die folgenden Zeilen enthält. Ersetzen Sie die Subject-Informationen durch die Ihrer Organisation. Eine Erklärung der einzelnen Parameter finden Sie in der [Dokumentation von Microsoft](#).

```
[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. Führen Sie `certreq.exe`. In diesem Beispiel speichern wir die CSR als `request.csr`.

```
certreq.exe -new request.inf request.csr
```

Intern wird auf Ihrem AWS CloudHSM Cluster ein neues key pair generiert, und der private Schlüssel des Paares wird verwendet, um die CSR zu erstellen.

4. Übermitteln Sie die CSR an Ihre CA. Wenn Sie eine Windows Server-CA verwenden, führen Sie diese Schritte aus:

- a. Geben Sie den folgenden Befehl ein, um das CA-Tool zu öffnen:

```
certsrv.msc
```

- b. Klicken Sie im neuen Fenster mit der rechten Maustaste auf den Namen des CA-Servers. Wählen Sie Alle Aufgaben aus, und wählen Sie dann Neue Anfrage senden aus.
- c. Navigieren Sie zum Speicherort von `request.csr` und wählen Sie Öffnen aus.
- d. Navigieren Sie zum Ordner Pending Requests (Ausstehende Anforderungen), indem Sie das Menü Server CA erweitern. Klicken Sie mit der rechten Maustaste auf die gerade erstellte Anforderung, und wählen Sie unter All tasks (Alle Aufgaben) Issue (Ausstellen) aus.
- e. Navigieren Sie nun zum Ordner Issued Certificates (Ausgestellte Zertifikate) (über dem Ordner Pending Requests (Ausstehende Anforderungen)).
- f. Wählen Sie Öffnen aus, um das Zertifikat anzuzeigen, und wählen Sie dann die Registerkarte Details aus.
- g. Wählen Sie In Datei kopieren aus, um den Assistenten für den Zertifikatsexport zu starten. Speichern Sie die DER-kodierte X.509-Datei an einem sicheren Ort als `signedCertificate.cer`.

- h. Verlassen Sie das CA-Tool und verwenden Sie den folgenden Befehl, der die Zertifikatsdatei in den persönlichen Zertifikatsspeicher unter Windows verschiebt. Sie kann dann von anderen Anwendungen verwendet werden.

```
certreq.exe -accept signedCertificate.cer
```

Sie können zum [Signieren einer Datei](#) nun Ihr importiertes Zertifikat in verwenden.

Schritt 3: Signieren Sie eine Datei

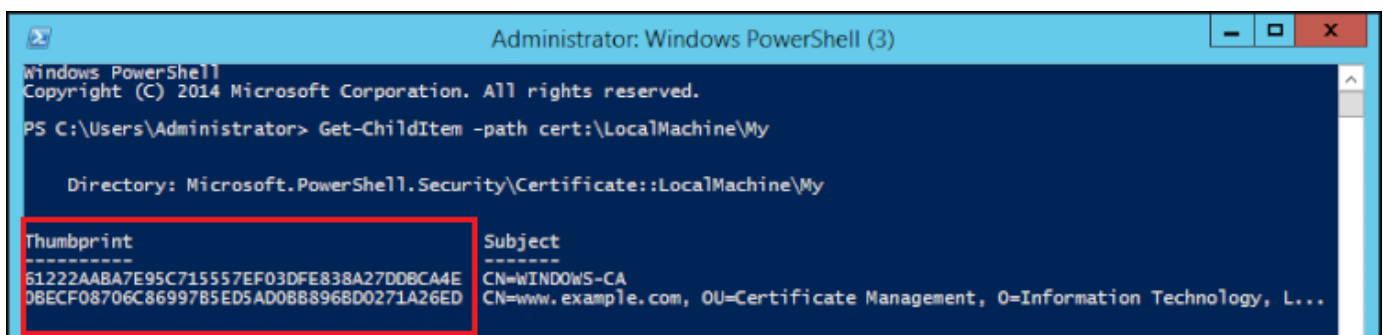
Sie können jetzt Ihr importiertes Zertifikat verwenden SignTool, um Ihre Beispieldatei zu signieren. Um dies zu tun, müssen Sie den SHA-1-Hash oder Fingerabdruck (Thumbprint) des Zertifikats kennen. Der Fingerabdruck wird verwendet, um sicherzustellen, dass SignTool nur Zertifikate verwendet werden, die von verifiziert wurden. AWS CloudHSM In diesem Beispiel verwenden wir, PowerShell um den Hash des Zertifikats abzurufen. Sie können auch die GUI der CA oder die ausführbare Datei certutil des Windows-SDKs verwenden.

So erhalten Sie den Fingerabdruck eines Zertifikats und signieren damit eine Datei:

1. Öffnen Sie PowerShell als Administrator und führen Sie den folgenden Befehl aus:

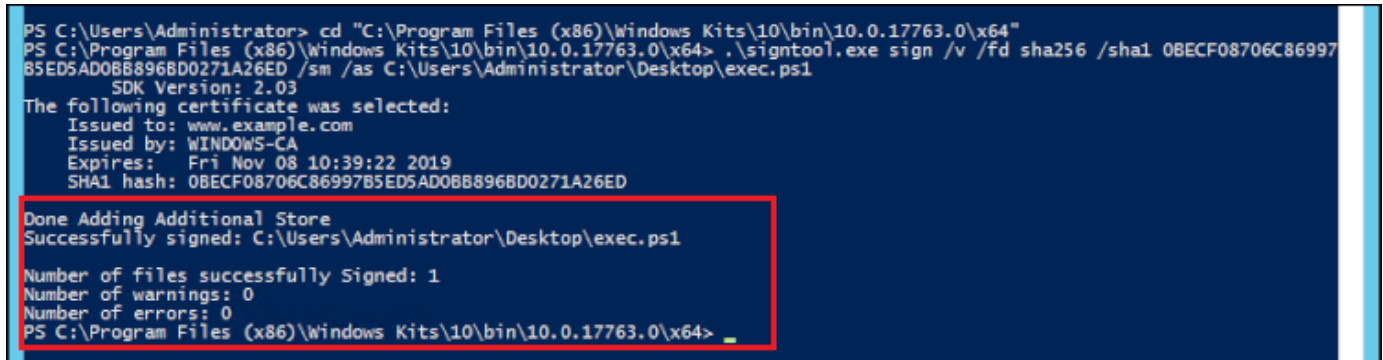
```
Get-ChildItem -path cert:\LocalMachine\My
```

Kopieren Sie den Thumbprint, der zurückgegeben wird.



2. Navigieren Sie zu dem Verzeichnis PowerShell, das Folgendes enthält SignTool.exe. Der Standardspeicherort ist C:\Program Files (x86)\Windows Kits\10\bin\10.0.17763.0\x64.
3. Signieren Sie dann Ihre Datei, indem Sie den folgenden Befehl ausführen. Wenn der Befehl erfolgreich ist, wird eine Erfolgsmeldung PowerShell zurückgegeben.

```
signtool.exe sign /v /fd sha256 /sha1 <thumbprint> /sm C:\Users\Administrator\
\Desktop\<test>.ps1
```



```
PS C:\Users\Administrator> cd "C:\Program Files (x86)\Windows Kits\10\bin\10.0.17763.0\x64"
PS C:\Program Files (x86)\Windows Kits\10\bin\10.0.17763.0\x64> .\signtool.exe sign /v /fd sha256 /sha1 0BECF08706C86997
85ED5AD0B8896BD0271A26ED /sm /as C:\Users\Administrator\Desktop\exec.ps1
    SDK Version: 2.03
The following certificate was selected:
    Issued to: www.example.com
    Issued by: WINDOWS-CA
    Expires:   Fri Nov 08 10:39:22 2019
    SHA1 hash: 0BECF08706C8699785ED5AD0B8896BD0271A26ED
Done Adding Additional Store
Successfully signed: C:\Users\Administrator\Desktop\exec.ps1
Number of files successfully Signed: 1
Number of warnings: 0
Number of errors: 0
PS C:\Program Files (x86)\Windows Kits\10\bin\10.0.17763.0\x64>
```

4. (Optional) Um die Signatur der Datei zu überprüfen, verwenden Sie den folgenden Befehl:

```
signtool.exe verify /v /pa C:\Users\Administrator\Desktop\<test>.ps1
```

Verwenden Sie Microsoft SignTool mit Client SDK 3 zum Signieren von Dateien

In der Kryptographie und bei PKIs (Public Key Infrastructure) werden digitale Signaturen verwendet, um zu bestätigen, dass Daten von einer vertrauenswürdigen Einheit gesendet wurden. Signaturen zeigen außerdem, dass die Daten während der Übertragung nicht manipuliert wurden. Eine Signatur ist ein verschlüsselter Hash, der mit dem privaten Schlüssel des Absenders erzeugt wird. Der Empfänger kann die Datenintegrität überprüfen, indem er seine Hash-Signatur mit dem öffentlichen Schlüssel des Absenders entschlüsselt. Im Gegenzug liegt es in der Verantwortung des Absenders, ein digitales Zertifikat zu pflegen. Das digitale Zertifikat belegt den Besitz des privaten Schlüssels durch den Absender und stellt dem Empfänger den öffentlichen Schlüssel zur Verfügung, der für die Entschlüsselung benötigt wird. Solange der private Schlüssel dem Absender gehört, kann der Signatur vertraut werden. AWS CloudHSM bietet sichere, nach FIPS 140-2 Level 3 validierte Hardware, mit der Sie diese Schlüssel mit exklusivem Einzelmandantenzugriff sichern können.

Viele Organisationen verwenden Microsoft SignTool, ein Befehlszeilentool, das Dateien signiert, verifiziert und mit einem Zeitstempel versehen kann, um den Codesignaturprozess zu vereinfachen. Sie können AWS CloudHSM verwenden, um Ihre Schlüsselpaare sicher zu speichern, bis sie von Ihnen benötigt werden SignTool, und so einen leicht automatisierbaren Arbeitsablauf für das Signieren von Daten schaffen.

Die folgenden Themen bieten einen Überblick über die Verwendung SignTool von AWS CloudHSM

Themen

- [Schritt 1: Einrichten der Voraussetzungen](#)
- [Schritt 2: Erstellen Sie ein Signaturzertifikat](#)
- [Schritt 3: Signieren Sie eine Datei](#)

Schritt 1: Einrichten der Voraussetzungen

Um Microsoft SignTool mit verwenden zu können AWS CloudHSM, benötigen Sie Folgendes:

- Eine Amazon-EC2-Client-Instance, auf der ein Windows-Betriebssystem läuft.
- Eine Zertifizierungsstelle (CA), die entweder selbst unterhalten oder von einem Drittanbieter eingerichtet wurde.
- Ein aktiver AWS CloudHSM Cluster in derselben virtuellen Public Cloud (VPC) wie Ihre EC2-Instance. Der Cluster muss mindestens ein HSM enthalten.
- Ein Krypto-Benutzer (CU), der Schlüssel im Cluster besitzt und verwaltet. AWS CloudHSM
- Eine unsignierte Datei oder ausführbare Datei.
- Das Microsoft Windows Software Development Kit (SDK).

Um die Voraussetzungen für die Verwendung AWS CloudHSM mit Windows einzurichten SignTool

1. Befolgen Sie die Anweisungen im Abschnitt [Erste Schritte](#) dieses Handbuchs, um eine Windows EC2-Instance und einen AWS CloudHSM -Cluster zu starten.
2. Wenn Sie Ihre eigene Windows Server-Zertifizierungsstelle hosten möchten, folgen Sie den Schritten 1 und 2 [unter Windows Server als Zertifizierungsstelle konfigurieren mit AWS CloudHSM](#). Verwenden Sie andernfalls weiterhin Ihre öffentlich vertrauenswürdige Zertifizierungsstelle eines Drittanbieters.
3. Laden Sie eine der folgenden Versionen des Microsoft Windows-SDKs herunter und installieren Sie es in Ihrer Windows EC2-Instance:
 - [Microsoft Windows SDK 10](#)
 - [Microsoft Windows SDK 8.1](#)
 - [Microsoft Windows SDK 7](#)

Die ausführbare Datei `SignTool` ist Teil der Installationsfeatures von Windows SDK Signing Tools für Desktop Apps. Sie können die anderen zu installierenden Features weglassen, wenn Sie sie nicht benötigen. Der Standardinstallationspfad ist:

```
C:\Program Files (x86)\Windows Kits\<SDK version>\bin\<version number>\<CPU architecture>\signtool.exe
```

Sie können jetzt das Microsoft Windows SDK, Ihren AWS CloudHSM Cluster und Ihre CA verwenden, um [ein Signaturzertifikat zu erstellen](#).

Schritt 2: Erstellen Sie ein Signaturzertifikat

Nachdem Sie das Windows-SDK auf Ihre EC2-Instance heruntergeladen haben, können Sie damit eine Certificate Signing Request (CSR) generieren. Die CSR ist ein unsigniertes Zertifikat, das möglicherweise zur Signatur an Ihre CA weitergeleitet wird. In diesem Beispiel verwenden wir die ausführbare Datei `certreq`, die im Windows-SDK enthalten ist, um die CSR zu generieren.

So erstellen Sie eine CSR mit der ausführbaren Datei **`certreq`**:

1. Wenn Sie dies noch nicht getan haben, verbinden Sie sich mit Ihrer Windows EC2-Instance. Weitere Informationen finden Sie unter [Connect to Your Instance](#) im Amazon EC2 EC2-Benutzerhandbuch.
2. Erstellen Sie eine Datei namens `request.inf`, die die folgenden Zeilen enthält. Ersetzen Sie die Subject-Informationen durch die Ihrer Organisation. Eine Erklärung der einzelnen Parameter finden Sie in der [Dokumentation von Microsoft](#).

```
[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. Führen Sie `certreq.exe`. In diesem Beispiel speichern wir die CSR als `request.csr`.

```
certreq.exe -new request.inf request.csr
```

Intern wird auf Ihrem AWS CloudHSM Cluster ein neues key pair generiert, und der private Schlüssel des Paares wird verwendet, um die CSR zu erstellen.

4. Übermitteln Sie die CSR an Ihre CA. Wenn Sie eine Windows Server-CA verwenden, führen Sie diese Schritte aus:

- a. Geben Sie den folgenden Befehl ein, um das CA-Tool zu öffnen:

```
certsrv.msc
```

- b. Klicken Sie im neuen Fenster mit der rechten Maustaste auf den Namen des CA-Servers. Wählen Sie Alle Aufgaben aus, und wählen Sie dann Neue Anfrage senden aus.
- c. Navigieren Sie zum Speicherort von `request.csr` und wählen Sie Öffnen aus.
- d. Navigieren Sie zum Ordner Pending Requests (Ausstehende Anforderungen), indem Sie das Menü Server CA erweitern. Klicken Sie mit der rechten Maustaste auf die gerade erstellte Anforderung, und wählen Sie unter All tasks (Alle Aufgaben) Issue (Ausstellen) aus.
- e. Navigieren Sie nun zum Ordner Issued Certificates (Ausgestellte Zertifikate) (über dem Ordner Pending Requests (Ausstehende Anforderungen)).
- f. Wählen Sie Öffnen aus, um das Zertifikat anzuzeigen, und wählen Sie dann die Registerkarte Details aus.
- g. Wählen Sie In Datei kopieren aus, um den Assistenten für den Zertifikatsexport zu starten. Speichern Sie die DER-kodierte X.509-Datei an einem sicheren Ort als `signedCertificate.cer`.
- h. Verlassen Sie das CA-Tool und verwenden Sie den folgenden Befehl, der die Zertifikatsdatei in den persönlichen Zertifikatsspeicher unter Windows verschiebt. Sie kann dann von anderen Anwendungen verwendet werden.

```
certreq.exe -accept signedCertificate.cer
```

Sie können zum [Signieren einer Datei](#) nun Ihr importiertes Zertifikat in verwenden.

Schritt 3: Signieren Sie eine Datei

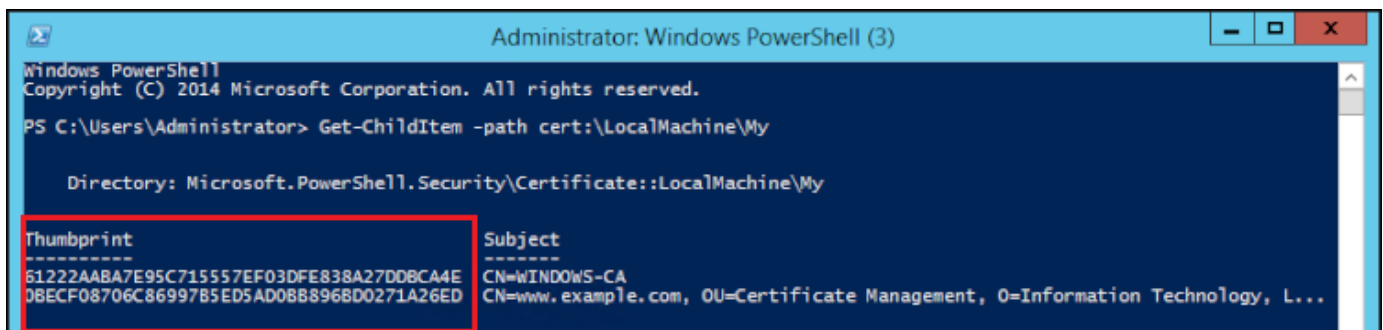
Sie können jetzt Ihr importiertes Zertifikat verwenden SignTool, um Ihre Beispieldatei zu signieren. Um dies zu tun, müssen Sie den SHA-1-Hash oder Fingerabdruck (Thumbprint) des Zertifikats kennen. Der Fingerabdruck wird verwendet, um sicherzustellen, dass SignTool nur Zertifikate verwendet werden, die von verifiziert wurden. AWS CloudHSM In diesem Beispiel verwenden wir, PowerShell um den Hash des Zertifikats abzurufen. Sie können auch die GUI der CA oder die ausführbare Datei `certutil` des Windows-SDKs verwenden.

So erhalten Sie den Fingerabdruck eines Zertifikats und signieren damit eine Datei:

1. Öffnen Sie PowerShell als Administrator und führen Sie den folgenden Befehl aus:

```
Get-ChildItem -path cert:\LocalMachine\My
```

Kopieren Sie den Thumbprint, der zurückgegeben wird.



2. Navigieren Sie zu dem Verzeichnis PowerShell, das Folgendes enthält `SignTool.exe`. Der Standardspeicherort ist `C:\Program Files (x86)\Windows Kits\10\bin\10.0.17763.0\x64`.
3. Signieren Sie dann Ihre Datei, indem Sie den folgenden Befehl ausführen. Wenn der Befehl erfolgreich ist, wird eine Erfolgsmeldung PowerShell zurückgegeben.

```
signtool.exe sign /v /fd sha256 /sha1 <thumbprint> /sm C:\Users\Administrator\Desktop\<test>.ps1
```

```

PS C:\Users\Administrator> cd "C:\Program Files (x86)\Windows Kits\10\bin\10.0.17763.0\x64"
PS C:\Program Files (x86)\Windows Kits\10\bin\10.0.17763.0\x64> .\signtool.exe sign /v /fd sha256 /sha1 0BECF08706C86997
85ED5AD0B8896BD0271A26ED /sm /as C:\Users\Administrator\Desktop\exec.ps1
    SDK Version: 2.03
The following certificate was selected:
    Issued to: www.example.com
    Issued by: WINDOWS-CA
    Expires:   Fri Nov 08 10:39:22 2019
    SHA1 hash: 0BECF08706C8699785ED5AD0B8896BD0271A26ED
Done Adding Additional Store
Successfully signed: C:\Users\Administrator\Desktop\exec.ps1
Number of files successfully Signed: 1
Number of warnings: 0
Number of errors: 0
PS C:\Program Files (x86)\Windows Kits\10\bin\10.0.17763.0\x64>

```

4. (Optional) Um die Signatur der Datei zu überprüfen, verwenden Sie den folgenden Befehl:

```
signtool.exe verify /v /pa C:\Users\Administrator\Desktop\<test>.ps1
```

Integration von Java Keytool und Jarsigner mit AWS CloudHSM

AWS CloudHSM bietet die Integration mit den Java-Dienstprogrammen Keytool und Jarsigner über das Client SDK 3 und das Client SDK 5. Die Schritte zur Verwendung dieser Tools hängen von der Version des Client-SDK ab, die Sie derzeit heruntergeladen haben. Die folgenden Abschnitte enthalten Informationen zu den einzelnen SDKs.

Themen

- [Verwenden Sie Client SDK 5 zur Integration AWS CloudHSM mit Java Keytool und Jarsigner](#)
- [Verwenden Sie Client SDK 3 zur Integration AWS CloudHSM mit Java Keytool und Jarsigner](#)

Verwenden Sie Client SDK 5 zur Integration AWS CloudHSM mit Java Keytool und Jarsigner

AWS CloudHSM Der Schlüsselspeicher ist ein spezieller JCE-Schlüsselspeicher, der Zertifikate verwendet, die mit Schlüsseln auf Ihrem Hardware-Sicherheitsmodul (HSM) verknüpft sind, und zwar über Tools von Drittanbietern wie und. keytool jarsigner AWS CloudHSM speichert keine Zertifikate auf dem HSM, da es sich bei Zertifikaten um öffentliche, nicht vertrauliche Daten handelt. Der AWS CloudHSM Schlüsselspeicher speichert die Zertifikate in einer lokalen Datei und ordnet die Zertifikate den entsprechenden Schlüsseln auf Ihrem HSM zu.

Wenn Sie den AWS CloudHSM Schlüsselspeicher verwenden, um neue Schlüssel zu generieren, werden keine Einträge in der lokalen Schlüsselspeicherdatei generiert — die Schlüssel werden

auf dem HSM erstellt. Wenn Sie den AWS CloudHSM -Schlüsselspeicher verwenden, um nach Schlüsseln zu suchen, wird die Suche an das HSM übergeben. Wenn Sie Zertifikate im AWS CloudHSM Schlüsselspeicher speichern, überprüft der Anbieter, ob ein key pair mit dem entsprechenden Alias auf dem HSM vorhanden ist, und ordnet dann das bereitgestellte Zertifikat dem entsprechenden key pair zu.

Themen

- [Voraussetzungen für die Integration AWS CloudHSM mit Java Keytool und Jarsigner mithilfe des Client SDK 5](#)
- [Verwenden Sie den AWS CloudHSM Schlüsselspeicher mit Keytool und verwenden Sie das Client SDK 5](#)
- [Verwenden Sie den AWS CloudHSM Schlüsselspeicher mit Jarsigner und verwenden Sie das Client SDK 5](#)
- [Bekannte Probleme bei der AWS CloudHSM Integration von Java Keytool und Jarsigner mithilfe des Client SDK 5](#)

Voraussetzungen für die Integration AWS CloudHSM mit Java Keytool und Jarsigner mithilfe des Client SDK 5

Um den AWS CloudHSM Schlüsselspeicher verwenden zu können, müssen Sie zuerst das JCE SDK initialisieren und konfigurieren. AWS CloudHSM Gehen Sie dazu wie folgt vor.

Schritt 1: Installieren der JCE

Um das JCE, einschließlich der AWS CloudHSM Client-Voraussetzungen, zu installieren, folgen Sie den Schritten zur [Installation der](#) Java-Bibliothek.

Schritt 2: Hinzufügen von HSM-Anmeldeinformationen zu Umgebungsvariablen

Richten Sie Umgebungsvariablen so ein, dass sie Ihre HSM-Anmeldeinformationen enthalten.

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

Das AWS CloudHSM JCE bietet verschiedene Anmeldeoptionen. Um den AWS CloudHSM Schlüsselspeicher mit Anwendungen von Drittanbietern zu verwenden, müssen Sie die implizite Anmeldung mit Umgebungsvariablen verwenden. Wenn Sie die explizite Anmeldung über Anwendungscode verwenden möchten, müssen Sie mithilfe des AWS CloudHSM Schlüsselspeichers Ihre eigene Anwendung erstellen. Weitere Informationen finden Sie im Artikel zur [Verwendung des AWS CloudHSM Schlüsselspeichers](#).

Schritt 3: Registrieren des JCE-Providers

Gehen Sie folgendermaßen vor, um den JCE-Anbieter in der CloudProvider Java-Konfiguration zu registrieren:

1. Öffnen Sie die `java.security`-Konfigurationsdatei in Ihrer Java-Installation zur Bearbeitung.
2. Fügen Sie in der `java.security`-Konfigurationsdatei `com.amazonaws.cloudhsm.jce.provider.CloudHsmProvider` als letzten Anbieter hinzu. Wenn sich beispielsweise neun Anbieter in der `java.security`-Datei befinden, fügen Sie den folgenden Anbieter als letzten Anbieter in dem Abschnitt hinzu:

```
security.provider.10=com.amazonaws.cloudhsm.jce.provider.CloudHsmProvider
```

Note

Wenn Sie den AWS CloudHSM Anbieter mit höherer Priorität hinzufügen, kann sich dies negativ auf die Leistung Ihres Systems auswirken, da dem AWS CloudHSM Anbieter für Vorgänge, die sicher in Software ausgelagert werden können, Priorität eingeräumt wird. Es hat sich bewährt, immer den Anbieter anzugeben, den Sie für einen Vorgang

verwenden möchten, unabhängig davon, ob es sich um den AWS CloudHSM oder einen softwarebasierten Anbieter handelt.

 Note

Die Angabe von `-providerName-providerclass`, und `-providerpath` Befehlszeilenoptionen beim Generieren von Schlüsseln `keytool` mithilfe des AWS CloudHSM CloudHSM-Schlüsselspeichers kann zu Fehlern führen.

Verwenden Sie den AWS CloudHSM Schlüsselspeicher mit Keytool und verwenden Sie das Client SDK 5

[Keytool](#) ist ein beliebtes Befehlszeilenprogramm für allgemeine Schlüssel- und Zertifikataufgaben. Ein vollständiges Tutorial zu `keytool` liegt außerhalb des Umfangs der AWS CloudHSM -Dokumentation. In diesem Artikel werden die spezifischen Parameter erläutert, die Sie mit verschiedenen Keytool-Funktionen verwenden sollten, wenn Sie sie über den AWS CloudHSM Schlüsselspeicher AWS CloudHSM als Vertrauensbasis verwenden.

Wenn Sie Keytool mit dem AWS CloudHSM Schlüsselspeicher verwenden, geben Sie die folgenden Argumente für jeden Keytool-Befehl an:

Linux

```
-storetype CLOUDHSM -J-classpath< '-J/opt/cloudhsm/java/*'>
```

Windows

```
-storetype CLOUDHSM -J-classpath<' -J"C:\Program Files\Amazon\CloudHSM\java\*" '>
```

Wenn Sie mithilfe des Schlüsselspeichers eine neue Schlüsselspeicherdatei erstellen möchten, finden Sie AWS CloudHSM weitere Informationen unter. [Verwenden Sie das SDK 3 für den Client AWS CloudHSM KeyStore AWS CloudHSM](#) Um einen vorhandenen Schlüsselspeicher zu verwenden, geben Sie seinen Namen (einschließlich Pfad) mit dem Argument `-keystore` zu `keytool` an. Wenn Sie in einem Keytool-Befehl eine nicht existierende Schlüsselspeicherdatei angeben, erstellt der AWS CloudHSM Schlüsselspeicher eine neue Schlüsselspeicherdatei.

Erstellen Sie neue AWS CloudHSM Schlüssel mit Keytool

Sie können Keytool verwenden, um RSA, AES und den vom JCE SDK unterstützten DESede Schlüsseltyp zu generieren. AWS CloudHSM

Important

Ein mit Keytool generierter Schlüssel wird in der Software generiert und dann AWS CloudHSM als extrahierbarer, persistenter Schlüssel importiert.

Wir empfehlen nachdrücklich, nicht exportierbare Schlüssel außerhalb von keytool zu generieren und dann entsprechende Zertifikate in den Schlüsselspeicher zu importieren. Wenn Sie extrahierbare RSA- oder EC-Schlüssel über Keytool und Jarsigner verwenden, exportieren die Anbieter Schlüssel aus dem AWS CloudHSM und verwenden den Schlüssel dann lokal für Signaturvorgänge.

Wenn Sie mehrere Client-Instanzen mit Ihrem AWS CloudHSM Cluster verbunden haben, beachten Sie, dass der Import eines Zertifikats in den Schlüsselspeicher einer Client-Instanz die Zertifikate nicht automatisch auf anderen Client-Instances verfügbar macht. Um den Schlüssel und die zugehörigen Zertifikate auf jeder Client-Instanz zu registrieren, müssen Sie eine Java-Anwendung ausführen, wie in [the section called “Erstellen eines CSR mit Keytool”](#) beschrieben. Alternativ können Sie die erforderlichen Änderungen auf einem Client vornehmen und die resultierende Schlüsselspeicherdatei auf jede andere Client-Instanz kopieren.

Beispiel 1: Um einen symmetrischen AES-256-Schlüssel zu generieren und ihn in einer Schlüsselspeicherdatei mit dem Namen „example_keystore.store“ im Arbeitsverzeichnis zu speichern. Durch eine eindeutige Bezeichnung ersetzen. *<secret label>*

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\*"'
```

Beispiel 2: Um ein RSA 2048-Schlüsselpaar zu generieren und es in einer Schlüsselspeicherdatei mit dem Namen „example_keystore.store“ im Arbeitsverzeichnis zu speichern. Durch eine eindeutige Bezeichnung ersetzen. *<RSA key pair label>*

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\*"'
```

Sie finden eine Liste der [unterstützten Signaturalgorithmen](#) in der Java-Bibliothek.

Löschen Sie einen AWS CloudHSM Schlüssel mit Keytool

Der AWS CloudHSM Schlüsselspeicher unterstützt das Löschen von Schlüsseln nicht. Sie können Schlüssel mit der Destroyable-Methode der [Destroyable-Schnittstelle](#) löschen.

```
((Destroyable) key).destroy();
```

Generieren Sie eine AWS CloudHSM CSR mit Keytool

Sie erhalten die größte Flexibilität beim Generieren einer Zertifikatsignieranforderung (Certificate Signing Request, CSR), wenn Sie die [OpenSSL Dynamic Engine für AWS CloudHSM Client SDK 5](#) verwenden. Der folgende Befehl verwendet keytool, um eine CSR für ein Schlüsselpaar mit dem Alias, example-key-pair, zu generieren.

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

Um ein Schlüsselpaar von keytool zu verwenden, muss dieses Schlüsselpaar einen Eintrag in der angegebenen Schlüsselspeicherdatei haben. Wenn Sie ein Schlüsselpaar verwenden möchten, das außerhalb von keytool generiert wurde, müssen Sie die Schlüssel- und Zertifikatmetadaten in den Schlüsselspeicher importieren. Anweisungen zum Importieren der Keystore-Daten finden Sie unter [the section called “Verwenden Sie Keytool, um Zertifikate in den Schlüsselspeicher zu importieren”](#).

Verwenden Sie Keytool, um Zwischen- und Stammzertifikate in den Schlüsselspeicher zu importieren AWS CloudHSM

Um ein CA-Zertifikat zu importieren AWS CloudHSM, müssen Sie die Überprüfung einer vollständigen Zertifikatskette für ein neu importiertes Zertifikat aktivieren. Im Folgenden wird ein Beispielbefehl gezeigt.

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\*"'
```

Wenn Sie mehrere Client-Instances mit Ihrem AWS CloudHSM Cluster verbinden, wird durch den Import eines Zertifikats in den Schlüsselspeicher einer Client-Instanz das Zertifikat nicht automatisch auf anderen Client-Instances verfügbar. Sie müssen das Zertifikat auf jeder Client-Instanz importieren.

Verwenden Sie Keytool, um Zertifikate aus dem AWS CloudHSM Schlüsselspeicher zu löschen

Der folgende Befehl zeigt ein Beispiel für das Löschen eines AWS CloudHSM Zertifikats aus einem Java-Keytool-Schlüsselspeicher.

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\*"'
```

Wenn Sie mehrere Client-Instances mit Ihrem AWS CloudHSM Cluster verbinden, wird beim Löschen eines Zertifikats im Schlüsselspeicher einer Client-Instanz das Zertifikat nicht automatisch aus anderen Client-Instances entfernt. Sie müssen das Zertifikat auf jeder Client-Instanz löschen.

Importieren Sie ein funktionierendes Zertifikat mit AWS CloudHSM Keytool in den Schlüsselspeicher. Sobald eine Zertifikatsignieranforderung (CSR) signiert ist, können Sie sie in den AWS CloudHSM - Schlüsselspeicher importieren und mit dem entsprechenden Schlüsselpaar verknüpfen. Der folgende Befehl bietet ein Beispiel.

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\*"'
```

Der Alias sollte ein Schlüsselpaar mit einem zugeordneten Zertifikat im Schlüsselspeicher sein. Wenn der Schlüssel außerhalb von keytool oder auf einer anderen Client-Instanz generiert wird, müssen Sie zuerst die Schlüssel- und Zertifikatmetadaten in den Schlüsselspeicher importieren.

Die Zertifikatkette muss überprüfbar sein. Wenn Sie das Zertifikat nicht überprüfen können, müssen Sie möglicherweise das Signaturzertifikat (Zertifizierungsstelle) in den Schlüsselspeicher importieren, damit die Kette überprüft werden kann.

Exportieren Sie ein Zertifikat AWS CloudHSM mit Keytool

Im folgenden Beispiel wird ein Zertifikat im Binärformat X.509 generiert. Um ein menschenlesbares Zertifikat von zu exportieren AWS CloudHSM, fügen Sie `-rfc` es dem `-exportcert` Befehl hinzu.

Linux

```
$ keytool -exportcert -alias <key pair label> \  
-file my_exported_certificate.crt \  
-keystore example_keystore.store \  
-storetype CLOUDHSM \  
-rfc
```

```
-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\*"'
```

Verwenden Sie den AWS CloudHSM Schlüsselspeicher mit Jarsigner und verwenden Sie das Client SDK 5

Jarsigner ist ein beliebtes Befehlszeilenprogramm zum Signieren von JAR-Dateien mithilfe eines Schlüssels, der sicher auf einem Hardware-Sicherheitsmodul (HSM) gespeichert ist. Ein vollständiges Tutorial zu Jarsigner ist in dieser Dokumentation nicht enthalten. AWS CloudHSM In diesem Abschnitt werden die Jarsigner-Parameter erläutert, die Sie verwenden sollten, um Signaturen AWS CloudHSM als Vertrauensbasis über den Schlüsselspeicher zu signieren und zu verifizieren. AWS CloudHSM

Richten Sie AWS CloudHSM Schlüssel und Zertifikate mit Jarsigner ein

Bevor Sie AWS CloudHSM JAR-Dateien mit Jarsigner signieren können, stellen Sie sicher, dass Sie die folgenden Schritte eingerichtet oder abgeschlossen haben:

1. Folgen Sie den Anweisungen unter [AWS CloudHSM -Schlüsselspeichervoraussetzungen](#).
2. Richten Sie Ihre Signaturschlüssel und die zugehörigen Zertifikate und die Zertifikatskette ein, die im AWS CloudHSM Schlüsselspeicher der aktuellen Server- oder Client-Instanz gespeichert werden sollen. Erstellen Sie die Schlüssel auf dem AWS CloudHSM und importieren Sie dann die zugehörigen Metadaten in Ihren AWS CloudHSM Schlüsselspeicher. Informationen zum Einrichten der Schlüssel und Zertifikate mit keytool finden Sie unter [the section called "Erstellen neuer Schlüssel mit Keytool"](#). Wenn Sie mehrere Client-Instanzen verwenden, um Ihre zu signieren JARs, erstellen Sie den Schlüssel und importieren Sie die Zertifikatskette. Kopieren Sie dann die resultierende Schlüsselspeicherdatei auf jede Client-Instanz. Wenn Sie häufig neue Schlüssel generieren, ist es möglicherweise einfacher, Zertifikate einzeln in jede Client-Instanz zu importieren.
3. Die gesamte Zertifikatskette sollte überprüfbar sein. Damit die Zertifikatskette verifizierbar ist, müssen Sie möglicherweise das CA-Zertifikat und die Zwischenzertifikate zum AWS CloudHSM

Schlüsselspeicher hinzufügen. Im Codeausschnitt in [the section called “Signieren Sie eine JAR-Datei”](#) finden Sie eine Anleitung zur Verwendung von Java-Code zur Überprüfung der Zertifikatskette. Wenn Sie möchten, können Sie keytool verwenden, um Zertifikate zu importieren. Eine Anleitung zur Verwendung von keytool finden Sie unter [the section called “Verwenden Sie Keytool, um Zertifikate in den Schlüsselspeicher zu importieren”](#).

Signieren Sie eine JAR-Datei mit AWS CloudHSM und Jarsigner

Verwenden Sie den folgenden Befehl, um eine JAR-Datei mit AWS CloudHSM einem Jarsigner zu signieren:

Linux;

Für OpenJDK 8

```
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>
```

Für OpenJDK 11, OpenJDK 17 und OpenJDK 21

```
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

Für Open JDK8

```
jarsigner -keystore example_keystore.store `
```

```
-signedjar signthisclass_signed.jar `
-sigalg sha512withrsa `
-storetype CloudHSM `
-J-classpath '-JC:\Program Files\Amazon\CloudHSM\java\*;C:\Program Files\Java
\jdk1.8.0_331\lib\tools.jar' `
"-J-Djava.library.path='C:\Program Files\Amazon\CloudHSM\lib\'" `
signthisclass.jar <key pair label>
```

Für OpenJDK 11, OpenJDK 17 und OpenJDK 21

```
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>
```

Verwenden Sie den folgenden Befehl, um eine signierte JAR zu überprüfen:

Linux

Für Open JDK8

```
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>
```

Für OpenJDK 11, OpenJDK 17 und OpenJDK 21

```
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

Für OpenJDK 8

```
jarsigner -verify `
-keystore example_keystore.store `
-sigalg sha512withrsa `
-storetype CloudHSM `
-J-classpath '-JC:\Program Files\Amazon\CloudHSM\java\*;C:\Program Files\Java
\jdk1.8.0_331\lib\tools.jar' `
"-J-Djava.library.path='C:\Program Files\Amazon\CloudHSM\lib\'" `
signthisclass_signed.jar <key pair label>
```

Für OpenJDK 11, OpenJDK 17 und OpenJDK 21

```
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>
```

Bekannte Probleme bei der AWS CloudHSM Integration von Java Keytool und Jarsigner mithilfe des Client SDK 5

Die folgende Liste enthält die aktuelle Liste der bekannten Probleme bei Integrationen mit Java Keytool AWS CloudHSM und Jarsigner, die das Client SDK 5 verwenden.

1. Wir unterstützen keine EC-Schlüssel mit Keytool und Jarsigner.

Verwenden Sie Client SDK 3 zur Integration AWS CloudHSM mit Java Keytool und Jarsigner

AWS CloudHSM Der Schlüsselspeicher ist ein spezieller JCE-Schlüsselspeicher, der Zertifikate verwendet, die mit Schlüsseln auf Ihrem Hardware-Sicherheitsmodul (HSM) verknüpft sind, und zwar über Tools von Drittanbietern wie und. `keytool` `jarsigner` AWS CloudHSM speichert keine Zertifikate auf dem HSM, da es sich bei Zertifikaten um öffentliche, nicht vertrauliche Daten handelt. Der AWS CloudHSM Schlüsselspeicher speichert die Zertifikate in einer lokalen Datei und ordnet die Zertifikate den entsprechenden Schlüsseln auf Ihrem HSM zu.

Wenn Sie den AWS CloudHSM Schlüsselspeicher verwenden, um neue Schlüssel zu generieren, werden keine Einträge in der lokalen Schlüsselspeicherdatei generiert — die Schlüssel werden auf dem HSM erstellt. Wenn Sie den AWS CloudHSM -Schlüsselspeicher verwenden, um nach Schlüsseln zu suchen, wird die Suche an das HSM übergeben. Wenn Sie Zertifikate im AWS CloudHSM Schlüsselspeicher speichern, überprüft der Anbieter, ob ein key pair mit dem entsprechenden Alias auf dem HSM vorhanden ist, und ordnet dann das bereitgestellte Zertifikat dem entsprechenden key pair zu.

Themen

- [Voraussetzungen für die Integration AWS CloudHSM mit Java Keytool und Jarsigner mithilfe des Client SDK 3](#)
- [Verwenden Sie den AWS CloudHSM Schlüsselspeicher mit Keytool und verwenden Sie das Client SDK 3](#)
- [Verwenden Sie den AWS CloudHSM Schlüsselspeicher mit Jarsigner und verwenden Sie das Client SDK 3](#)
- [Bekannte Probleme bei der AWS CloudHSM Integration von Java, Keytool und Jarsigner mithilfe des Client SDK 3](#)
- [Registrieren Sie bereits vorhandene Schlüssel im Schlüsselspeicher AWS CloudHSM](#)

Voraussetzungen für die Integration AWS CloudHSM mit Java Keytool und Jarsigner mithilfe des Client SDK 3

Um den AWS CloudHSM Schlüsselspeicher verwenden zu können, müssen Sie zuerst das JCE SDK initialisieren und konfigurieren. AWS CloudHSM Gehen Sie dazu wie folgt vor.

Schritt 1: Installieren der JCE

Um das JCE, einschließlich der AWS CloudHSM Client-Voraussetzungen, zu installieren, folgen Sie den Schritten zur [Installation der](#) Java-Bibliothek.

Schritt 2: Hinzufügen von HSM-Anmeldeinformationen zu Umgebungsvariablen

Richten Sie Umgebungsvariablen so ein, dass sie Ihre HSM-Anmeldeinformationen enthalten.

```
export HSM_PARTITION=PARTITION_1
export HSM_USER=<HSM user name>
export HSM_PASSWORD=<HSM password>
```

Note

Die CloudHSM JCE bietet verschiedene Anmeldeoptionen. Um den AWS CloudHSM Schlüsselspeicher mit Anwendungen von Drittanbietern zu verwenden, müssen Sie die implizite Anmeldung mit Umgebungsvariablen verwenden. Wenn Sie die explizite Anmeldung über Anwendungscode verwenden möchten, müssen Sie mithilfe des AWS CloudHSM Schlüsselspeichers Ihre eigene Anwendung erstellen. Weitere Informationen finden Sie im Artikel zur [Verwendung des AWS CloudHSM Schlüsselspeichers](#).

Schritt 3: Registrieren Sie den JCE-Anbieter

Um den JCE-Anbieter in der CloudProvider Java-Konfiguration zu registrieren.

1. Öffnen Sie die java.security-Konfigurationsdatei in Ihrer Java-Installation zur Bearbeitung.
2. Fügen Sie in der Konfigurationsdatei `java.security com.cavium.provider.CaviumProvider` als letzten Provider hinzu. Wenn sich beispielsweise neun Provider in der Datei `java.security` befinden, fügen Sie den folgenden Provider als letzten Provider in dem Abschnitt hinzu. Das Hinzufügen des Cavium-Providers mit höherer Priorität kann sich negativ auf die Leistung Ihres Systems auswirken.

```
security.provider.10=com.cavium.provider.CaviumProvider
```


 Note

Intensive Nutzer können daran gewöhnt sein, `-providerName-`, `-providerclass-` und `-providerpath-` Befehlszeilenoptionen bei Verwendung von `keytool` anzugeben, anstatt die Sicherheitskonfigurationsdatei zu aktualisieren. Wenn Sie versuchen, beim Generieren von Schlüsseln mit dem AWS CloudHSM Schlüsselspeicher Befehlszeilenoptionen anzugeben, führt dies zu Fehlern.

Verwenden Sie den AWS CloudHSM Schlüsselspeicher mit Keytool und verwenden Sie das Client SDK 3

[Keytool](#) ist ein beliebtes Befehlszeilenprogramm für allgemeine Schlüssel- und Zertifikataufgaben auf Linux-Systemen. Ein vollständiges Tutorial zu `keytool` liegt außerhalb des Umfangs der AWS CloudHSM -Dokumentation. In diesem Artikel werden die spezifischen Parameter erläutert, die Sie mit verschiedenen Keytool-Funktionen verwenden sollten, wenn Sie sie über den AWS CloudHSM Schlüsselspeicher AWS CloudHSM als Vertrauensbasis verwenden.

Wenn Sie Keytool mit dem AWS CloudHSM Schlüsselspeicher verwenden, geben Sie die folgenden Argumente für jeden Keytool-Befehl an:

```
-storetype CLOUDHSM \  
-J-classpath '-J/opt/cloudhsm/java/*' \  
-J-Djava.library.path=/opt/cloudhsm/lib
```

Wenn Sie mithilfe des Schlüsselspeichers eine neue Schlüsselspeicherdatei erstellen möchten, finden Sie AWS CloudHSM weitere Informationen unter. [Verwenden Sie das SDK 3 für den Client AWS CloudHSM KeyStore AWS CloudHSM](#) Um einen vorhandenen Schlüsselspeicher zu verwenden, geben Sie seinen Namen (einschließlich Pfad) mit dem Argument `-keystore` zu `keytool` an. Wenn Sie in einem Keytool-Befehl eine nicht existierende Schlüsselspeicherdatei angeben, erstellt der AWS CloudHSM Schlüsselspeicher eine neue Schlüsselspeicherdatei.

Erstellen Sie neue AWS CloudHSM Schlüssel mit Keytool

Sie können Keytool verwenden, um jede Art von Schlüssel zu generieren, die vom AWS CloudHSM JCE SDK unterstützt werden. Eine vollständige Liste der Schlüssel und Längen finden Sie im Artikel [Unterstützte Schlüssel](#) in der Java-Bibliothek.

⚠ Important

Ein mit Keytool generierter Schlüssel wird in der Software generiert und dann AWS CloudHSM als extrahierbarer, persistenter Schlüssel importiert.

[Anweisungen zum Erstellen nicht extrahierbarer Schlüssel direkt auf dem Hardware-Sicherheitsmodul \(HSM\) und deren anschließende Verwendung mit Keytool oder Jarsigner finden Sie im Codebeispiel unter Registrierung bereits vorhandener Schlüssel mit dem Schlüsselspeicher. AWS CloudHSM](#) Wir empfehlen nachdrücklich, nicht exportierbare Schlüssel außerhalb von keytool zu generieren und dann entsprechende Zertifikate in den Schlüsselspeicher zu importieren. Wenn Sie extrahierbare RSA- oder EC-Schlüssel über Keytool und Jarsigner verwenden, exportieren die Anbieter Schlüssel aus dem und verwenden den Schlüssel dann lokal für Signaturvorgänge. AWS CloudHSM

Wenn mehrere Client-Instanzen mit Ihrem CloudHSM-Cluster verbunden sind, beachten Sie, dass das Importieren eines Zertifikats im Schlüsselspeicher einer Clientinstanz die Zertifikate nicht automatisch auf anderen Client-Instanzen verfügbar macht. Um den Schlüssel und die zugehörigen Zertifikate auf jeder Client-Instanz zu registrieren, müssen Sie eine Java-Anwendung ausführen, wie unter [Generieren eines CSR mit Keytool](#) beschrieben. Alternativ können Sie die erforderlichen Änderungen auf einem Client vornehmen und die resultierende Schlüsselspeicherdatei auf jede andere Client-Instanz kopieren.

Beispiel 1: Um einen symmetrischen AES-256-Schlüssel zu generieren und ihn in einer Schlüsselspeicherdatei mit dem Namen „example_keystore.store“ im Arbeitsverzeichnis zu speichern. Durch eine eindeutige Bezeichnung ersetzen. *<secret label>*

```
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/
```

Beispiel 2: Um ein RSA 2048-Schlüsselpaar zu generieren und es in einer Schlüsselspeicherdatei mit dem Namen „example_keystore.store“ im Arbeitsverzeichnis zu speichern. Durch eine eindeutige Bezeichnung ersetzen. *<RSA key pair label>*

```
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/
```

Beispiel 3: Um einen P256-ED-Schlüssel zu generieren und ihn in einer Schlüsselspeicherdatei mit dem Namen „example_keystore.store“ im Arbeitsverzeichnis zu speichern. Durch eine eindeutige Bezeichnung ersetzen. *<ec key pair label>*

```
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/
```

Sie finden eine Liste der [unterstützten Signaturalgorithmen](#) in der Java-Bibliothek.

Löschen Sie einen AWS CloudHSM Schlüssel mit Keytool

Der AWS CloudHSM Schlüsselspeicher unterstützt das Löschen von Schlüsseln nicht. Um einen Schlüssel zu löschen, müssen Sie die deleteKey Funktion AWS CloudHSM des Befehlszeilentools von verwenden [Löschen Sie einen AWS CloudHSM Schlüssel mit KMU](#).

Generieren Sie eine AWS CloudHSM CSR mit Keytool

Sie erhalten die größte Flexibilität beim Generieren einer Zertifikatsignieranforderung (Certificate Signing Request, CSR), wenn Sie die [OpenSSL Dynamic Engine für AWS CloudHSM Client SDK 5](#) verwenden. Der folgende Befehl verwendet keytool, um eine CSR für ein Schlüsselpaar mit dem Alias, example-key-pair, zu generieren.

```
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

Um ein Schlüsselpaar von keytool zu verwenden, muss dieses Schlüsselpaar einen Eintrag in der angegebenen Schlüsselspeicherdatei haben. Wenn Sie ein Schlüsselpaar verwenden möchten, das außerhalb von keytool generiert wurde, müssen Sie die Schlüssel- und Zertifikatmetadaten in den Schlüsselspeicher importieren. Anweisungen zum Importieren der Keystore-Daten finden Sie unter [Import von Zwischen- und Stammzertifikaten in AWS CloudHSM Key Store mit Keytool](#).

Verwenden Sie Keytool, um Zwischen- und Stammzertifikate in den Schlüsselspeicher zu importieren AWS CloudHSM

Um ein CA-Zertifikat zu importieren AWS CloudHSM, müssen Sie die Überprüfung einer vollständigen Zertifikatskette für ein neu importiertes Zertifikat aktivieren. Im Folgenden wird ein Beispielbefehl gezeigt.

```
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/
```

Wenn Sie mehrere Client-Instances mit Ihrem AWS CloudHSM Cluster verbinden, wird durch den Import eines Zertifikats in den Schlüsselspeicher einer Client-Instance das Zertifikat nicht automatisch auf anderen Client-Instances verfügbar gemacht. Sie müssen das Zertifikat auf jeder Client-Instanz importieren.

Verwenden Sie Keytool, um Zertifikate aus dem AWS CloudHSM Schlüsselspeicher zu löschen

Der folgende Befehl zeigt ein Beispiel für das Löschen eines AWS CloudHSM Zertifikats aus einem Java-Keytool-Schlüsselspeicher.

```
keytool -delete -alias mydomain -keystore \  
-keystore example_keystore.store \  
-storetype CLOUDHSM \  
-J-classpath '-J/opt/cloudhsm/java/*' \  
-J-Djava.library.path=/opt/cloudhsm/lib/
```

Wenn Sie mehrere Client-Instances mit Ihrem AWS CloudHSM Cluster verbinden, wird durch das Löschen eines Zertifikats im Schlüsselspeicher einer Client-Instanz das Zertifikat nicht automatisch aus anderen Client-Instances entfernt. Sie müssen das Zertifikat auf jeder Client-Instanz löschen.

Importieren Sie ein funktionierendes Zertifikat mit AWS CloudHSM Keytool in den Schlüsselspeicher

Sobald eine Zertifikatsignieranforderung (CSR) signiert ist, können Sie sie in den AWS CloudHSM - Schlüsselspeicher importieren und mit dem entsprechenden Schlüsselpaar verknüpfen. Der folgende Befehl bietet ein Beispiel.

```
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/
```

Der Alias sollte ein Schlüsselpaar mit einem zugeordneten Zertifikat im Schlüsselspeicher sein. Wenn der Schlüssel außerhalb von keytool oder auf einer anderen Client-Instanz generiert wird, müssen Sie zuerst die Schlüssel- und Zertifikatmetadaten in den Schlüsselspeicher importieren. Anweisungen zum Importieren der Zertifikatsmetadaten finden Sie im Codebeispiel unter [Registrierung bereits vorhandener Schlüssel mit dem AWS CloudHSM Schlüsselspeicher](#).

Die Zertifikatkette muss überprüfbar sein. Wenn Sie das Zertifikat nicht überprüfen können, müssen Sie möglicherweise das Signaturzertifikat (Zertifizierungsstelle) in den Schlüsselspeicher importieren, damit die Kette überprüft werden kann.

Exportieren Sie ein Zertifikat AWS CloudHSM mit Keytool

Im folgenden Beispiel wird ein Zertifikat im Binärformat X.509 generiert. Um ein menschenlesbares Zertifikat von zu exportieren AWS CloudHSM, fügen Sie -rfc es dem -exportcert Befehl hinzu.

```
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/
```

Verwenden Sie den AWS CloudHSM Schlüsselspeicher mit Jarsigner und verwenden Sie das Client SDK 3

Jarsigner ist ein beliebtes Befehlszeilenprogramm zum Signieren von JAR-Dateien mithilfe eines Schlüssels, der sicher auf einem Hardware-Sicherheitsmodul (HSM) gespeichert ist. Ein vollständiges Tutorial zu Jarsigner ist in dieser Dokumentation nicht enthalten. AWS CloudHSM In diesem Abschnitt werden die Jarsigner-Parameter erläutert, die Sie verwenden sollten, um Signaturen AWS CloudHSM als Vertrauensbasis über den Schlüsselspeicher zu signieren und zu verifizieren. AWS CloudHSM

Richten Sie AWS CloudHSM Schlüssel und Zertifikate mit Jarsigner ein

Bevor Sie AWS CloudHSM JAR-Dateien mit Jarsigner signieren können, stellen Sie sicher, dass Sie die folgenden Schritte eingerichtet oder abgeschlossen haben:

1. Folgen Sie den Anweisungen unter [AWS CloudHSM -Schlüsselspeichervoraussetzungen](#).
2. Richten Sie Ihre Signaturschlüssel und die zugehörigen Zertifikate und die Zertifikatskette ein, die im AWS CloudHSM Schlüsselspeicher der aktuellen Server- oder Client-Instanz gespeichert werden sollen. Erstellen Sie die Schlüssel auf dem AWS CloudHSM und importieren Sie dann die zugehörigen Metadaten in Ihren AWS CloudHSM Schlüsselspeicher. Verwenden Sie das Codebeispiel unter [Registrierung bereits vorhandener Schlüssel beim AWS CloudHSM Schlüsselspeicher](#), um Metadaten in den Schlüsselspeicher zu importieren. Informationen zum Einrichten der Schlüssel und Zertifikate mit keytool finden Sie unter [Erstellen Sie neue AWS CloudHSM Schlüssel mit Keytool](#). Wenn Sie mehrere Client-Instanzen verwenden, um Ihre zu signieren JARs, erstellen Sie den Schlüssel und importieren Sie die Zertifikatskette. Kopieren Sie dann die resultierende Schlüsselspeicherdatei auf jede Client-Instanz. Wenn Sie häufig neue Schlüssel generieren, ist es möglicherweise einfacher, Zertifikate einzeln in jede Client-Instanz zu importieren.
3. Die gesamte Zertifikatskette sollte überprüfbar sein. Damit die Zertifikatskette verifizierbar ist, müssen Sie möglicherweise das CA-Zertifikat und die Zwischenzertifikate zum AWS CloudHSM Schlüsselspeicher hinzufügen. Anweisungen zur [Verwendung von Java-Code zur Überprüfung der Zertifikatskette finden Sie im Codeausschnitt unter Signieren einer JAR-Datei mit AWS CloudHSM und Jarsigner](#). Wenn Sie möchten, können Sie keytool verwenden, um Zertifikate zu importieren. Anweisungen zur Verwendung von Keytool finden Sie unter [Verwenden von Keytool zum Importieren von Zwischen- und Stammzertifikaten in Key Store](#). AWS CloudHSM

Signieren Sie eine JAR-Datei mit AWS CloudHSM und Jarsigner

Verwenden Sie den folgenden Befehl, um eine JAR-Datei mit AWS CloudHSM einem Jarsigner zu signieren:

```
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>
```

Verwenden Sie den folgenden Befehl, um eine signierte JAR zu überprüfen:

```
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>
```

Bekannte Probleme bei der AWS CloudHSM Integration von Java, Keytool und Jarsigner mithilfe des Client SDK 3

Die folgende Liste enthält die aktuelle Liste der bekannten Probleme bei Integrationen mit Java Keytool AWS CloudHSM und Jarsigner, die das Client SDK 3 verwenden.

- Beim Generieren von Schlüsseln mit Keytool kann der erste Anbieter in der Anbieterkonfiguration nicht verwendet werden. CaviumProvider
- Beim Generieren von Schlüsseln mit keytool wird der erste (unterstützte) Provider in der Sicherheitskonfigurationsdatei verwendet, um den Schlüssel zu generieren. Dies ist in der Regel ein Software-Provider. Der generierte Schlüssel erhält dann einen Alias und wird während des Hinzufügens des Schlüssels als persistenter (Token-) Schlüssel in das AWS CloudHSM HSM importiert.
- Wenn Sie Keytool mit AWS CloudHSM Schlüsselspeicher verwenden -providerName-providerclass, geben Sie keine -providerpath Optionen oder in der Befehlszeile an. Geben Sie diese Optionen in der Sicherheitsanbieterdatei an, wie in den [Key Store-Voraussetzungen](#) beschrieben.

- Wenn Sie nicht extrahierbare EC-Schlüssel über Keytool und Jarsigner verwenden, muss der SunEC-Anbieter removed/disabled aus der Liste der Anbieter in der Datei java.security stammen. Wenn Sie extrahierbare EC-Schlüssel über Keytool und Jarsigner verwenden, exportieren die Anbieter Schlüsselbits aus dem HSM und verwenden den Schlüssel lokal für Signiervorgänge. AWS CloudHSM Wir empfehlen nicht, exportierbare Schlüssel mit keytool oder Jarsigner zu verwenden.

Registrieren Sie bereits vorhandene Schlüssel im Schlüsselspeicher AWS CloudHSM

Für maximale Sicherheit und Flexibilität bei Attributen und Kennzeichnungen empfehlen wir Ihnen, Ihre AWS CloudHSM Signaturschlüssel mit [key_mgmt_util](#) zu generieren. Sie können auch eine Java-Anwendung verwenden, um den Schlüssel in AWS CloudHSM zu generieren.

Der folgende Abschnitt enthält ein Codebeispiel, das zeigt, wie ein neues key pair auf dem HSM generiert und es mit vorhandenen Schlüsseln registriert wird, die in den AWS CloudHSM Schlüsselspeicher importiert wurden. Die importierten Schlüssel stehen für die Verwendung mit Tools von Drittanbietern wie keytool und Jarsigner zur Verfügung.

Wenn Sie einen bereits vorhandenen Schlüssel verwenden möchten, ändern Sie das Codebeispiel so, dass es nach einem Schlüssel nach Benennung sucht, anstatt einen neuen Schlüssel zu generieren. Ein Beispielcode für das Nachschlagen eines Schlüssels anhand seiner Bezeichnung ist im [KeyUtilitiesRunner.java-Beispiel](#) unter verfügbar. GitHub

Important

Bei der Registrierung eines Schlüssels, der in AWS CloudHSM einem lokalen Schlüsselspeicher gespeichert ist, wird der Schlüssel nicht exportiert. Wenn der Schlüssel registriert ist, registriert der Schlüsselspeicher den Alias (oder die Benennung) des Schlüssels und korreliert lokal Speicherzertifikatobjekte mit einem Schlüsselpaar auf dem AWS CloudHSM. Solange das Schlüsselpaar als nicht exportierbar erstellt wird, verlassen die Schlüsselbits das HSM nicht.

```
//  
// 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.Enumuration;

//
// 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\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());
}
}
```

Verwenden Sie das Microsoft Manifest Generation and Editing Tool (Mage.exe) mit AWS CloudHSM , um Dateien zu signieren

 Note

AWS CloudHSM unterstützt nur das 64-Bit-Mage-Tool, das im Windows SDK for .NET Framework 4.8.1 und höher enthalten ist.

Die folgenden Themen bieten einen Überblick über die Verwendung von [Mage.exe](#) mit AWS CloudHSM.

Themen

- [Schritt 1: Einrichten der Voraussetzungen](#)
- [Schritt 2: Erstellen Sie ein Signaturzertifikat](#)
- [Schritt 3: Signieren Sie eine Datei](#)

Schritt 1: Einrichten der Voraussetzungen

Um Microsoft Mage.exe mit zu verwenden AWS CloudHSM, benötigen Sie Folgendes:

- Eine EC2 Amazon-Instance, auf der ein Windows-Betriebssystem ausgeführt wird
- Eine Zertifizierungsstelle (CA), entweder selbst verwaltet oder von einem Drittanbieter
- Ein aktiver AWS CloudHSM Cluster in derselben Virtual Private Cloud (VPC) wie Ihre EC2 Instanz, mit mindestens einem HSM
- Ein Krypto-Benutzer (CU), der Schlüssel im Cluster besitzt und verwaltet AWS CloudHSM
- Eine unsignierte Datei oder eine ausführbare Datei
- Das Microsoft Windows Software Development Kit (SDK)

Um die Voraussetzungen für die Verwendung AWS CloudHSM mit Mage.exe einzurichten

1. Starten Sie eine EC2 Windows-Instanz und einen AWS CloudHSM Cluster, indem Sie den Anweisungen im Abschnitt [Erste Schritte](#) dieses Handbuchs folgen.
2. Wenn Sie Ihre eigene Windows Server-Zertifizierungsstelle hosten möchten, führen Sie die Schritte 1 und 2 [unter Konfiguration von Windows Server als Zertifizierungsstelle mit](#) aus AWS CloudHSM. Verwenden Sie andernfalls Ihre öffentlich vertrauenswürdige Drittanbieter-Zertifizierungsstelle.
3. Laden Sie das Microsoft Windows SDK for .NET Framework 4.8.1 oder höher herunter und installieren Sie es auf Ihrer EC2 Windows-Instanz:
 - [Microsoft Windows SDK 10](#)

Die `mage.exe` ausführbare Datei ist Teil der Windows SDK-Tools. Der Standardinstallationspfad ist:

```
C:\Program Files (x86)\Windows Kits\<SDK version>\bin\<version number>\x64\Mage.exe
```

Nachdem Sie diese Schritte abgeschlossen haben, können Sie das Microsoft Windows SDK, Ihren AWS CloudHSM Cluster und Ihre CA verwenden, um [ein Signaturzertifikat zu erstellen](#).

Schritt 2: Erstellen Sie ein Signaturzertifikat

Nachdem Sie das Windows SDK nun auf Ihrer EC2 Instanz installiert haben, können Sie es verwenden, um eine Zertifikatsignieranforderung (CSR) zu generieren. Die CSR ist ein unsigniertes Zertifikat, das Sie zur Unterzeichnung an Ihre Zertifizierungsstelle senden. In diesem Beispiel verwenden wir die im Windows SDK enthaltene `certreq` ausführbare Datei, um die CSR zu generieren.

Um eine CSR mit der ausführbaren Datei `certreq` zu generieren

1. Connect zu Ihrer EC2 Windows-Instanz her. Weitere Informationen finden Sie unter [Connect to Your Instance](#) im EC2 Amazon-Benutzerhandbuch.
2. Erstellen Sie eine Datei mit dem Namen `request.inf` und dem folgenden Inhalt. Ersetzen Sie die Subject Informationen durch die Details Ihrer Organisation:

```
[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
```

Eine Erklärung der einzelnen Parameter finden Sie in der [Dokumentation von Microsoft](#).

3. Führen Sie `certreq.exe` das Programm aus, um die CSR zu generieren:

```
certreq.exe -new request.inf request.csr
```

Dieser Befehl generiert ein neues key pair in Ihrem AWS CloudHSM Cluster und verwendet den privaten Schlüssel, um die CSR zu erstellen.

4. Übermitteln Sie die CSR an Ihre CA. Wenn Sie eine Windows Server-Zertifizierungsstelle verwenden, gehen Sie wie folgt vor:

- a. Öffnen Sie das CA-Tool:

```
certsrv.msc
```

- b. Klicken Sie im neuen Fenster mit der rechten Maustaste auf den Namen des CA-Servers. Wählen Sie Alle Aufgaben aus, und wählen Sie dann Neue Anfrage senden aus.
- c. Navigieren Sie zum Speicherort von `request.csr` und wählen Sie Öffnen.
- d. Erweitern Sie das Menü Server CA und navigieren Sie zum Ordner Pending Requests. Klicken Sie mit der rechten Maustaste auf die Anfrage, die Sie gerade erstellt haben, wählen Sie Alle Aufgaben und dann Issue aus.
- e. Navigieren Sie zum Ordner „Ausgestellte Zertifikate“.
- f. Wählen Sie Öffnen aus, um das Zertifikat anzuzeigen, und wählen Sie dann die Registerkarte Details aus.
- g. Wählen Sie In Datei kopieren aus, um den Assistenten für den Zertifikatsexport zu starten. Speichern Sie die DER-kodierte X.509-Datei an einem sicheren Ort als `signedCertificate.cer`.
- h. Beenden Sie das CA-Tool und führen Sie den folgenden Befehl aus, um die Zertifikatsdatei in den Personal Certificate Store in Windows zu verschieben:

```
certreq.exe -accept signedCertificate.cer
```

Sie können jetzt Ihr importiertes Zertifikat verwenden, um [eine Datei zu signieren](#).

Schritt 3: Signieren Sie eine Datei

Jetzt, da Sie über Mage.exe und Ihr importiertes Zertifikat verfügen, können Sie eine Datei signieren. Sie müssen den SHA-1-Hash oder den Fingerabdruck des Zertifikats kennen. Der Fingerabdruck stellt sicher, dass Mage.exe nur Zertifikate verwendet, die von verifiziert wurden. AWS CloudHSM In diesem Beispiel verwenden wir, PowerShell um den Hash des Zertifikats abzurufen.

So erhalten Sie den Fingerabdruck eines Zertifikats und signieren damit eine Datei:

1. Navigieren Sie zu dem Verzeichnis, das enthält `mage.exe`. Der Standardspeicherort ist:

```
C:\Program Files (x86)\Microsoft SDKs\Windows\v10.0A\bin\NETFX 4.8.1 Tools\x64
```

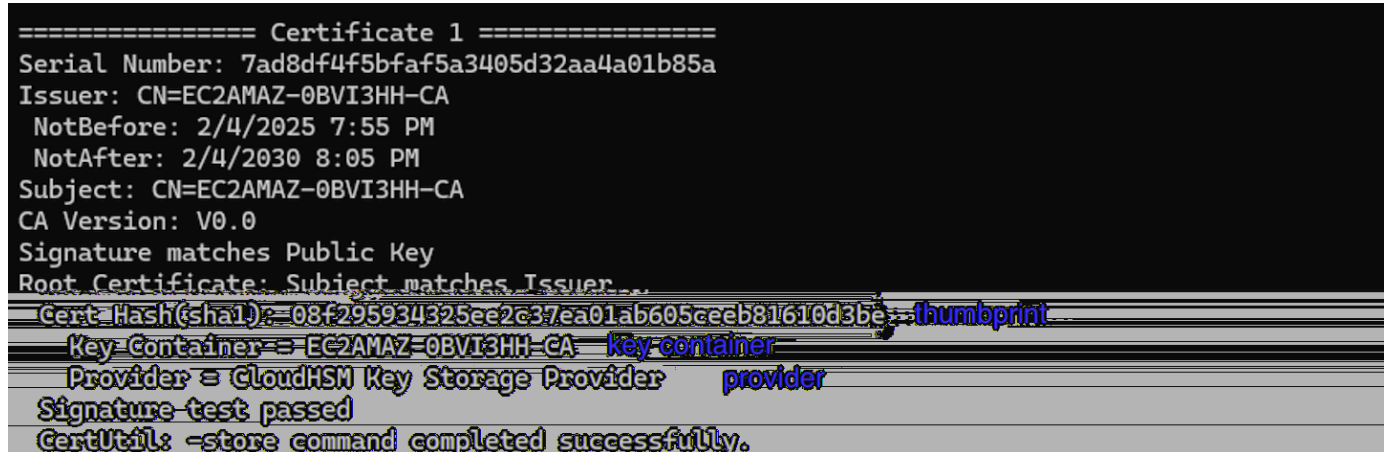
2. Führen Sie den folgenden Befehl aus, um mit Mage.exe eine Beispielanwendungsdatei zu erstellen:

```
mage.exe -New Application -ToFile C:\Users\Administrator\Desktop\sample.application
```

3. Öffnen Sie PowerShell als Administrator und führen Sie den folgenden Befehl aus:

```
Get-ChildItem -path cert:\LocalMachine\My
```

Kopieren Sie die Provider Werte ThumbprintKey Container, und aus der Ausgabe.



```

===== Certificate 1 =====
Serial Number: 7ad8df4f5bfaf5a3405d32aa4a01b85a
Issuer: CN=EC2AMAZ-0BVI3HH-CA
NotBefore: 2/4/2025 7:55 PM
NotAfter: 2/4/2030 8:05 PM
Subject: CN=EC2AMAZ-0BVI3HH-CA
CA Version: V0.0
Signature matches Public Key
Root Certificate: Subject matches Issuer...
Cert-Hash(sha1):=08f295934325ee2c37ea01ab605cee581610d3be thumbprint
Key-Container:= EC2AMAZ-0BVI3HH-CA key container
Provider = CloudHSM Key Storage Provider provider
Signature test passed
CertUtil: -store command completed successfully.
  
```

4. Signieren Sie Ihre Datei, indem Sie den folgenden Befehl ausführen:

```
mage.exe -Sign -CertHash <thumbprint> -KeyContainer <keycontainer> -
CryptoProvider <CloudHSM Key Storage Provider/Cavium Key Storage Provider> C:\Users
\Administrator\Desktop\<sample.application>
```

Wenn der Befehl erfolgreich ist, wird eine Erfolgsmeldung PowerShell zurückgegeben.

5. Verwenden Sie den folgenden Befehl, um die Signatur der Datei zu überprüfen:

```
mage.exe -Verify -CryptoProvider <CloudHSM Key Storage Provider/Cavium Key Storage
Provider> C:\Users\Administrator\Desktop\<sample.application>
```

Integrationen anderer Drittanbieter mit AWS CloudHSM

Mehrere Drittanbieter bieten Unterstützung AWS CloudHSM als Vertrauensbasis an. Dies bedeutet, dass Sie beim Erstellen und Speichern der zugrunde liegenden Schlüssel in Ihrem CloudHSM-Cluster eine Software-Lösung Ihrer Wahl nutzen können. Dadurch AWS kann sich Ihr Workload auf die

Vorteile von CloudHSM in Bezug auf Latenz, Verfügbarkeit, Zuverlässigkeit und Elastizität verlassen. Die folgende Liste enthält Drittanbieter, die CloudHSM unterstützen.

 Note

AWS befürwortet keine Drittanbieter und steht auch nicht für diese ein.

- [Hashicorp Vault](#) ist ein Verschlüsselungsverwaltungs-Tool, das dafür entwickelt wurde, Zusammenarbeit und Governance organisationsübergreifend zu ermöglichen. Es unterstützt AWS Key Management Service und dient AWS CloudHSM als Vertrauensbasis für zusätzlichen Schutz.
- [Thycotic Secrets Server](#) hilft Kunden, vertrauliche Anmeldeinformationen über privilegierte Konten hinweg zu verwalten. Es unterstützt AWS CloudHSM als Vertrauensbasis.
- Mit dem [KMIP-Adapter von P6R können Sie Ihre AWS CloudHSM Instances über eine Standard-KMIP-Schnittstelle](#) nutzen.
- [PrimeKey EJBCA](#) ist eine beliebte Open-Source-Lösung für PKI. Es ermöglicht Ihnen, Schlüsselpaare sicher mit zu erstellen und zu speichern. AWS CloudHSM
- [Box KeySafe](#) bietet vielen Unternehmen mit strengen Sicherheits-, Datenschutz- und Compliance-Anforderungen die Verwaltung von Verschlüsselungsschlüsseln für Cloud-Inhalte. Kunden können KeySafe Schlüssel zusätzlich direkt AWS Key Management Service oder indirekt AWS CloudHSM über den AWS KMS Custom Key Store sichern.
- [Insyde Software](#) unterstützt AWS CloudHSM Firmware-Signaturen als Vertrauensbasis.
- [F5 BIG-IP LTM](#) unterstützt AWS CloudHSM als Vertrauensbasis.
- [Cloudera Navigator Key HSM](#) gibt Ihnen die Möglichkeit, Ihren CloudHSM-Cluster zum Erstellen und Speichern von Schlüsseln für Cloudera Navigator Key Trustee Server zu verwenden.
- Die [Venafi Trust Protection Platform](#) bietet umfassendes Maschinenidentitätsmanagement für TLS, SSH und Codesignatur mit AWS-CloudHSM-Schlüsselgenerierung und -schutz.

Überwachung AWS CloudHSM

Zusätzlich zu den im Client SDK integrierten Protokollierungsfunktionen können Sie auch Amazon CloudWatch Logs und Amazon CloudWatch zur Überwachung verwenden AWS CloudTrail AWS CloudHSM.

Client-SDK-Protokolle

Verwenden Sie die Client-SDK-Protokollierung, um die Diagnose- und Fehlerbehebungsinformationen der von Ihnen erstellten Anwendungen zu überwachen.

CloudTrail

Wird verwendet, CloudTrail um alle API-Aufrufe in Ihrem AWS Konto zu überwachen, einschließlich der Aufrufe, die Sie zum Erstellen und Löschen von Clustern, Hardware-Sicherheitsmodulen (HSM) und Ressourcen-Tags tätigen.

CloudWatch Logs

Verwenden Sie CloudWatch Logs, um die Logs Ihrer HSM-Instances zu überwachen. Dazu gehören Ereignisse zum Erstellen und Löschen von HSM-Benutzern, zum Ändern von Benutzerkennwörtern, zum Erstellen und Löschen von Schlüsseln und mehr.

CloudWatch

Verwenden Sie diese CloudWatch Option, um den Zustand Ihres Clusters in Echtzeit zu überwachen.

Themen

- [Arbeiten mit AWS CloudHSM Client-SDK-Protokollen](#)
- [Arbeiten mit AWS CloudTrail und AWS CloudHSM](#)
- [Arbeiten mit Amazon CloudWatch Logs und AWS CloudHSM Audit Logs](#)
- [CloudWatch Metriken abrufen für AWS CloudHSM](#)

Arbeiten mit AWS CloudHSM Client-SDK-Protokollen

Sie können vom Client-SDK generierte Protokolle abrufen. AWS CloudHSM bietet eine Implementierung der Protokollierung mit Client SDK 3 und Client SDK 5.

Themen

- [Protokollierung im Client-SDK 5](#)
- [Protokollierung im Client-SDK 3](#)

Protokollierung im Client-SDK 5

Die Protokolle des Client-SDK 5 enthalten Informationen für jede Komponente in einer nach der Komponente benannten Datei. Sie können das Configure-Tool für Client-SDK 5 verwenden, um die Protokollierung für jede Komponente zu konfigurieren.

Wenn Sie keinen Speicherort für die Datei angeben, schreibt das System Protokolle an den Standardspeicherort:

AWS CloudHSM CLI

- Linux

```
/opt/cloudhsm/run/cloudhsm-cli.log
```

Windows

```
C:\Program Files\Amazon\CloudHSM\cloudhsm-cli.log
```

PKCS #11 library

- Linux

```
/opt/cloudhsm/run/cloudhsm-pkcs11.log
```

Windows

```
C:\Program Files\Amazon\CloudHSM\cloudhsm-pkcs11.log
```

OpenSSL Dynamic Engine

- Linux

```
stderr
```

JCE provider

- Linux

```
/opt/cloudhsm/run/cloudhsm-jce.log
```

Windows

```
C:\Program Files\Amazon\CloudHSM\cloudhsm-jce.log
```

Key Storage Provider

- Windows

```
C:\Program Files\Amazon\CloudHSM\cloudhsm-ksp.log
```

OpenSSL Dynamic Engine Provider

- Linux

```
stderr
```

Informationen zur Konfiguration der Protokollierung für das Client-SDK 5 finden Sie im [Client-SDK-5-Configure-Tool](#)

Protokollierung im Client-SDK 3

Die Protokolle des Client-SDK 3 enthalten detaillierte Informationen vom AWS CloudHSM Client-Daemon. Der Speicherort der Protokolle hängt vom Betriebssystem der EC2 Amazon-Client-Instance ab, auf der Sie den Client-Daemon ausführen.

Amazon Linux

In Amazon Linux werden die AWS CloudHSM Client-Logs in die Datei mit dem Namen `geschrieben/opt/cloudhsm/run/cloudhsm_client.log`. Sie können `logrotate` oder ein ähnliches Tool verwenden, um diese Protokolle zu rotieren und zu verwalten.

Amazon Linux 2

In Amazon Linux 2 werden die AWS CloudHSM Client-Protokolle gesammelt und im Journal gespeichert. Sie können `journalctl` verwenden, um diese Protokolle anzuzeigen und zu verwalten. Verwenden Sie beispielsweise den folgenden Befehl, um die AWS CloudHSM Client-Protokolle anzuzeigen.

```
journalctl -f -u cloudhsm-client
```

CentOS 7

In CentOS 7 werden die AWS CloudHSM Client-Protokolle gesammelt und im Journal gespeichert. Sie können `journalctl` verwenden, um diese Protokolle anzuzeigen und zu verwalten. Verwenden Sie beispielsweise den folgenden Befehl, um die AWS CloudHSM Client-Protokolle anzuzeigen.

```
journalctl -f -u cloudhsm-client
```

CentOS 8

In CentOS 8 werden die AWS CloudHSM Client-Protokolle gesammelt und im Journal gespeichert. Sie können `journalctl` verwenden, um diese Protokolle anzuzeigen und zu verwalten. Verwenden Sie beispielsweise den folgenden Befehl, um die AWS CloudHSM Client-Protokolle anzuzeigen.

```
journalctl -f -u cloudhsm-client
```

RHEL 7

In Red Hat Enterprise Linux 7 werden die AWS CloudHSM Client-Logs gesammelt und im Journal gespeichert. Sie können `journalctl` verwenden, um diese Protokolle anzuzeigen und zu verwalten. Verwenden Sie zum Beispiel den folgenden Befehl, um die AWS CloudHSM Client-Logs anzusehen.

```
journalctl -f -u cloudhsm-client
```

RHEL 8

In Red Hat Enterprise Linux 8 werden die AWS CloudHSM Client-Logs gesammelt und im Journal gespeichert. Sie können `journalctl` verwenden, um diese Protokolle anzuzeigen und zu verwalten. Verwenden Sie zum Beispiel den folgenden Befehl, um die AWS CloudHSM Client-Logs anzusehen.

```
journalctl -f -u cloudhsm-client
```

Ubuntu 16.04

In Ubuntu 16.04 werden die AWS CloudHSM Client-Protokolle gesammelt und im Journal gespeichert. Sie können `journalctl` verwenden, um diese Protokolle anzuzeigen und zu verwalten. Verwenden Sie beispielsweise den folgenden Befehl, um die AWS CloudHSM Client-Protokolle anzuzeigen.

```
journalctl -f -u cloudhsm-client
```

Ubuntu 18.04

In Ubuntu 18.04 werden die AWS CloudHSM Client-Protokolle gesammelt und im Journal gespeichert. Sie können `journalctl` verwenden, um diese Protokolle anzuzeigen und zu verwalten. Verwenden Sie beispielsweise den folgenden Befehl, um die AWS CloudHSM Client-Protokolle anzuzeigen.

```
journalctl -f -u cloudhsm-client
```

Windows

- Für Windows-Client 1.1.2 und höher:

AWS CloudHSM Client-Protokolle werden in eine `cloudhsm-kps.log` Datei im AWS CloudHSM Programmdateiordner (`C:\Program Files\Amazon\CloudHSM\`) geschrieben. Jedem Namen der Protokolldatei wird ein Zeitstempel angehängt, der angibt, wann der AWS CloudHSM Client gestartet wurde.

- Für Windows-Client 1.1.1 und früher:

Die Client-Protokolle werden nicht in eine Datei geschrieben. Die Protokolle werden in der Befehlszeile oder in dem PowerShell Fenster angezeigt, in dem Sie den Client gestartet haben.

AWS CloudHSM

Arbeiten mit AWS CloudTrail und AWS CloudHSM

AWS CloudHSM ist in einen Dienst integriert AWS CloudTrail, der eine Aufzeichnung der Aktionen bereitstellt, die von einem Benutzer, einer Rolle oder einem AWS Dienst in ausgeführt wurden AWS CloudHSM. CloudTrail erfasst alle API-Aufrufe AWS CloudHSM als Ereignisse. Zu den erfassten Aufrufen gehören Aufrufe von der AWS CloudHSM Konsole und Codeaufrufen für die AWS CloudHSM API-Operationen. Wenn Sie einen Trail erstellen, können Sie die kontinuierliche Bereitstellung von CloudTrail Ereignissen an einen Amazon S3 S3-Bucket aktivieren, einschließlich Ereignissen für AWS CloudHSM. Wenn Sie keinen Trail konfigurieren, können Sie die neuesten Ereignisse trotzdem in der CloudTrail Konsole im Ereignisverlauf anzeigen. Anhand der von gesammelten Informationen können Sie die Anfrage ermitteln CloudTrail, an die die Anfrage gestellt wurde AWS CloudHSM, die IP-Adresse, von der aus die Anfrage gestellt wurde, wer die Anfrage gestellt hat, wann sie gestellt wurde, und weitere Details.

Weitere Informationen CloudTrail dazu finden Sie im [AWS CloudTrail Benutzerhandbuch](#). Eine vollständige Liste der AWS CloudHSM API-Operationen finden Sie unter [Aktionen](#) in der AWS CloudHSM API-Referenz.

AWS CloudHSM Informationen in CloudTrail

CloudTrail ist in Ihrem AWS Konto aktiviert, wenn Sie das Konto erstellen. Wenn eine Aktivität in stattfindet AWS CloudHSM, wird diese Aktivität zusammen mit anderen CloudTrail AWS Serviceereignissen im Ereignisverlauf in einem Ereignis aufgezeichnet. Sie können die neusten Ereignisse in Ihr AWS -Konto herunterladen und dort suchen und anzeigen. Weitere Informationen finden Sie unter [Ereignisse mit CloudTrail Ereignisverlauf anzeigen](#).

Für eine fortlaufende Aufzeichnung der Ereignisse in Ihrem AWS Konto, einschließlich der Ereignisse für AWS CloudHSM, erstellen Sie einen Trail. Ein Trail ermöglicht CloudTrail die Übermittlung von Protokolldateien an einen Amazon S3 S3-Bucket. Wenn Sie einen Pfad in der Konsole anlegen, gilt dieser für alle AWS-Regionen. Der Trail protokolliert Ereignisse aus allen Regionen der AWS Partition und übermittelt die Protokolldateien an den von Ihnen angegebenen Amazon S3 S3-Bucket. Darüber hinaus können Sie andere AWS Dienste konfigurieren, um die in den CloudTrail Protokollen

gesammelten Ereignisdaten weiter zu analysieren und darauf zu reagieren. Weitere Informationen finden Sie hier:

- [Übersicht zum Erstellen eines Trails](#)
- [CloudTrail Unterstützte Dienste und Integrationen](#)
- [Konfiguration von Amazon SNS SNS-Benachrichtigungen für CloudTrail](#)
- [Empfangen von CloudTrail Protokolldateien aus mehreren Regionen](#) und [Empfangen von CloudTrail Protokolldateien von mehreren Konten](#)

CloudTrail protokolliert alle AWS CloudHSM Vorgänge, einschließlich schreibgeschützter Operationen wie `DescribeClusters` und `ListTags` und Verwaltungsvorgänge wie `InitializeClusterCreateHsm`, und `DeleteBackup`

Jeder Ereignis- oder Protokolleintrag enthält Informationen zu dem Benutzer, der die Anforderung generiert hat. Die Identitätsinformationen unterstützen Sie bei der Ermittlung der folgenden Punkte:

- Ob die Anfrage mit Root- oder AWS Identity and Access Management (IAM-) Benutzeranmeldedaten gestellt wurde.
- Gibt an, ob die Anforderung mit temporären Sicherheitsanmeldeinformationen für eine Rolle oder einen Verbundbenutzer gesendet wurde.
- Ob die Anfrage von einem anderen AWS Dienst gestellt wurde.

Weitere Informationen finden Sie unter [CloudTrail userIdentity-Element](#).

AWS CloudHSM Logdateieinträge verstehen

Ein Trail ist eine Konfiguration, die die Übertragung von Ereignissen als Protokolldateien an einen von Ihnen angegebenen Amazon S3 S3-Bucket ermöglicht. CloudTrail Protokolldateien enthalten einen oder mehrere Protokolleinträge. Ein Ereignis stellt eine einzelne Anforderung aus einer beliebigen Quelle dar und enthält Informationen über die angeforderte Aktion, Datum und Uhrzeit der Aktion, Anforderungsparameter usw. CloudTrail Protokolldateien sind kein geordneter Stack-Trace der öffentlichen API-Aufrufe, sodass sie nicht in einer bestimmten Reihenfolge angezeigt werden.

Das folgende Beispiel zeigt einen CloudTrail Protokolleintrag, der die AWS CloudHSM `CreateHsm` Aktion demonstriert.

```
{
```

```

"eventVersion": "1.05",
"userIdentity": {
  "type": "AssumedRole",
  "principalId": "AR0AJZVM5NEGZSTCITAMM:ExampleSession",
  "arn": "arn:aws:sts::111122223333:assumed-role/AdminRole/ExampleSession",
  "accountId": "111122223333",
  "accessKeyId": "ASIAIY22AX6VRYNBJSA",
  "sessionContext": {
    "attributes": {
      "mfaAuthenticated": "false",
      "creationDate": "2017-07-11T03:48:44Z"
    },
    "sessionIssuer": {
      "type": "Role",
      "principalId": "AR0AJZVM5NEGZSTCITAMM",
      "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"  
}
```

Arbeiten mit Amazon CloudWatch Logs und AWS CloudHSM Audit Logs

Wenn ein HSM in Ihrem Konto einen Befehl von den AWS CloudHSM [Befehlszeilentools](#) oder [Softwarebibliotheken](#) empfängt, zeichnet es die Ausführung des Befehls in Form eines Auditprotokolls auf. Die HSM-Audit-Protokolle enthalten alle Client-initiierten [Verwaltungsbefehle](#), einschließlich derjenigen, die das HSM erstellen und löschen, die in das HSM ein- und ausloggen sowie die Benutzer und Schlüssel verwalten. Diese Protokolle bieten eine zuverlässige Aufzeichnung von Aktionen, die den Zustand des HSM verändert haben.

AWS CloudHSM sammelt Ihre HSM-Prüfprotokolle und sendet sie in Ihrem Namen an [Amazon CloudWatch Logs](#). Sie können die Funktionen von CloudWatch Logs verwenden, um Ihre AWS CloudHSM Audit-Logs zu verwalten, einschließlich der Suche und Filterung der Logs und des Exports von Protokolldaten nach Amazon S3. Sie können mit Ihren HSM-Prüfprotokollen in der [CloudWatch Amazon-Konsole](#) arbeiten oder die CloudWatch Logs-Befehle in den Logs [AWS CLI](#) und [CloudWatch Logs SDKs](#) verwenden.

Themen

- [So funktioniert die HSM-Auditprotokollierung](#)
- [AWS CloudHSM Audit-Logs in CloudWatch Logs anzeigen](#)
- [Interpretation von AWS CloudHSM Audit-Logs](#)
- [AWS CloudHSM Referenz zum Auditprotokoll](#)

So funktioniert die HSM-Auditprotokollierung

Die Auditprotokollierung wird automatisch in allen AWS CloudHSM Clustern aktiviert. Sie kann nicht deaktiviert oder ausgeschaltet werden, und keine Einstellungen können AWS CloudHSM verhindern, dass die Protokolle in Logs exportiert werden. CloudWatch Jedes Protokollereignis verfügt über eine Sequenznummer, die Zeitstempel und die Reihenfolge der Ereignisse angibt und Ihnen hilft, etwaige Manipulationsversuche zu erkennen.

Jede HSM-Instance generiert ihre eigenen Protokolle. Es ist wahrscheinlich HSMs, dass sich die Audit-Logs verschiedener Protokolle unterscheiden, auch wenn sie sich im selben Cluster befinden. Beispiel: nur das erste HSM in jedem Cluster zeichnet die Initialisierung des HSM auf. Initialisierungsereignisse erscheinen nicht in den Protokollen von Dateien, die aus HSMs Backups geklont wurden. Ebenso zeichnet das HSM, das die Schlüssel generiert, beim Anlegen eines Schlüssels ein Ereignis zur Schlüsselerzeugung auf. Die anderen Mitglieder HSMs des Clusters zeichnen ein Ereignis auf, wenn sie den Schlüssel per Synchronisation erhalten.

AWS CloudHSM sammelt die Protokolle und veröffentlicht sie in den CloudWatch Protokollen in Ihrem Konto. Um in Ihrem Namen mit dem CloudWatch Logs-Service zu kommunizieren, AWS CloudHSM verwendet eine [dienstbezogene Rolle](#). Die der Rolle zugeordnete IAM-Richtlinie ermöglicht es, nur die Aufgaben auszuführen AWS CloudHSM , die zum Senden der Audit-Logs an CloudWatch Logs erforderlich sind.

 Important

Wenn Sie einen Cluster vor dem 20. Januar 2018 angelegt haben und noch keine angehängte, servicegebundene Rolle angelegt haben, müssen Sie diese manuell erstellen. Dies ist erforderlich CloudWatch , um Audit-Logs von Ihrem AWS CloudHSM Cluster zu empfangen. Weitere Informationen über die Erstellung von dienstverknüpften Rollen finden Sie unter [Grundlegendes zu dienstverknüpften Rollen](#) sowie unter [Erstellen einer dienstverknüpften Rolle](#) im IAM-Benutzerhandbuch.

AWS CloudHSM Audit-Logs in CloudWatch Logs anzeigen

Amazon CloudWatch Logs organisiert die Audit-Logs in Protokollgruppen und innerhalb einer Protokollgruppe in Log-Streams. Jeder Protokolleintrag ist ein Ereignis. AWS CloudHSM erstellt eine Protokollgruppe für jeden Cluster und einen Protokollstream für jedes HSM im Cluster. Sie müssen keine CloudWatch Protokollkomponenten erstellen oder Einstellungen ändern.

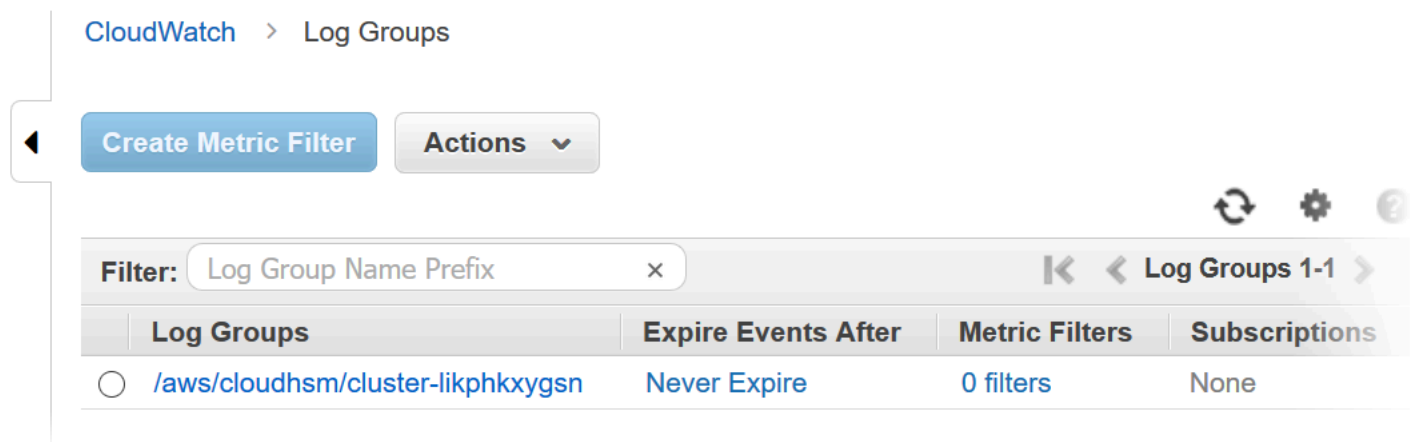
- Der Name der Protokollgruppe ist `/aws/cloudhsm/<cluster ID>`; zum Beispiel `/aws/cloudhsm/cluster-likphkxygsn`. Wenn Sie den Namen der Protokollgruppe in einem PowerShell Befehl AWS CLI oder verwenden, achten Sie darauf, ihn in doppelte Anführungszeichen zu setzen.
- Der Name des Protokollstreams ist die HSM-ID; zum Beispiel `hsm-nwbbiqbj4jk`.

Im Allgemeinen gibt es für jedes HSM einen Protokollstream. Jede Aktion, die die HSM-ID ändert, z.B. wenn ein HSM ausfällt und ersetzt wird, erzeugt jedoch einen neuen Protokollstream.

Weitere Informationen zu CloudWatch Logs-Konzepten finden Sie unter [Konzepte](#) im Amazon CloudWatch Logs-Benutzerhandbuch.

[Sie können die Audit-Logs für ein HSM auf der Seite CloudWatch Logs im AWS-Managementkonsole, den CloudWatch Logs-Befehlen in den AWS CLI, den CloudWatch PowerShellLogs-Cmdlets oder den Logs einsehen. CloudWatch SDKs](#) Anweisungen finden Sie unter [Protokolldaten anzeigen](#) im Amazon CloudWatch Logs-Benutzerhandbuch.

In der folgenden Abbildung wird beispielsweise die Protokollgruppe für den `cluster-likphkxygsn`-Cluster in der AWS-Managementkonsole angezeigt.



Wenn Sie den Namen der Cluster-Protokollgruppe wählen, können Sie den Log-Stream für jeden HSMs im Cluster anzeigen. Die folgende Abbildung zeigt die Log-Streams für die HSMs im `cluster-likphkxygsn` Cluster.

CloudWatch > Log Groups > Streams for /aws/cloudhsm/cluster-likphkxygsn

Search Log Group Create Log Stream Delete Log Stream

Filter: Log Stream Name Prefix x Log Streams 1-2

☐	Log Streams	Last Event Time
☐	hsm-aht4p3sgs3c	2017-12-28 06:12 UTC-8
☐	hsm-xkvjp4wk5o3	2017-12-28 06:12 UTC-8

Wenn Sie einen HSM-Protokollstreamnamen auswählen, können Sie die Ereignisse im Auditprotokoll anzeigen. So ist beispielsweise dieses Ereignis, das eine Sequenznummer von 0x0 und einen Opcode von CN_INIT_TOKEN hat, typischerweise das erste Ereignis für das erste HSM in jedem Cluster. Es zeichnet die Initialisierung des HSM im Cluster auf.

Filter events

Time (UTC +00:00)	Message
2017-12-19	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)

Sie können all die vielen Funktionen in CloudWatch Logs verwenden, um Ihre Audit-Logs zu verwalten. Beispiel: Sie können die Funktion Ereignisse filtern nutzen, um einen speziellen Text in einem Ereignis zu suchen, wie CN_CREATE_USER Opcode.

Um alle Ereignisse zu finden, die den angegebenen Text nicht enthalten, fügen Sie ein Minuszeichen (-) vor dem Text ein. Wenn Sie beispielsweise Ereignisse suchen möchten, die CN_CREATE_USER nicht enthalten, geben Sie -CN_CREATE_USER ein.

CN_CREATE_USER	
Time (UTC +00:00)	Message
2017-12-20	
No older events	
▼ 00:04:53	Time: 12/20/17 00:04:53.635826, u
Time: 12/20/17 00:04:53.635826, usecs:1513728293635826 Sequence No : 0x13a Reboot counter : 0xe8 Command Type(hex) : CN_MGMT_CMD (0x0) Opcode : CN_CREATE_USER (0x3) Session Handle : 0x1014006 Response : 0:HSM Return: SUCCESS Log type : MGMT_USER_DETAILS_LOG (2) User Name : testuser User Type : CN_CRYPTO_USER (1)	

Interpretation von AWS CloudHSM Audit-Logs

Die Ereignisse in den HSM-Audit-Protokollen haben standardisierte Felder. Einige Ereignistypen umfassen weitere Felder, die nützliche Informationen über das Ereignis aufzeichnen. Beispiel: Benutzeranmeldungs- und Benutzerverwaltungsereignisse enthalten den Benutzernamen und Typ des Benutzers. Key-Management-Befehle enthalten das Schlüssel-Handle.

Einige der Felder liefern besonders wichtige Informationen. Opcode identifiziert den Verwaltungsbefehl, der aufgezeichnet wird. Sequence No identifiziert ein Ereignis im Protokollstream und gibt die Reihenfolge an, in der es aufgenommen wurde.

Beispiel: Das folgende Beispielergebnis ist das zweite Ereignis (Sequence No: 0x1) im Protokollstream für ein HSM. Es zeigt, wie das HSM einen Passwort-Verschlüsselungsschlüssel erzeugt, der Teil seiner Startroutine ist.

```
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)
```

Die folgenden Felder sind allen AWS CloudHSM Ereignissen im Auditprotokoll gemeinsam.

Zeit

Die Zeit, zu der das Ereignis in der UTC-Zeitzone aufgetreten ist. Die Zeit wird als menschenlesbare Zeit und Unix-Zeit in Mikrosekunden angezeigt.

Neustart des Zählers

Ein 32-Bit persistenter Ordinalzähler, der beim Neustart der HSM-Hardware erhöht wird.

Alle Ereignisse in einem Protokollstream haben den gleichen Neustartzählerwert. Der Neustartzähler ist jedoch möglicherweise nicht eindeutig für einen Protokollstream, da er sich zwischen verschiedenen HSM-Instances im selben Cluster unterscheiden kann.

Sequenz Nr.

Ein 64-Bit-Ordinalzähler, der bei jedem Protokollereignis erhöht wird. Das erste Ereignis in jedem Protokollstream hat eine Sequenznummer von 0x0. Es dürfen keine Lücken in den Sequence No-Werten vorhanden sein. Die Sequenznummer ist nur in einem Protokollstream eindeutig.

Befehlstyp

Ein hexadezimaler Wert, der die Kategorie des Befehls repräsentiert. Befehle im AWS CloudHSM -Protokollstream haben einen Befehlstyp CN_MGMT_CMD (0x0) oder CN_CERT_AUTH_CMD (0x9).

Opcode

Identifiziert den ausgeführten Verwaltungsbefehl. Eine Liste der Opcode Werte in den AWS CloudHSM Audit-Logs finden Sie unter [AWS CloudHSM Referenz zum Auditprotokoll](#).

Session-Handle

Identifiziert die Sitzung, in der der Befehl ausgeführt und das Ereignis protokolliert wurde.

Antwort

Zeichnet die Antwort auf den Verwaltungsbefehl auf. Sie können das Response-Feld nach den Werten SUCCESS und ERROR durchsuchen.

Protokolltyp

Gibt den Protokolltyp des Protokolls an, in dem AWS CloudHSM der Befehl aufgezeichnet wurde.

- MINIMAL_LOG_ENTRY (0)
- MGMT_KEY_DETAILS_LOG (1)
- MGMT_USER_DETAILS_LOG (2)
- GENERIC_LOG

Beispiele für Audit-Protokollereignisse

Die Ereignisse in einem Protokollstream zeichnen die Historie des HSM von der Erstellung bis zum Löschen auf. Sie können das Protokoll verwenden, um den Lebenszyklus Ihres Computers zu überprüfen HSMs und Einblicke in dessen Betrieb zu erhalten. Wenn Sie die Ereignisse interpretieren, beachten Sie den Opcode, der den Befehl oder die Aktion zur Verwaltung anzeigt, und die Sequence No, die die Reihenfolge der Ereignisse angibt.

Themen

- [Beispiel: Initialisieren des ersten HSM in einem Cluster](#)
- [An- und Abmeldeereignisse](#)
- [Beispiel: Erstellen und Löschen von Benutzern](#)
- [Beispiel: Erstellen und Löschen eines Schlüsselpaares](#)
- [Beispiel: Generieren und Synchronisieren eines Schlüssels](#)
- [Beispiel: Exportieren eines Schlüssels](#)
- [Beispiel: Importieren eines Schlüssels](#)
- [Beispiel: Freigeben eines Schlüssels und Aufheben der Freigabe](#)

Beispiel: Initialisieren des ersten HSM in einem Cluster

Der Audit-Log-Stream für das erste HSM in jedem Cluster unterscheidet sich erheblich von den Log-Streams anderer HSMs HSMs im Cluster. Das Auditprotokoll für das erste HSM in jedem Cluster zeichnet seine Erstellung und Initialisierung auf. Die Protokolle der weiteren Mitglieder HSMs des Clusters, die aus Backups generiert werden, beginnen mit einem Anmeldeereignis.

⚠ Important

Die folgenden Initialisierungseinträge werden nicht in den CloudWatch Protokollen von Clustern angezeigt, die vor der Veröffentlichung der CloudHSM-Audit-Logging-Funktion (30. August 2018) initialisiert wurden. Weitere Informationen finden Sie unter [Dokumentenhistorie](#).

Die folgenden Beispielergebnisse erscheinen im Protokollstream für den ersten HSM in einem Cluster. Das erste Ereignis im Protokoll – das mit Sequence No 0x0 – entspricht dem Befehl für die Initialisierung des HSM (CN_INIT_TOKEN). Die Antwort (Response : 0: HSM Return: SUCCESS) zeigt an, dass der Befehl erfolgreich war.

```
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)
```

Das zweite Ereignis in diesem exemplarischen Protokollstream (Sequence No 0x1) zeichnet den Befehl zum Erstellen des Kennwortverschlüsselungsschlüssels auf, den das HSM verwendet (CN_GEN_PSWD_ENC_KEY).

Dies ist eine typische Startsequenz für das erste HSM in jedem Cluster. Da sich HSMs im selben Cluster nacheinander Klone des ersten Clusters befinden, verwenden sie denselben Kennwortverschlüsselungsschlüssel.

```
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)
```

Das dritte Ereignis in diesem beispielhaften Protokollstream (Sequence No 0x2) ist die Erstellung des Benutzers [Appliance-Benutzer \(AU\)](#). Dabei handelt es sich um den AWS CloudHSM -

Service. Ereignisse, an denen HSM-Benutzer beteiligt sind, enthalten zusätzliche Felder für den Benutzernamen und den Benutzertyp.

```
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)
```

Das vierte Ereignis in diesem Beispiel eines Protokollstreams (Sequence No 0x3) zeichnet das CN_INIT_DONE-Ereignis, das die Initialisierung des HSM abschließt.

```
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)
```

Sie können den verbleibenden Ereignissen in der Startsequenz folgen. Diese Ereignisse können mehrere An- und Abmeldeereignisse und die Generierung des Schlüsselverschlüsselungscode (Key Encryption Key, KEK) umfassen. Das folgende Ereignis zeichnet den Befehl auf, der das Passwort des [Precrypto Officer \(PRECO\)](#) ändert. Dieser Befehl aktiviert den Cluster.

```
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_CRYPT0_PRE_OFFICER (6)
```

An- und Abmeldeereignisse

Beachten Sie die Ereignisse bei der Interpretation Ihres Audit-Protokolls, die das Ein- und Ausloggen von Benutzern in das HSM repräsentieren. Diese Ereignisse helfen Ihnen zu bestimmen, welcher Benutzer für die Verwaltungsbefehle verantwortlich ist, die in der Reihenfolge zwischen dem Anmelde- und dem Abmeldebefehl erscheinen.

Dieser Protokolleintrag zeichnet beispielsweise eine Anmeldung durch einen Verschlüsselungsverantwortlichen mit dem Namen `admin` auf. Die Sequenznummer, `0x0`, gibt an, dass dies das erste Ereignis in diesem Protokollstream ist.

Wenn sich ein Benutzer bei einem HSM anmeldet, zeichnet der andere Benutzer HSMs im Cluster ebenfalls ein Anmeldeereignis für den Benutzer auf. Sie finden die entsprechenden Anmeldeereignisse kurz nach dem ersten Anmeldeereignis HSMs in den Protokolldatenströmen anderer Benutzer im Cluster.

```
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_CRYPT0_OFFICER (2)
```

Das folgende Beispielergebnis zeichnet den `admin`-Verschlüsselungsverantwortlichen auf, der sich abmeldet. Die Sequenznummer, `0x2`, gibt an, dass dies das dritte Ereignis im Protokollstream ist.

Wenn der angemeldete Benutzer die Sitzung schließt, ohne sich abzumelden, enthält der Protokollstream ein `CN_APP_FINALIZE`-Ereignis oder ein Ereignis für das Schließen der Sitzung (`CN_SESSION_CLOSE`), anstelle eines `CN_LOGOUT`-Ereignisses. Im Gegensatz zum Anmeldeereignis wird dieses Abmeldeereignis typischerweise nur von jenem HSM aufgezeichnet, das den Befehl ausführt.

```
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_CRYPT0_OFFICER (2)
```

Wenn ein Anmeldeversuch fehlschlägt, weil der Benutzername ungültig ist, zeichnet das HSM ein CN_LOGIN-Ereignis mit dem Benutzernamen und -typ im Login-Befehl auf. Die Antwort zeigt die Fehlermeldung 157, die erläutert, dass der Benutzername nicht vorhanden ist.

```
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_CRYPT0_USER (1)
```

Wenn ein Anmeldeversuch fehlschlägt, weil das Passwort ungültig ist, zeichnet das HSM ein CN_LOGIN-Ereignis mit dem Benutzernamen und -typ im Login-Befehl auf. Die Antwort zeigt die Fehlermeldung mit dem RET_USER_LOGIN_FAILURE-Fehlercode.

```
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_CRYPT0_USER (1)
```

Beispiel: Erstellen und Löschen von Benutzern

Dieses Beispiel zeigt die Protokollereignisse, die aufgezeichnet werden, wenn ein Verschlüsselungsverantwortlicher (CO) Benutzer anlegt und löscht.

Das erste Ereignis zeichnet einen CO, `admin`, bei der Anmeldung im HSM auf. Die Sequenznummer, `0x0`, gibt an, dass dies das erste Ereignis im Protokollstream ist. Der Name und Typ des Benutzers, der sich anmeldet, sind im Ereignis enthalten.

```
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_CRYPT0_OFFICER (2)
```

Das nächste Ereignis im Protokollstream (Sequenz `0x1`) zeichnet die Erstellung eines neuen Verschlüsselungsbenedutzers (CU, Crypto-User) durch den CO auf. Der Name und Typ des neuen Benutzers sind im Ereignis enthalten.

```
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_CRYPT0_USER (1)
```

Dann erstellt der CO einen anderen Verschlüsselungsverantwortlichen, `alice`. Die Sequenznummer zeigt an, dass diese Aktion der vorherigen ohne dazwischenliegende Aktionen folgte.

```
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_CRYPT0_OFFICER (2)
```

Später meldet sich der CO mit Namen `admin` an und löscht den Verschlüsselungsverantwortlichen mit Namen `alice`. Das HSM zeichnet ein `CN_DELETE_USER`-Ereignis auf. Der Name und Typ des gelöschten Benutzers sind im Ereignis enthalten.

```
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_CRYPT0_OFFICER (2)
```

Beispiel: Erstellen und Löschen eines Schlüsselpaares

Dieses Beispiel zeigt die Ereignisse, die in einem HSM-Auditprotokoll aufgezeichnet werden, wenn Sie ein Schlüsselpaar erstellen und löschen.

Das folgende Ereignis zeichnet die Anmeldung des Verschlüsselungsbenedutzers (CU, Crypto User) mit Namen `crypto_user` am HSM auf.

```
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_CRYPT0_USER (1)
```

Anschließend erzeugt der CU ein Schlüsselpaar (`CN_GENERATE_KEY_PAIR`). Das Schlüssel-Handle des privaten Schlüssels ist `131079`. Das Schlüssel-Handle des öffentlichen Schlüssels ist `131078`.

```
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
```

Der CU löscht das Schlüsselpaar sofort wieder. Ein CN_DESTROY_OBJECT-Ereignis zeichnet die Löschung des öffentlichen Schlüssels (131078) auf.

```
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
```

Anschließend zeichnet ein zweites CN_DESTROY_OBJECT-Ereignis die Löschung des privaten Schlüssels (131079) auf.

```
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
```

Und schließlich meldet sich der CU ab.

```
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_CRYPT0_USER (1)

```

Beispiel: Generieren und Synchronisieren eines Schlüssels

Dieses Beispiel zeigt den Effekt der Erstellung eines Schlüssels in einem Cluster mit mehreren HSMs. Der Schlüssel wird auf einem HSM generiert, als maskiertes Objekt aus dem HSM extrahiert und in das andere HSMs als maskiertes Objekt eingefügt.

Note

Die Client-Tools können den Schlüssel möglicherweise nicht synchronisieren. Oder der Befehl könnte den `min_srv` Parameter enthalten, der den Schlüssel nur mit der angegebenen Anzahl von synchronisiert. HSMs In beiden Fällen synchronisiert der AWS CloudHSM Dienst den Schlüssel mit dem anderen Schlüssel HSMs im Cluster. Da sie nur clientseitige Verwaltungsbefehle in ihren Protokollen HSMs aufzeichnen, wird die serverseitige Synchronisation nicht im HSM-Protokoll aufgezeichnet.

Betrachten Sie zunächst den Protokollstream des HSM, das die Befehle empfängt und ausführt. Der Protokollstream ist nach der HSM-ID, `hsm-abcde123456`, benannt, aber die HSM-ID erscheint nicht in den Protokollereignissen.

Zunächst meldet sich der `testuser`-Verschlüsselungsbenutzer (CU, Crypto User) am `hsm-abcde123456`-HSM an.

```

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_CRYPT0_USER (1)

```

Die CU führt einen [exSymKey](#) Befehl aus, um einen symmetrischen Schlüssel zu generieren. Das hsm-abcde123456-HSM generiert einen symmetrischen Schlüssel mit dem Schlüssel-Handle 262152. Das HSM vermerkt ein CN_GENERATE_KEY-Ereignis im Protokoll.

```
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
```

Das nächste Ereignis im Protokollstream für hsm-abcde123456 zeichnet den ersten Schritt im Schlüssel-Synchronisationsprozess auf. Der neue Schlüssel (Schlüssel-Handle 262152) wird aus dem HSM als maskiertes Objekt extrahiert.

```
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
```

Betrachten Sie jetzt den Protokollstream für das HSM hsm-zyxwv987654, ein anderes HSM im selben Cluster. Dieser Protokollstream enthält auch ein Anmelde-Ereignis für den CU testuser. Der Zeitwert zeigt an, dass es kurz nach der Anmeldung des Benutzers am HSM hsm-abcde123456 auftritt.

```
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_CRYPT0_USER (1)
```

Dieser Protokollstream für diese HSM verfügt nicht über ein CN_GENERATE_KEY-Ereignis. Aber es hat ein Ereignis, das die Synchronisation des Schlüssels zu diesem HSM aufzeichnet. Das CN_INSERT_MASKED_OBJECT_USER-Ereignis zeichnet den Empfang des Schlüssels 262152 als maskiertes Objekt auf. Jetzt ist der Schlüssel für beide HSMs im Cluster 262152 vorhanden.

```
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
```

Wenn der CU-Benutzer sich abmeldet, wird dieses CN_LOGOUT-Ereignis nur in den Protokollstream jenes HSM aufgenommen, das die Befehle erhalten hat.

Beispiel: Exportieren eines Schlüssels

Dieses Beispiel zeigt die Auditprotokollereignisse, die aufgezeichnet werden, wenn ein Krypto-Benutzer (CU) Schlüssel aus einem Cluster mit mehreren exportiert HSMs.

Das folgende Ereignis zeichnet die CU (testuser)-Anmeldung bei [key_mgmt_util](#) auf.

```
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_CRYPT0_USER (1)
```

Die CU führt einen [exSymKey](#) Befehl zum Exportieren eines Schlüssels aus 7, eines 256-Bit-AES-Schlüssels. Der Befehl verwendet den Schlüssel 6, einen 256-Bit-AES-Schlüssel auf dem HSMs, als Umschließungsschlüssel.

Das HSM, das den Befehl empfängt, vermerkt ein CN_WRAP_KEY-Ereignis für den Schlüssel 7, den Schlüssel, der exportiert wird.

```
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
```

Anschließend verzeichnet das HSM ein CN_NIST_AES_WRAP-Ereignis für den Verpackungsschlüssel 6. Der Schlüssel wird verpackt und anschließend sofort wieder entpackt, aber der HSM zeichnet nur ein Ereignis auf.

```
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
```

Der exSymKey-Befehl schreibt den exportierten Schlüssel in eine Datei, ändert aber nicht den Schlüssel auf dem HSM. Folglich gibt es keine entsprechenden Ereignisse in den Protokollen anderer Mitglieder des HSMs Clusters.

Beispiel: Importieren eines Schlüssels

Dieses Beispiel zeigt die Auditprotokollereignisse, die aufgezeichnet werden, wenn Sie Schlüssel HSMs in einen Cluster importieren. In diesem Beispiel verwendet der Crypto-Benutzer (CU) den

[imSymKey](#) Befehl, um einen AES-Schlüssel in den zu importieren HSMs. Der Befehl verwendet Schlüssel 6 als Verpackungsschlüssel.

Das HSM, das den Befehl empfängt, vermerkt ein CN_NIST_AES_WRAP-Ereignis für den Schlüssel 6, den Verpackungsschlüssel.

```
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
```

Anschließend verzeichnet das HSM ein CN_UNWRAP_KEY-Ereignis für den Importvorgang. Der importierte Schlüssel ist dem Schlüssel-Handle 11 zugeordnet.

```
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
```

Wenn ein neuer Schlüssel generiert oder importiert wird, versuchen die Client-Tools automatisch, den neuen Schlüssel mit HSMs einem anderen Schlüssel im Cluster zu synchronisieren. In diesem Fall zeichnet das HSM ein CN_EXTRACT_MASKED_OBJECT_USER-Ereignis auf, wenn der Schlüssel 11 aus dem HSM als maskiertes Objekt extrahiert wird.

```
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
```

Die Protokolldatenströme anderer Benutzer HSMs im Cluster spiegeln den Eingang des neu importierten Schlüssels wider.

Dieses Ereignis wurde beispielsweise im Protokollstream eines anderen HSM im gleichen Cluster aufgezeichnet. Dieses CN_INSERT_MASKED_OBJECT_USER-Ereignis zeichnet die Ankunft eines maskierten Objekts auf, das den Schlüssel 11 darstellt.

```
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
```

Beispiel: Freigeben eines Schlüssels und Aufheben der Freigabe

Dieses Beispiel zeigt das Prüfprotokollereignis, das aufgezeichnet wird, wenn ein CU den privaten ECC-Schlüssel für andere CUs freigibt oder die Freigabe aufhebt. Der CU verwendet den Befehl [shareKey](#) und stellt das Schlüssel-Handle, die Benutzer-ID und den Wert 1 zur Verfügung, um den Schlüssel freizugeben oder den Wert 0, um die Freigabe aufzuheben.

Im folgenden Beispiel zeichnet der HSM, der den Befehl empfängt, ein CM_SHARE_OBJECT-Ereignis auf. Dieses Ereignis repräsentiert den Vorgang der Freigabe.

```
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 Referenz zum Auditprotokoll

AWS CloudHSM zeichnet HSM-Verwaltungsbefehle in Auditprotokollereignissen auf. Jedes Ereignis hat einen Operationscode-Wert (Opcode) der die aufgetretene Aktion und ihre Reaktion identifiziert. Sie können die Opcode-Werte zum Durchsuchen, Sortieren und Filtern der Protokolle verwenden.

In der folgenden Tabelle werden die Opcode Werte in einem AWS CloudHSM Auditprotokoll definiert.

Operationscode (Opcode)	Description
Benutzeranmeldung: Diese Ereignisse enthalten den Benutzernamen und Typ.	
CN_LOGIN (0xd)	Anmeldung des Benutzers
CN_LOGOUT (0xe)	Abmeldung des Benutzers
CN_APP_FINALIZE	Die Verbindung mit dem HSM wurde geschlossen. Alle Sitzungsschlüssel oder Quorum-Token aus dieser Verbindung wurden gelöscht.
CN_CLOSE_SESSION	Die Sitzung mit dem HSM wurde geschlossen. Alle Sitzungsschlüssel oder Quorum-Token aus dieser Sitzung wurden gelöscht.
Benutzerverwaltung: Diese Ereignisse enthalten den Benutzernamen und Typ.	
CN_CREATE_USER (0x3)	Erstellen eines Verwaltungsbenutzers (CU, Crypto User).
CN_CREATE_CO	Erstellen eines Verschlüsselungsverantwortlichen (CO)
CN_DELETE_USER	Löschen eines Benutzers
CN_CHANGE_PSWD	Ändern eines Benutzer-Passworts
CN_SET_M_VALUE	Legen Sie die Quorumauthentifizierung (M von N) für eine Benutzeraktion fest
CN_APPROVE_TOKEN	Genehmigen Sie ein Quorum-Authentifizierungstoken für eine Benutzeraktion

Operationscode (Opcode)	Description
CN_DELETE_TOKEN	Löschen Sie ein oder mehrere Quorum-Token
CN_GET_TOKEN	Fordern Sie ein Signaturtoken an, um einen Quorumvorgang einzuleiten
Schlüsselverwaltung: Diese Ereignisse enthalten das Schlüssel-Handle.	
CN_GENERATE_KEY	Generieren eines symmetrischen Schlüssels
CN_GENERATE_KEY_PAIR (0x19)	Generieren Sie ein asymmetrisches key pair
CN_CREATE_OBJECT	Importieren eines öffentlichen Schlüssels (ohne Verschlüsselungsschlüssel)
CN_MODIFY_OBJECT	Legen Sie ein identifizierendes Attribut fest
CN_DESTROY_OBJECT (0x11)	Löschen eines Sitzungsschlüssels
CN_TOMBSTONE_OBJECT	Löschen eines Token-Schlüssels
CN_SHARE_OBJECT	Freigabe oder Aufheben der Freigabe eines Schlüssels
CN_WRAP_KEY	Exportieren einer verschlüsselten Kopie eines Schlüssels (wrapKey)
CN_UNWRAP_KEY	Importieren einer verschlüsselten Kopie eines Schlüssels (unwrapKey)
CN_DERIVE_KEY	Leiten Sie einen symmetrischen Schlüssel aus einem vorhandenen Schlüssel ab
CN_NIST_AES_WRAP	Verschlüsseln oder entschlüsseln Sie einen Schlüssel mit einem AES-Schlüssel
CN_INSERT_MASKED_OBJECT_USER	Fügen Sie einen verschlüsselten Schlüssel mit Attributen aus einem anderen HSM in den Cluster ein.

Operationscode (Opcode)	Description
CN_EXTRACT_MASKED_OBJECT_USER	Umschließt/verschlüsselt einen Schlüssel mit Attributen aus dem HSM, der an ein anderes HSM im Cluster gesendet werden soll.
Session Management	
CN_ENCRYPT_SESSION_V2 (0x107)	Stellt eine authentifizierte verschlüsselte Sitzung her. end-to-end
END_MARKER_OPCODE (0xffff)	Fügt eine Endmarkierung in den Audit-Log-Puffer ein, die angibt, dass auf dem HSM keine protokollierbaren Befehle mehr zulässig sind
Back up HSMs	
CN_BACKUP_BEGIN	Beginnt mit dem Backup-Vorgang
CN_BACKUP_END	Der Backup-Vorgang wurde abgeschlossen
CN_RESTORE_BEGIN	Beginnen Sie mit der Wiederherstellung aus einem Backup
CN_RESTORE_END	Der Wiederherstellungsvorgang aus einem Backup wurde abgeschlossen
Certificate-Based Authentication	
CN_CERT_AUTH_STORE_CERT	Speichert das Cluster-Zertifikat
HSM Instance Commands	
CN_INIT_TOKEN (0x1)	Starten Sie den HSM-Initialisierungsprozess
CN_INIT_DONE	Der HSM-Initialisierungsprozess ist abgeschlossen
CN_GEN_KEY_ENC_KEY	Generieren eines Schlüsselverschlüsselungsschlüssels (Key Encryption Key, KEK)

Operationscode (Opcode)	Description
CN_GEN_PSWD_ENC_KEY (0x1d)	Generieren eines Passwort-Verschlüsselungsschlüssels (Key Encryption Key, PEK)
HSM crypto commands	
CN_FIPS_RAND	Generieren Sie eine FIPS-konforme Zufallszahl 1

[1] Wird nur für hsm1.medium-Cluster protokolliert.

CloudWatch Metriken abrufen für AWS CloudHSM

Wird verwendet CloudWatch , um Ihren AWS CloudHSM Cluster in Echtzeit zu überwachen. Die Metriken können nach Region, Cluster-ID oder Cluster-ID und HSM-ID gruppiert werden.

Der AWS/CloudHSM-Namespace enthält die folgenden Metriken:

Metrik	Description
HsmUnhealthy	Die HSM-Instanz funktioniert nicht ordnungsgemäß. AWS CloudHSM ersetzt automatisch fehlerhafte Instanzen für Sie. Sie können die Cluster-Größe proaktiv erweitern, um die Auswirkungen auf die Leistung während des Austauschs des HSM zu reduzieren.
HsmTemperature 1	Die Junction-Temperatur des Hardware-Prozessors. Das System wird heruntergefahren, wenn die Temperatur 110 °C erreicht.
HsmKeysSessionOccupied	Die Anzahl der Sitzungsschlüssel, die von der HSM-Instance verwendet werden.
HsmKeysTokenOccupied	Die Anzahl der Token-Schlüssel, die von der HSM-Instance und dem Cluster verwendet werden.
HsmSslContextsOccupied 1	Die Anzahl der end-to-end verschlüsselten Kanäle, die derzeit für die HSM-Instanz eingerichtet wurden. Es sind bis zu 2.048 Kanäle zulässig.

Metrik	Description
HsmSessionCount	Die Anzahl der offenen Verbindungen zur HSM-Instance. Es sind bis zu 2.048 zulässig. Standardmäßig ist der Client-Daemon so konfiguriert, dass er zwei Sitzungen mit jeder HSM-Instanz unter einem end-to-end verschlüsselten Kanal öffnet. AWS CloudHSM kann auch bis zu 2 Verbindungen mit dem HSM geöffnet haben, um den Zustand des zu überwachen. HSMs
HsmUsersAvailable	Die Anzahl der zusätzlichen Benutzer, die erstellt werden können. Dies entspricht der maximalen Anzahl von Benutzern (aufgeführt unter HsmUsersMax) abzüglich der bisher erstellten Benutzer.
HsmUsersMax ¹	Maximale Anzahl von Benutzern, die in der HSM-Instance erstellt werden können.
InterfaceEth2OctetsInput ¹	Die kumulierte Summe des bisher auf das HSM eingehenden Datenverkehrs.
InterfaceEth2OctetsOutput ¹	Die kumulierte Summe des bisher auf das HSM ausgehenden Datenverkehrs.

- [1] Diese Metrik ist für hsm2m.medium nicht verfügbar.

AWS CloudHSM Informationen zur Leistung

Für AWS CloudHSM Produktionscluster sollten Sie über mindestens zwei HSM-Instances (Hardware Security Module) verfügen, die über verschiedene Availability Zones in einer Region verteilt sind. Wir empfehlen, Ihren Cluster einem Belastungstest zu unterziehen, um die zu erwartende Spitzenlast zu ermitteln, und dann ein weiteres HSM hinzuzufügen, um eine hohe Verfügbarkeit zu gewährleisten. Für Anwendungen, die auf zuverlässige, neu generierte Schlüssel angewiesen sind, empfehlen wir mindestens drei HSM-Instances, die über verschiedene Availability Zones in einer Region verteilt sind.

Leistungsdaten

Die Leistung von AWS CloudHSM Clustern hängt von der jeweiligen Arbeitslast ab. Um die Leistung zu erhöhen, können Sie Ihren Clustern zusätzliche HSM-Instances hinzufügen. Die Leistung kann je nach Konfiguration, Datengröße und zusätzlicher Anwendungslast auf Ihren EC2-Instances variieren. Wir empfehlen Ihnen, Ihre Anwendung einem Belastungstest zu unterziehen, um den Skalierungsbedarf zu ermitteln.

Die folgende Tabelle zeigt die ungefähre Leistung für gängige kryptografische Algorithmen, die auf einer EC2-Instance mit hsm1.medium-Instances ausgeführt werden.

Leistungsdaten für hsm1.medium

Operation	Cluster mit zwei HSMs ¹	Cluster mit drei HSMs ²	Cluster mit sechs HSMs ³
RSA 2048-bit sign	2.000 Operationen/ Sekunde	3.000 Operationen/ Sekunde	5.000 Operationen/ Sekunde
EC P256 sign	500 Operationen/ Sekunde	750 Operationen/ Sekunde	1.500 Operationen/ Sekunde

Die folgende Tabelle zeigt die ungefähre Leistung für gängige kryptografische Algorithmen, die auf einer EC2-Instance mit hsm2m.medium ausgeführt werden.

Leistungsdaten für hsm2m.medium

Operation	Cluster mit zwei HSMs ¹	Cluster mit drei HSMs ²	Cluster mit sechs HSMs ³
RSA 2048-bit sign	2000 Ops/Sek	3000 Bilder/Sek	5000 Ops/Sek
EC P256 sign	3000 Ops/Sek	4500 Ops/Sek	7000 Ops/Sek

- [1] Ein Cluster mit zwei HSMs, bei dem die Java-Multithread-Anwendung auf einer [c4.large-EC2-Instance mit einem HSM in derselben AZ wie die EC2-Instance](#) ausgeführt wird.
- [2] Ein Cluster mit drei HSMs, bei dem die Java-Multithread-Anwendung auf einer [c4.large-EC2-Instance mit einem HSM in derselben AZ wie die EC2-Instance](#) ausgeführt wird.
- [3] Ein Cluster mit sechs HSMs, bei dem die Java-Multithread-Anwendung auf einer [c4.large-EC2-Instance mit zwei HSMs in derselben AZ wie die EC2-Instance](#) ausgeführt wird.

HSM-Drosselung

Wenn Ihre Arbeitslast die HSM-Kapazität Ihres Clusters überschreitet, erhalten Sie Fehlermeldungen, die besagen, dass sie ausgelastet oder HSMs gedrosselt sind. Einzelheiten dazu, was in einem solchen Fall zu tun ist, finden Sie unter [HSM-Drosselung](#).

Sicherheit in AWS CloudHSM

Cloud-Sicherheit AWS hat höchste Priorität. Als AWS Kunde profitieren Sie von Rechenzentren und Netzwerkarchitekturen, die darauf ausgelegt sind, die Anforderungen der sicherheitssensibelsten Unternehmen zu erfüllen.

Sicherheit ist eine gemeinsame AWS Verantwortung von Ihnen und Ihnen. Das [Modell der geteilten Verantwortung](#) beschreibt dies als Sicherheit der Cloud und Sicherheit in der Cloud:

- Sicherheit der Cloud — AWS ist verantwortlich für den Schutz der Infrastruktur, auf der AWS Dienste in der ausgeführt AWS Cloud werden. AWS bietet Ihnen auch Dienste, die Sie sicher nutzen können. Externe Prüfer testen und verifizieren regelmäßig die Wirksamkeit unserer Sicherheitsmaßnahmen im Rahmen der [AWS](#) . Weitere Informationen zu den Compliance-Programmen, die für gelten AWS CloudHSM, finden Sie unter [AWS Services im Umfang nach Compliance-Programmen AWS](#) .
- Sicherheit in der Cloud — Ihre Verantwortung richtet sich nach dem AWS Dienst, den Sie nutzen. Sie sind auch für andere Faktoren verantwortlich, etwa für die Vertraulichkeit Ihrer Daten, für die Anforderungen Ihres Unternehmens und für die geltenden Gesetze und Vorschriften.

Diese Dokumentation hilft Ihnen zu verstehen, wie Sie das Modell der gemeinsamen Verantwortung bei der Nutzung anwenden können AWS CloudHSM. In den folgenden Themen erfahren Sie, wie Sie die Konfiguration vornehmen AWS CloudHSM , um Ihre Sicherheits- und Compliance-Ziele zu erreichen. Sie lernen auch, wie Sie andere AWS-Services nutzen können, die Sie bei der Überwachung und Sicherung Ihrer AWS CloudHSM Ressourcen unterstützen.

Inhalt

- [Steuern Sie den API-Zugriff mit IAM-Richtlinien](#)
- [Datenschutz in AWS CloudHSM](#)
- [Identitäts- und Zugriffsmanagement für AWS CloudHSM](#)
- [Konformitätsprüfung für AWS CloudHSM](#)
- [Resilienz in AWS CloudHSM](#)
- [Sicherheit der Infrastruktur in AWS CloudHSM](#)
- [AWS CloudHSM und VPC-Endpunkte](#)
- [Aktualisieren Sie die Verwaltung in AWS CloudHSM](#)

Steuern Sie den API-Zugriff mit IAM-Richtlinien

Aktualisieren Sie die IAM-Richtlinien auf IPv6

AWS CloudHSM Kunden verwenden IAM-Richtlinien, um den Zugriff auf IP-Adressen außerhalb des konfigurierten Bereichs zu kontrollieren AWS CloudHSM APIs und zu verhindern, dass sie darauf zugreifen können. AWS CloudHSM APIs

Die Cloud hsmv2. *<region>*.api.aws-Dual-Stack-Endpunkt, auf dem AWS CloudHSM APIs gehostet wird, unterstützt zusätzlich IPv6 IPv4

Kunden, die beide unterstützen müssen IPv4 und ihre Richtlinien zur IP-Adressfilterung aktualisieren IPv6 müssen, um IPv6 Adressen zu verarbeiten. Andernfalls wird sich dies auf ihre Fähigkeit auswirken, eine Verbindung zu Over herzustellen. AWS CloudHSM IPv6

Wer sollte ein Upgrade durchführen?

Kunden, die duale Adressierung mit Richtlinien verwenden, die aws:sourcelP enthalten, sind von diesem Upgrade betroffen. Duale Adressierung bedeutet, dass das Netzwerk sowohl als auch unterstützt. IPv4 IPv6

Wenn Sie die duale Adressierung verwenden, müssen Sie Ihre IAM-Richtlinien, die derzeit mit IPv4 Formatadressen konfiguriert sind, so aktualisieren, dass sie auch IPv6 Formatadressen enthalten.

Wenn Sie Hilfe bei Zugriffsproblemen benötigen, wenden Sie sich an [Support](#).

Note

Die folgenden Kunden sind von diesem Upgrade nicht betroffen:

- Kunden, die nur in IPv4 Netzwerken aktiv sind.

Was ist IPv6?

IPv6 ist der IP-Standard der nächsten Generation, der irgendwann ersetzt IPv4 werden soll. Die vorherige Version verwendet ein 32-Bit-Adressierungsschema zur Unterstützung von 4,3 Milliarden Geräten. IPv4 IPv6 verwendet stattdessen 128-Bit-Adressierung, um etwa 340 Billionen Billionen (oder 2 bis 128.) Geräte zu unterstützen.

Weitere Informationen finden Sie auf der [IPv6 VPC-Webseite](#).

```
2001:cdba:0000:0000:0000:0000:3257:9652
2001:cdba:0:0:0:0:3257:9652
2001:cdba::3257:965
```

Aktualisierung einer IAM-Richtlinie für IPv6

IAM-Richtlinien werden derzeit verwendet, um mithilfe des Filters einen zulässigen Bereich von IP-Adressen festzulegen. `aws:SourceIp`

Die duale Adressierung unterstützt sowohl den Datenverkehr als IPv4 auch IPv6 den Datenverkehr. Wenn Ihr Netzwerk die duale Adressierung verwendet, müssen Sie alle IAM-Richtlinien, die für die IP-Adressfilterung verwendet werden, so aktualisieren, dass sie IPv6 Adressbereiche enthalten.

In der folgenden Richtlinie werden beispielsweise die zulässigen IPv4 Adressbereiche `192.0.2.0.*` und `203.0.113.0.*` im Condition Element angegeben.

```
# 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"
      }
    }
  }
}
```

Um diese Richtlinie zu aktualisieren, ändern Sie das Condition Element so, dass es die IPv6 Adressbereiche `2001:DB8:1234:5678::/64` und `2001:cdba:3257:8593::/64`.

Note

ENTFERNEN SIE DIE vorhandenen IPv4 Adressen NICHT, da sie aus Gründen der Abwärtskompatibilität benötigt werden.

```
"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"
    }
}
```

Stellen Sie sicher, dass Ihr Kunde Folgendes unterstützt IPv6

Kunden, die Cloudhsmv2 verwenden. Es wird empfohlen, den Endpunkt {region}.api.aws zu überprüfen, ob sie eine Verbindung zu diesem Endpunkt herstellen können. In den folgenden Schritten wird beschrieben, wie die Überprüfung durchgeführt wird.

Dieses Beispiel verwendet Linux und Curl Version 8.6.0 und verwendet die [AWS CloudHSM Service-Endpunkte, für die Endpunkte](#) IPv6 aktiviert sind, die sich am api.aws-Endpunkt befinden.

Note

Wechseln Sie AWS-Region zu derselben Region, in der sich der Client befindet. In diesem Beispiel verwenden wir den Endpunkt us-east-1 in USA Ost (Nord-Virginia).

1. Stellen Sie mit dem folgenden dig Befehl fest, ob der Endpunkt mit einer IPv6 Adresse aufgelöst wird.

```
dig +short AAAA cloudhsmv2.us-east-1.api.aws
```

```
2600:1f18:e2f:4e05:1a8a:948e:7c08:c1c3
```

2. Stellen Sie mit dem folgenden `curl` Befehl fest, ob das Client-Netzwerk eine IPv6 Verbindung herstellen kann. Ein 404-Antwortcode bedeutet, dass die Verbindung erfolgreich war, während ein 0-Antwortcode bedeutet, dass die Verbindung fehlgeschlagen ist.

```
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
```

Wenn eine Remote-IP identifiziert wurde und der Antwortcode nicht angegeben ist⁰, wurde mithilfe von erfolgreich eine Netzwerkverbindung zum Endpunkt hergestellt IPv6. Die Remote-IP sollte eine IPv6 Adresse sein, da das Betriebssystem das für den Client gültige Protokoll auswählen sollte. Wenn es sich bei der Remote-IP nicht um eine IPv6 Adresse handelt, verwenden Sie den folgenden Befehl, um die Verwendung `curl` zu erzwingen IPv4.

```
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
```

Wenn die Remote-IP leer ist oder der Antwortcode leer ist⁰, ist das Client-Netzwerk oder der Netzwerkpfad zum Endpunkt IPv4 -only. Sie können diese Konfiguration mit dem folgenden `curl` Befehl überprüfen.

```
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
```

Datenschutz in AWS CloudHSM

Das [Modell der AWS gemeinsamen Verantwortung](#) und geteilter Verantwortung gilt für den Datenschutz in AWS CloudHSM. Wie in diesem Modell beschrieben, AWS ist verantwortlich für

den Schutz der globalen Infrastruktur, auf der alle Systeme laufen AWS Cloud. Sie sind dafür verantwortlich, die Kontrolle über Ihre in dieser Infrastruktur gehosteten Inhalte zu behalten. Sie sind auch für die Sicherheitskonfiguration und die Verwaltungsaufgaben für die von Ihnen verwendeten AWS-Services verantwortlich. Weitere Informationen zum Datenschutz finden Sie unter [Häufig gestellte Fragen zum Datenschutz](#). Informationen zum Datenschutz in Europa finden Sie im Blog-Beitrag [AWS -Modell der geteilten Verantwortung und in der DSGVO](#) im AWS -Sicherheitsblog.

Aus Datenschutzgründen empfehlen wir, dass Sie AWS-Konto Anmeldeinformationen schützen und einzelne Benutzer mit AWS IAM Identity Center oder AWS Identity and Access Management (IAM) einrichten. So erhält jeder Benutzer nur die Berechtigungen, die zum Durchführen seiner Aufgaben erforderlich sind. Außerdem empfehlen wir, die Daten mit folgenden Methoden schützen:

- Verwenden Sie für jedes Konto die Multi-Faktor-Authentifizierung (MFA).
- Wird verwendet SSL/TLS , um mit AWS Ressourcen zu kommunizieren. Wir benötigen TLS 1.2 und empfehlen TLS 1.3.
- Richten Sie die API und die Protokollierung von Benutzeraktivitäten mit ein AWS CloudTrail. Informationen zur Verwendung von CloudTrail Pfaden zur Erfassung von AWS Aktivitäten finden Sie unter [Arbeiten mit CloudTrail Pfaden](#) im AWS CloudTrail Benutzerhandbuch.
- Verwenden Sie AWS Verschlüsselungslösungen zusammen mit allen darin enthaltenen Standardsicherheitskontrollen AWS-Services.
- Verwenden Sie erweiterte verwaltete Sicherheitsservices wie Amazon Macie, die dabei helfen, in Amazon S3 gespeicherte persönliche Daten zu erkennen und zu schützen.
- Wenn Sie für den Zugriff AWS über eine Befehlszeilenschnittstelle oder eine API FIPS 140-3-validierte kryptografische Module benötigen, verwenden Sie einen FIPS-Endpunkt. Weitere Informationen über verfügbare FIPS-Endpunkte finden Sie unter [Federal Information Processing Standard \(FIPS\) 140-3](#).

Wir empfehlen dringend, in Freitextfeldern, z. B. im Feld Name, keine vertraulichen oder sensiblen Informationen wie die E-Mail-Adressen Ihrer Kunden einzugeben. Dies gilt auch, wenn Sie mit der Konsole, der AWS CloudHSM API oder auf andere AWS-Services Weise arbeiten oder diese verwenden. AWS CLI AWS SDKs Alle Daten, die Sie in Tags oder Freitextfelder eingeben, die für Namen verwendet werden, können für Abrechnungs- oder Diagnoseprotokolle verwendet werden. Wenn Sie eine URL für einen externen Server bereitstellen, empfehlen wir dringend, keine Anmeldeinformationen zur Validierung Ihrer Anforderung an den betreffenden Server in die URL einzuschließen.

Verschlüsselung im Ruhezustand

Wenn ein AWS CloudHSM Backup von einem HSM erstellt wird, verschlüsselt das HSM seine Daten, bevor es sie an sendet. AWS CloudHSM Die Daten werden mit einem spezifischen flüchtigen Verschlüsselungsschlüssel verschlüsselt. Weitere Informationen finden Sie unter [AWS CloudHSM Cluster-Backups](#).

Verschlüsselung während der Übertragung

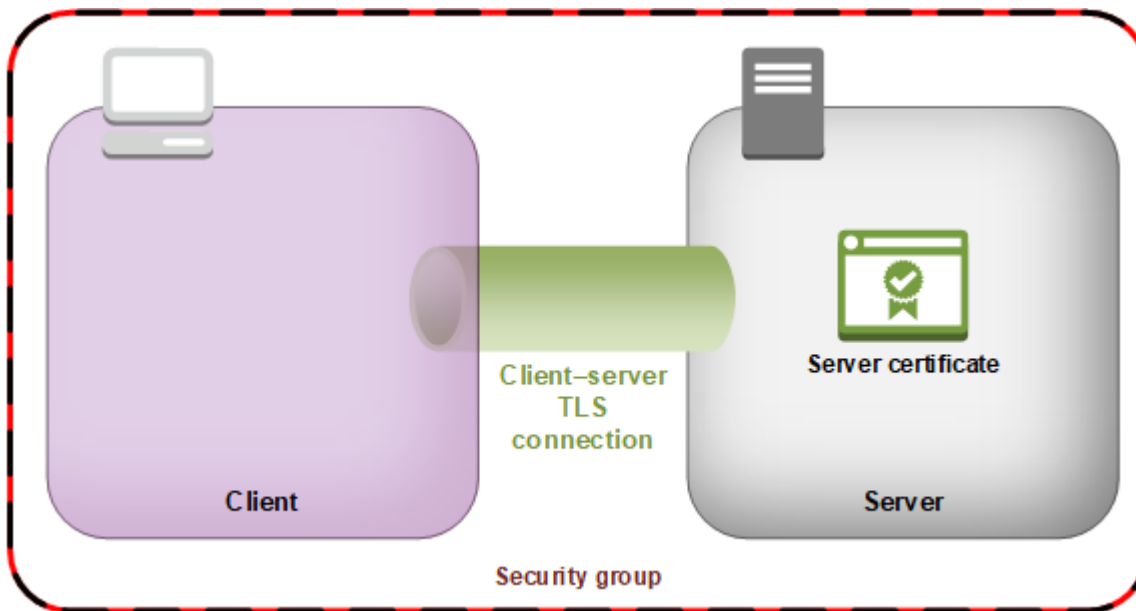
Die Kommunikation zwischen dem AWS CloudHSM Client und dem HSM in Ihrem Cluster ist von Ende zu Ende verschlüsselt. Diese Kommunikation kann nur vom Client und vom HSM entschlüsselt werden. Weitere Informationen finden Sie unter [End-to-end Verschlüsselung](#).

AWS CloudHSM Client-Verschlüsselung end-to-end

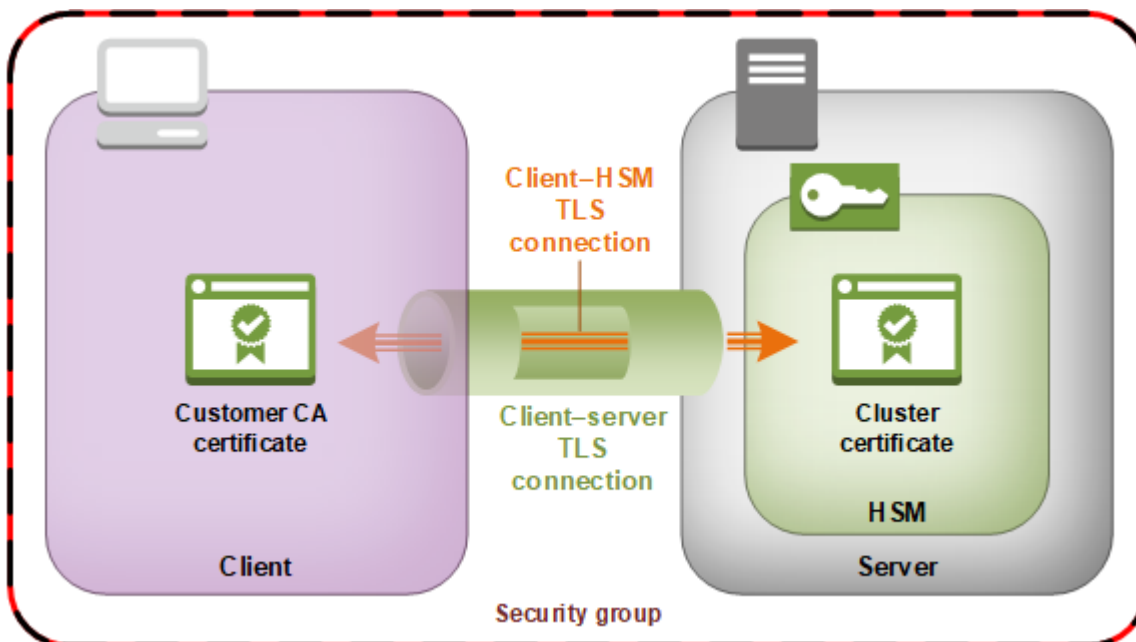
Die Kommunikation zwischen der Client-Instance und der HSMs in Ihrem Cluster ist von Ende zu Ende verschlüsselt. Nur Ihr Client und Sie HSMs können die Kommunikation entschlüsseln.

Der folgende Prozess erklärt, wie der Client eine end-to-end verschlüsselte Kommunikation mit einem HSM herstellt.

1. Ihr Client stellt eine gegenseitig authentifizierte Transport Layer Security (TLS)-Verbindung zu dem Server her, der Ihre HSM-Hardware hostet. Die Sicherheitsgruppe Ihres Clusters erlaubt eingehenden Datenverkehr an den Server nur von Client-Instances in der Sicherheitsgruppe. Der Client überprüft auch das Zertifikat des Servers, um sicherzustellen, dass es sich um einen vertrauenswürdigen Server handelt.



2. Als nächstes richtet der Client eine verschlüsselte Verbindung mit der HSM-Hardware ein. Das HSM hat das Cluster-Zertifikat, das Sie mit Ihrer eigenen Zertifikatsstelle (CA) signiert haben, und der Client hat das Stammzertifikat der CA. Bevor die verschlüsselte Client-HSM-Verbindung eingerichtet wird, überprüft der Client das Cluster-Zertifikat des HSM anhand seines Stammzertifikats. Die Verbindung wird nur hergestellt, wenn der Client nur erfolgreich sichergestellt hat, dass der HSM vertrauenswürdig ist.



Sicherheit von Cluster-Sicherungen

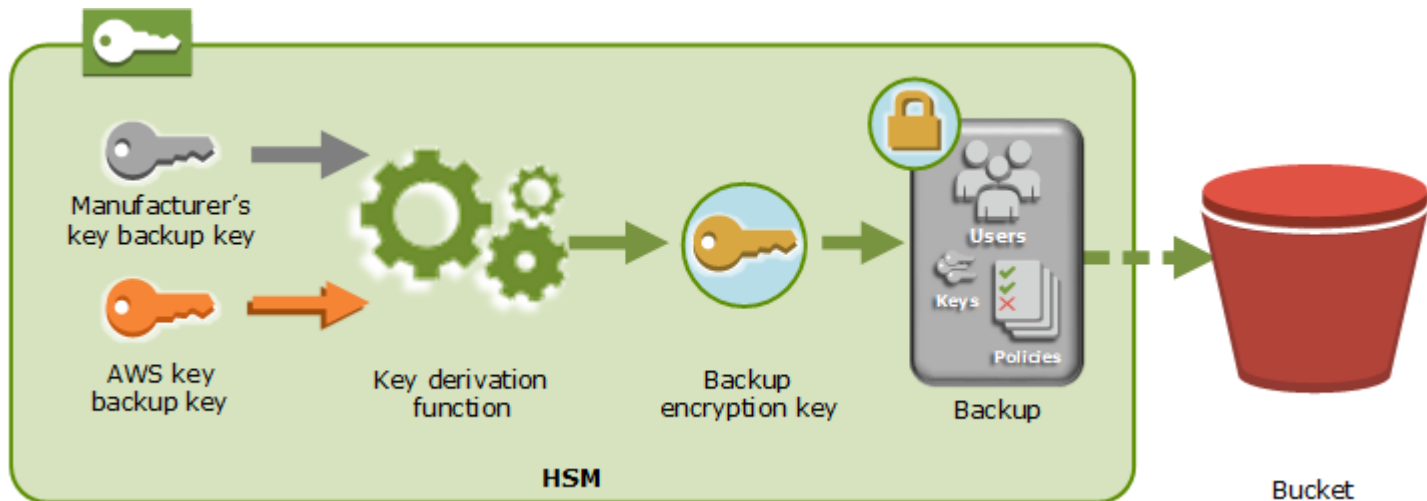
Wenn AWS CloudHSM ein Backup vom HSM erstellt wird, verschlüsselt das HSM alle Daten, bevor es sie an sendet. AWS CloudHSM Die Daten werden zu keinem Zeitpunkt unverschlüsselt aus dem HSM ausgegeben. Darüber hinaus können Backups nicht entschlüsselt werden, AWS da AWS es keinen Zugriff auf den Schlüssel hat, der zum Entschlüsseln der Backups verwendet wurde.

Um seine Daten zu verschlüsseln, verwendet das HSM einen eindeutigen, flüchtigen Verschlüsselungsschlüssel als vorübergehenden Schlüssel für die Sicherung (EBK, Ephemeral Backup Key). Das EBK ist ein AES-256-Bit-Verschlüsselungsschlüssel, der bei der Erstellung eines Backups im HSM generiert wird. AWS CloudHSM Das HSM generiert den EBK und verwendet ihn anschließend, um die Daten des HSM unter Verwendung einer FIPS-konformen Wrapping-Methode für AES-Schlüssel entsprechend der [NIST Special Publication 800-38F](#) zu verschlüsseln. Dann gibt das HSM die verschlüsselten Daten an weiter. AWS CloudHSM Die verschlüsselten Daten enthalten eine verschlüsselte Kopie des EBK.

Um den EBK zu verschlüsseln, verwendet HSM einen anderen Verschlüsselungsschlüssel, der als persistenter Sicherungsschlüssel (PBK, Persistent Backup Key) bezeichnet wird. Der PBK (Persistent Backup Key) ist ebenfalls ein AES 256-Bit-Verschlüsselungsschlüssel. Um den PBK zu erzeugen, verwendet das HSM eine FIPS-konforme Schlüsselableitungsfunktion (KDF, Key Derivation Function) in einem Counter-Modus entsprechend der [NIST Special Publication 800-108](#). Diese KDF verwendet u. a. die folgenden Eingaben:

- Ein Herstellerschlüssel zur Schlüsselsicherung (MKBK, Manufacturer Key Backup Key), der vom Hardwarehersteller fest in die HSM-Hardware integriert ist
- Ein AWS Schlüssel-Backup-Schlüssel (AKBK), der bei der ersten Konfiguration von sicher im HSM installiert wurde. AWS CloudHSM

Die Verschlüsselungsverfahren sind in der folgenden Abbildung zusammengefasst. Der Sicherungs-Verschlüsselungsschlüssel stellt den PBK und den EBK dar.



AWS CloudHSM kann Backups nur auf Geräten wiederherstellen AWS, die HSMs vom selben Hersteller stammen. Da jede Sicherung alle Benutzer, Schlüssel und Konfigurationen des ursprünglichen HSM enthält, enthält das wiederhergestellte HSM enthält dieselben Schutzmechanismen und Zugriffssteuerungen wie das ursprüngliche. Bei der Wiederherstellung werden alle anderen Daten, die sich möglicherweise vor der Wiederherstellung noch auf dem HSM befinden, vollständig überschrieben.

Eine Sicherung besteht aus nur verschlüsselten Daten. Bevor der Service ein Backup in Amazon S3 speichert, verschlüsselt der Service das Backup erneut mit AWS Key Management Service (AWS KMS).

Identitäts- und Zugriffsmanagement für AWS CloudHSM

AWS verwendet Sicherheitsanmeldeinformationen, um Sie zu identifizieren und Ihnen Zugriff auf Ihre AWS-Ressourcen zu gewähren. Sie können Funktionen von AWS Identity and Access Management (IAM) verwenden, um anderen Benutzern, Services und Anwendungen die vollständige oder eingeschränkte Nutzung Ihrer AWS-Ressourcen zu ermöglichen. Sie können dies tun, ohne Ihre Sicherheitsanmeldeinformationen zu teilen.

IAM-Benutzer sind standardmäßig nicht berechtigt, AWS-Ressourcen zu erstellen, anzuzeigen oder zu ändern. Um einem IAM-Benutzer den Zugriff auf Ressourcen wie einen Load Balancer und das Ausführen von Aufgaben zu ermöglichen, gehen Sie folgendermaßen vor:

1. Erstellen Sie eine IAM-Richtlinie, die dem IAM-Benutzer die Berechtigung zur Nutzung der jeweiligen Ressourcen und API-Aktionen erteilt, die er benötigt.
2. Weisen Sie die Richtlinie dem IAM-Benutzer oder der Gruppe zu, zu der der IAM-Benutzer gehört.

Wenn Sie einem Benutzer oder einer Benutzergruppe eine Richtlinie zuordnen, wird den Benutzern die Ausführung der angegebenen Aufgaben für die angegebenen Ressourcen gestattet oder verweigert.

Sie können IAM beispielsweise verwenden, um Benutzer und Gruppen unter Ihrem AWS-Konto zu erstellen. Ein IAM-Benutzer kann eine Person, ein System oder eine Anwendung sein. Anschließend gewähren Sie den Benutzern und Gruppen Berechtigungen, um bestimmte Aktionen für die angegebenen Ressourcen mithilfe einer IAM-Richtlinie durchzuführen.

Erteilen von Berechtigungen mithilfe von IAM-Richtlinien

Wenn Sie einem Benutzer oder einer Benutzergruppe eine Richtlinie zuordnen, wird den Benutzern die Ausführung der angegebenen Aufgaben für die angegebenen Ressourcen gestattet oder verweigert.

Eine IAM-Richtlinie ist ein JSON-Dokument, das eine oder mehrere Anweisungen enthält.

Das folgende Beispiel für eine IAM-Richtlinie ermöglicht es Benutzern, AWS CloudHSM Backups im Osten der USA (Nord-Virginia) zu beschreiben. Es sind nur Anfragen zur Beschreibung von Anfragen zulässig, die aus dem Osten der USA (Nord-Virginia) stammen.

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:** Der effect-Wert kann Allow oder Deny lauten. IAM-Benutzer verfügen standardmäßig nicht über die Berechtigung zur Verwendung von Ressourcen und API-Aktionen. Daher werden alle Anfragen abgelehnt. Dieser Standardwert kann durch eine explizite Zugriffserlaubnis überschrieben werden. Eine explizite Zugriffsverweigerung überschreibt jedwede Zugriffserlaubnis.
- **Aktion:** Mit Aktion wird die API-Aktion spezifiziert, für die Sie Berechtigungen erteilen oder verweigern. Für weitere Informationen zum Angeben von Aktionen siehe [API-Aktionen für AWS CloudHSM](#).
- **Ressource** — Die Ressource, die von der Aktion betroffen ist. AWS CloudHSM unterstützt keine Berechtigungen auf Ressourcenebene. Sie müssen den Platzhalter „*“ verwenden, um alle Ressourcen anzugeben. AWS CloudHSM
- **Condition:** Sie können optional Bedingungen verwenden, um zu steuern, wann die Richtlinie wirksam ist. Weitere Informationen finden Sie unter [Bedingungsschlüssel für AWS CloudHSM](#).

Weitere Informationen finden Sie im [IAM-Benutzerhandbuch](#).

API-Aktionen für AWS CloudHSM

Im Action-Element Ihrer IAM-Richtlinienerklärung können Sie jede API-Aktion angeben, die AWS CloudHSM angeboten wird. Dem Namen der Aktion muss wie im folgenden Beispiel die Zeichenfolge `cloudhsm:` in Kleinbuchstaben vorangestellt werden.

```
"Action": "cloudhsm:DescribeClusters"
```

Wenn Sie mehrere Aktionen in einer einzigen Anweisung angeben möchten, setzen Sie sie in eckige Klammern und trennen Sie sie wie im folgenden Beispiel dargestellt durch Kommas.

```
"Action": [  
    "cloudhsm:DescribeClusters",  
    "cloudhsm:DescribeHsm"  
]
```

Sie können auch mehrere Aktionen mithilfe des Platzhalters `*` angeben. Im folgenden Beispiel werden alle API-Aktionsnamen angegeben AWS CloudHSM, die mit `List` beginnen.

```
"Action": "cloudhsm:List*"
```

Um alle API-Aktionen für anzugeben AWS CloudHSM, verwenden Sie den Platzhalter *, wie im folgenden Beispiel gezeigt.

```
"Action": "cloudhsm:*"
```

Eine Liste der API-Aktionen für finden Sie AWS CloudHSM unter [AWS CloudHSM Aktionen](#).

Bedingungsschlüssel für AWS CloudHSM

Beim Erstellen einer Richtlinie können Sie die Bedingungen angeben, die steuern, wann die Richtlinie wirksam wird. Jede Bedingung enthält ein oder mehrere Schlüssel-Wert-Paare. Es gibt globale Bedingungsschlüssel und servicespezifische Bedingungsschlüssel.

AWS CloudHSM hat keine dienstspezifischen Kontextschlüssel.

Weitere Informationen über globale Bedingungsschlüssel finden Sie unter [Globale Bedingungskontextschlüssel in AWS](#) im IAM-Benutzer-Leitfaden.

Vordefinierte AWS-verwaltete Richtlinien für AWS CloudHSM

Die von AWS erstellten verwalteten Richtlinien erteilen die erforderlichen Berechtigungen für häufige Anwendungsfälle. Sie können diese Richtlinien basierend auf dem erforderlichen Zugriff auf AWS CloudHSM an IAM-Benutzer anfügen:

- **AWSCloudHSMFullZugriff** — Gewährt vollen Zugriff, der für die Nutzung der AWS CloudHSM Funktionen erforderlich ist.
- **AWSCloudHSMReadOnlyAccess**— Gewährt schreibgeschützten Zugriff auf AWS CloudHSM Funktionen.

Vom Kunden verwaltete Richtlinien für AWS CloudHSM

Wir empfehlen Ihnen, dafür eine IAM-Administratorgruppe zu erstellen AWS CloudHSM , die nur die für die Ausführung AWS CloudHSM erforderlichen Berechtigungen enthält. Fügen Sie die Richtlinie mit den benötigten Berechtigungen an diese Gruppe an. Fügen Sie der Gruppe bei Bedarf IAM-Benutzer hinzu. Jeder hinzugefügte Benutzer erbt die Richtlinie von der Administratorgruppe.

Außerdem empfehlen wir, zusätzliche Benutzergruppen basierend auf den Berechtigungen zu erstellen, die die jeweiligen Benutzer benötigen. Dadurch wird sichergestellt, dass nur

vertrauenswürdige Benutzer Zugriff auf kritische API-Aktionen erhalten. Sie könnten beispielsweise eine Benutzergruppe erstellen, mit der Sie Clustern und nur Lesezugriff gewähren. HSMs Da diese Gruppe es einem Benutzer nicht erlaubt, Cluster zu löschen HSMs, oder ein nicht vertrauenswürdiger Benutzer kann die Verfügbarkeit eines Produktions-Workloads nicht beeinflussen.

Da im Laufe der Zeit neue AWS CloudHSM Verwaltungsfunktionen hinzugefügt werden, können Sie sicherstellen, dass nur vertrauenswürdige Benutzer sofort Zugriff erhalten. Indem Sie den Richtlinien bei der Erstellung begrenzte Berechtigungen zuweisen, können Sie diesen zu einem späteren Zeitpunkt neue Funktionsberechtigungen manuell zuweisen.

Im Folgenden finden Sie Beispielrichtlinien für AWS CloudHSM. Informationen zum Erstellen einer Richtlinie und zum Anfügen an eine IAM-Benutzergruppe finden Sie unter [Erstellen von Richtlinien auf der Registerkarte „JSON“](#) im IAM-Benutzer-Leitfaden.

Beispiele

- [Berechtigungen mit Schreibschutz](#)
- [Berechtigungen für Hauptbenutzer](#)
- [Berechtigungen für Administratoren](#)

Example Beispiel: Berechtigungen mit Schreibschutz

Diese Richtlinie gewährt Zugriff auf die API-Aktionen `DescribeClusters` und `DescribeBackups`. Sie enthält außerdem zusätzliche Berechtigungen für bestimmte Amazon-EC2-API-Aktionen. Es erlaubt dem Benutzer nicht, Cluster zu löschen oder HSMs.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Action": [
      "cloudhsm:DescribeClusters",
      "cloudhsm:DescribeBackups",
      "cloudhsm:ListTags"
    ],
    "Resource": "*"
  }
}
```

```
}
```

Example Beispiel: Berechtigungen für Hauptbenutzer

Diese Richtlinie ermöglicht den Zugriff auf eine Teilmenge der AWS CloudHSM API-Aktionen. Sie enthält außerdem zusätzliche Berechtigungen für bestimmte Amazon-EC2-Aktionen. Es erlaubt dem Benutzer nicht, Cluster zu löschen oder HSMs. Sie müssen die `iam:CreateServiceLinkedRole` Aktion angeben, mit der die automatische Erstellung der mit dem `AWSServiceRoleForCloudHSM`-Dienst verknüpften Rolle in Ihrem Konto ermöglicht AWS CloudHSM wird. Diese Rolle ermöglicht das AWS CloudHSM Protokollieren von Ereignissen. Weitere Informationen finden Sie unter [Mit Diensten verknüpfte Rollen für AWS CloudHSM](#).

Note

Eine Liste der Aktionen in jedem Service finden Sie unter [Aktionen, Ressourcen und Konditionsschlüssel für AWS CloudHSM](#) in der Referenz zur Serviceberechtigung.

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 Beispiel: Berechtigungen für Administratoren

Diese Richtlinie ermöglicht den Zugriff auf alle AWS CloudHSM API-Aktionen, einschließlich der Aktionen zum Löschen von HSMs und Clustern. Sie enthält außerdem zusätzliche Berechtigungen für bestimmte Amazon-EC2-Aktionen. Sie müssen die `iam:CreateServiceLinkedRole` Aktion angeben, mit der die automatische Erstellung der mit dem `AWSServiceRoleForCloudHSM`-Service verknüpften Rolle in Ihrem Konto ermöglicht AWS CloudHSM wird. Diese Rolle ermöglicht das AWS CloudHSM Protokollieren von Ereignissen. Weitere Informationen finden Sie unter [Mit Diensten verknüpfte Rollen für AWS CloudHSM](#).

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": "*"
}
}

```

Mit Diensten verknüpfte Rollen für AWS CloudHSM

Die IAM-Richtlinie, die Sie zuvor erstellt haben, [Vom Kunden verwaltete Richtlinien für AWS CloudHSM](#) beinhaltet die `iam:CreateServiceLinkedRole` Aktion. AWS CloudHSM definiert eine [serviceverknüpfte Rolle mit dem Namen HSM](#) `AWSServiceRoleForCloud`. Die Rolle ist vordefiniert von AWS CloudHSM und umfasst auch Berechtigungen, die AWS CloudHSM erforderlich sind, um andere AWS Dienste in Ihrem Namen aufzurufen. Die Rolle vereinfacht die Einrichtung Ihres Service, da Sie die Rollenrichtlinie und Berechtigungen der Vertrauensrichtlinie nicht manuell hinzufügen müssen.

Die Rollenrichtlinie ermöglicht es AWS CloudHSM, Amazon CloudWatch Logs-Protokollgruppen und Log-Streams zu erstellen und Protokollereignisse in Ihrem Namen zu schreiben. Sie können sie nachstehend und in der IAM-Konsole einsehen.

JSON

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "logs:CreateLogGroup",
        "logs:CreateLogStream",
        "logs:PutLogEvents",
        "logs:DescribeLogStreams"
      ]
    }
  ]
}

```

```

    ],
    "Resource": [
        "arn:aws:logs:*:*:*"
    ]
}
]
}

```

Die Vertrauensrichtlinie für die AWSServiceRoleForCloudHSM-Rolle ermöglicht es AWS CloudHSM, die Rolle zu übernehmen.

JSON

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "cloudhsm.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}

```

Erstellen einer serviceverknüpften Rolle (automatisch)

AWS CloudHSM erstellt die AWSServiceRoleForCloudHSM-Rolle, wenn Sie einen Cluster erstellen, wenn Sie die `iam:CreateServiceLinkedRole` Aktion in die Berechtigungen aufnehmen, die Sie bei der Erstellung der AWS CloudHSM Administratorgruppe definiert haben. Siehe [Vom Kunden verwaltete Richtlinien für AWS CloudHSM](#).

Wenn Sie bereits über einen oder mehrere Cluster verfügen und nur die AWSServiceRoleForCloudHSM-Rolle hinzufügen möchten, können Sie die Konsole, den Befehl [create-cluster](#) oder die [CreateCluster](#) API-Operation verwenden, um einen Cluster zu erstellen. Verwenden Sie dann die Konsole, den Befehl [delete-cluster](#) oder den API-Vorgang, um ihn zu löschen. [DeleteCluster](#) Beim Erstellen des neuen Clusters wird die serviceverknüpfte Rolle erstellt

und auf alle Cluster in Ihrem Konto angewandt. Alternativ können Sie die Rolle manuell erstellen. Weitere Informationen finden Sie im folgenden Abschnitt .

Note

Sie müssen nicht alle unter Erstellen eines Clusters beschriebenen Schritte ausführen [Erste Schritte mit AWS CloudHSM](#), wenn Sie ihn nur erstellen, um die AWSService RoleForCloud HSM-Rolle hinzuzufügen.

Erstellen einer serviceverknüpften Rolle (manuell)

Sie können die IAM-Konsole oder API verwenden AWS CLI, um die AWSServiceRoleForCloudHSM-Rolle zu erstellen. Weitere Informationen finden Sie unter [Erstellen einer serviceverknüpfte Rolle](#) im IAM-Leitfaden.

Bearbeiten der serviceverknüpften Rolle

AWS CloudHSM erlaubt Ihnen nicht, die AWSServiceRoleForCloudHSM-Rolle zu bearbeiten. Nachdem die Rolle erstellt wurde, können Sie z. B. nicht ihren Namen ändern, da möglicherweise verschiedene Entitäten unter dem Namen auf die Rolle verweisen. Die Rollenrichtlinie kann ebenfalls nicht geändert werden. Sie können mithilfe von IAM jedoch die Beschreibung der Rolle bearbeiten. Weitere Informationen finden Sie unter [Bearbeiten einer serviceverknüpften Rolle](#) im IAM-Benutzerhandbuch.

Löschen der serviceverknüpften -Rolle

Eine serviceverknüpfte Rolle kann nicht gelöscht werden, solange noch Cluster vorhanden sind, auf die sie angewendet wurde. Um die Rolle zu löschen, müssen Sie zuerst die einzelnen HSM im Cluster und dann den Cluster selbst löschen. Jeder Cluster in Ihrem Konto muss gelöscht werden. Sie können dann die IAM-Konsole oder API verwenden AWS CLI, um die Rolle zu löschen. Weitere Informationen zum Löschen eines Clusters finden Sie unter [Löschen eines AWS CloudHSM Clusters](#). Weitere Informationen finden Sie unter [Löschen einer serviceverknüpften Rolle](#) im IAM-Leitfaden.

Konformitätsprüfung für AWS CloudHSM

Für Cluster im FIPS-Modus werden FIPS-zertifizierte Produkte AWS CloudHSM bereitgestellt, die die PCI-PIN-, PCI-3DS- und HSMs Compliance-Anforderungen erfüllen. SOC2 AWS CloudHSM bietet

Kunden außerdem die Möglichkeit, Cluster auszuwählen, die sich nicht im FIPS-Modus befinden. Einzelheiten dazu, welche Zertifizierungs- und Compliance-Anforderungen jeweils gelten, finden Sie unter [AWS CloudHSM Cluster-Modi](#)

Wenn Sie sich auf ein FIPS-validiertes HSM verlassen, können Sie die unternehmensinternen, vertraglichen und behördlichen Compliance-Anforderungen für die Datensicherheit in der Cloud erfüllen. AWS

Compliance mit FIPS 140-2

Die Veröffentlichung 140-2 des Federal Information Processing Standard (FIPS) ist ein Sicherheitsstandard der US-Regierung, mit dem die Sicherheitsanforderungen für Verschlüsselungsmodule angegeben werden, die vertrauliche Informationen schützen. [Der Instance-Typ AWS CloudHSM hsm1.medium ist nach FIPS 140-2 Level 3 zertifiziert \(Zertifikat #4218\)](#). Am 4. Januar 2026 wird das Zertifikat für hsm1.medium in die historische Liste aufgenommen. [Wir empfehlen Kunden, zu hsm2m.medium zu migrieren, das nach FIPS 140-3 zertifiziert ist \(Zertifikat #4703\)](#). [Weitere Informationen finden Sie unter FIPS-Validierung für Hardware](#).

FIPS 140-3-Konformität

Die Veröffentlichung 140-3 des Federal Information Processing Standard (FIPS) ist ein Sicherheitsstandard der US-Regierung, der Sicherheitsanforderungen für kryptografische Module zum Schutz vertraulicher Informationen festlegt. [Der von HSMs bereitgestellte Typ hsm2m.medium ist nach FIPS 140-3 Level 3 AWS CloudHSM zertifiziert \(Zertifikat #4703\)](#). [Weitere Informationen finden Sie unter FIPS-Validierung für Hardware](#).

[PCI-DSS-Compliance](#)

Der Payment Card Industry Data Security Standard (PCI-DSS) ist ein vom [Payment Card Industry Security Standards Council](#) festgelegter proprietärer Standard für Informationssicherheit. Die von HSMs bereitgestellten AWS CloudHSM entsprechen PCI DSS.

[PCI-PIN-Compliance](#)

PCI PIN bietet Sicherheitsanforderungen und Bewertungsstandards für die Übertragung, Verarbeitung und Verwaltung von PIN-Daten (Personal Identification Number), d. h. Informationen, die für Transaktionen an ATMs und point-of-sale (POS-) Terminals verwendet werden. Die von bereitgestellten Dateien hsm1.medium und hsm2m.medium sind beide HSMs PCI-PIN-konform. AWS CloudHSM Weitere Informationen finden Sie im Artikel [AWS CloudHSM ist jetzt PCI-PIN-zertifiziert](#).

Konformität mit PCI-3DS

PCI 3DS (oder Three Domain Secure, 3-D Secure) bietet Datensicherheit für sichere EMV-3D-E-Commerce-Zahlungen. PCI 3DS bietet eine weitere Sicherheitsebene für Online-Einkäufe. Die von bereitgestellten Versionen hsm1.medium und hsm2m.medium HSMs sind beide PCI-3DS-kompatibel. AWS CloudHSM

SOC2

SOC2 ist ein Framework, das Serviceorganisationen dabei unterstützt, ihre Sicherheitskontrollen für Cloud und Rechenzentren nachzuweisen. AWS CloudHSM hat in kritischen Bereichen SOC2 Kontrollen implementiert, um die Trusted Service Principles einzuhalten. Weitere Informationen finden Sie auf [der AWS FAQs SOC-Seite](#).

AWS CloudHSM PCI-PIN-Konformität FAQs

Die PCI-PIN enthält Sicherheitsanforderungen und Bewertungsstandards für die Übertragung, Verarbeitung und Verwaltung von PIN-Daten (Personal Identification Number), d. h. Informationen, die für Transaktionen an ATMs und point-of-sale (POS-) Terminals verwendet werden.

Die PCI-PIN-Konformitätsbescheinigung (AOC) und die Zusammenfassung der Verantwortlichkeiten stehen Kunden über AWS Artifact zur Verfügung, ein Self-Service-Portal für den On-Demand-Zugriff auf AWS-Compliance-Berichte. Für weitere Informationen melden Sie sich in der [AWS-Managementkonsole bei AWS Artifact](#) an. Sie erfahren auch mehr unter [Erste Schritte mit AWS Artifact](#).

FAQs

F: Was ist die Konformitätsbescheinigung und die Zusammenfassung der Verantwortlichkeiten?

Die Konformitätsbescheinigung (AOC) wird von einem Qualified PIN Assessor (QPA) ausgestellt, der bescheinigt, dass die geltenden Kontrollen des AWS CloudHSM PCI-PIN-Standards eingehalten werden. In der Übersicht über die Zuständigkeiten werden die Kontrollen beschrieben, für die die jeweiligen Kunden verantwortlich sind. AWS CloudHSM

F: Wie erhalte ich die AWS CloudHSM Konformitätsbescheinigung?

Die PCI-PIN-Konformitätsbescheinigung (AOC) steht Kunden über AWS Artifact zur Verfügung, ein Self-Service-Portal für den On-Demand-Zugriff auf AWS-Compliance-Berichte. Für weitere

Informationen melden Sie sich in der [AWS-Managementkonsole bei AWS Artifact](#) an. Sie erfahren auch mehr unter [Erste Schritte mit AWS Artifact](#).

F: Wie kann ich erfahren, für welche PCI-PIN-Kontrollen ich verantwortlich bin?

Detaillierte Informationen finden Sie unter „Zusammenfassung der AWS CloudHSM PCI-PIN-Verantwortung“ aus dem AWS PCI PIN Compliance Package, das Kunden über AWS Artifact, ein Self-Service-Portal für On-Demand-Zugriff auf AWS-Compliance-Berichte, zur Verfügung steht. Für weitere Informationen melden Sie sich in der [AWS-Managementkonsole bei AWS Artifact](#) an. Sie erfahren auch mehr unter [Erste Schritte mit AWS Artifact](#).

F: Kann ich mich als AWS CloudHSM Kunde auf die PCI-PIN Attestation of Compliance (AOC) verlassen?

Kunden müssen ihre PCI-PIN-Konformität selbst verwalten. Sie müssen ein formelles PCI-PIN-Bescheinigungsverfahren durch einen qualifizierten PIN-Assessor (QPA) durchlaufen, um zu überprüfen, ob Ihre Zahlungs-Workload alle PCI-PIN-Kontrollen/-Anforderungen erfüllt. Für die Kontrollen, für die AWS verantwortlich ist, kann sich Ihre QPA jedoch ohne weitere Tests auf die Konformitätsbescheinigung (AWS CloudHSM Attestation of Compliance, AOC) verlassen.

F: Ist AWS CloudHSM verantwortlich für die PCI-PIN-Anforderungen im Zusammenhang mit dem Lebenszyklus der Schlüsselmanagement?

AWS CloudHSM ist verantwortlich für den Lebenszyklus der physischen Geräte von HSMs. Die Kunden sind für die wichtigsten Anforderungen des PCI-PIN-Standards über den Lebenszyklus der Verwaltung verantwortlich.

F: Welche AWS CloudHSM Steuerungen sind PCI-PIN-konform?

Das AOC fasst die AWS CloudHSM Kontrollen zusammen, die von QPA bewertet werden. Die PCI-PIN-Zusammenfassung der Verantwortlichkeiten steht Kunden über AWS Artifact zur Verfügung, ein Self-Service-Portal für den On-Demand-Zugriff auf AWS-Compliance-Berichte.

F: AWS CloudHSM Unterstützt es Zahlungsfunktionen wie PIN-Übersetzung und DUKPT?

Nein, AWS CloudHSM dient allgemeinen Zwecken HSMs. Im Laufe der Zeit können wir ggf. Zahlungsfunktionen anbieten. Obwohl der Dienst Zahlungsfunktionen nicht direkt ausführt, ermöglicht die AWS CloudHSM PCI-PIN-Konformitätsbescheinigung den Kunden, ihre eigene PCI-Konformität für ihre laufenden Dienste zu erreichen. AWS CloudHSM Wenn Sie daran interessiert sind, die Services von AWS Payment Cryptography für Ihre Workload zu nutzen, lesen Sie bitte den Blog [„Verlagerung der Zahlungsabwicklung in die Cloud mit AWS Payment Cryptography“](#).

Benachrichtigungen über veraltete Funktionen

AWS CloudHSM Kann von Zeit zu Zeit Funktionen verwerfen, um die Anforderungen von FIPS 140, PCI-DSS, PCI-PIN, PCI-3DS zu erfüllen, oder aufgrund von Hardware. SOC2 end-of-support Auf dieser Seite sind die derzeit geltenden Änderungen aufgeführt.

HSM1 Veraltet

Der Support für den Instance-Typ AWS CloudHSM hsm1.medium endet am 31. März 2026. Um einen kontinuierlichen Service zu gewährleisten, führen wir die folgenden Änderungen ein:

- Ab April 2025 können Sie keine neuen hsm1.medium-Cluster mehr erstellen.
- Ab Januar 2026 werden wir damit beginnen, bestehende hsm1.medium-Cluster automatisch auf den neuen Instance-Typ hsm2m.medium zu migrieren.

Der Instance-Typ hsm2m.medium ist mit Ihrem aktuellen Instance-Typ kompatibel und bietet eine verbesserte Leistung. AWS CloudHSM Um Unterbrechungen Ihrer Anwendungen zu vermeiden, müssen Sie auf die neueste Version des Client-SDK aktualisieren. Anweisungen zum Upgrade finden Sie unter [???](#).

Sie haben zwei Optionen für die Migration:

1. Melden Sie sich für eine von CloudHSM verwaltete Migration an, wenn Sie bereit sind. Weitere Informationen finden Sie unter [???](#).
2. Erstellen Sie einen neuen hsm2m.medium-Cluster aus einem Backup Ihres hsm1-Clusters und leiten Sie Ihre Anwendung auf den neuen Cluster um. Wir empfehlen, für diesen Ansatz eine blue/green Bereitstellungsstrategie zu verwenden. Weitere Informationen finden Sie unter [???](#).

FIPS-140-Konformität: Mechanismus 2024 nicht mehr unterstützt

Das National Institute of Standards and Technology (NIST) ¹weist darauf hin, dass die Unterstützung von Triple-DES-Verschlüsselung (DESede, 3DES DES3) und RSA-Schlüsselumbruch und -entpackung mit PKCS #1 v1.5-Padding nach dem 31. Dezember 2023 nicht mehr zulässig ist. Daher endet die Unterstützung für diese Programme am 1. Januar 2024 in unseren Clustern im FIPS-Modus (Federal Information Processing Standard). Diese werden weiterhin für Cluster im FIPs Nichtmodus Support.

Diese Anleitung gilt für die folgenden kryptografischen Operationen:

- Generierung von Triple-DES-Schlüsseln
 - CKM_DES3_KEY_GEN für die PKCS-#11-Bibliothek
 - DESede-Keygen für den JCE-Anbieter
 - genSymKey mit -t=21 für die KMU
- Verschlüsselung mit Triple-DES-Schlüsseln (Hinweis: Entschlüsselungsvorgänge sind zulässig)
 - Für die PKCS-#11-Bibliothek: CKM_DES3_CBC verschlüsseln, CKM_DES3_CBC_PAD verschlüsseln und CKM_DES3_ECB verschlüsseln
 - Für den JCE-Anbieter: DESede/CBC/PKCS5Padding verschlüsseln, DESede/CBC/NoPadding verschlüsseln, DESede/ECB/Padding verschlüsseln und DESede/ECB/NoPadding verschlüsseln
- Verpacken, Entpacken, Verschlüsseln und Entschlüsseln von RSA-Schlüsseln mit PKCS #1 v1.5-Padding
 - CKM_RSA_PKCSVerpacken, Entpacken, Verschlüsseln und Entschlüsseln für das PKCS-#11-SDK
 - RSA/ECB/PKCS1Padding für das JCE SDK einpacken, entpacken, verschlüsseln und entschlüsseln
 - wrapKey und unwrapKey mit -m 12 für die KMU (Hinweis: 12 ist der Wert für den Mechanismus RSA_PKCS)

[1] Einzelheiten zu dieser Änderung finden Sie in Tabelle 1 und Tabelle 5 unter [Umstellung auf den Einsatz kryptografischer Algorithmen und Schlüssellängen](#).

Resilienz in AWS CloudHSM

Die AWS globale Infrastruktur basiert auf AWS Regionen und Availability Zones. AWS Regionen bieten mehrere physisch getrennte und isolierte Availability Zones, die über Netzwerke mit niedriger Latenz, hohem Durchsatz und hoher Redundanz miteinander verbunden sind. Mithilfe von Availability Zones können Sie Anwendungen und Datenbanken erstellen und ausführen, die automatisch Failover zwischen Zonen ausführen, ohne dass es zu Unterbrechungen kommt. Availability Zones sind besser verfügbar, fehlertoleranter und skalierbarer als herkömmliche Infrastrukturen mit einem oder mehreren Rechenzentren.

Weitere Informationen zu AWS Regionen und Availability Zones finden Sie unter [AWS Globale Infrastruktur](#). Für weitere Informationen zu AWS CloudHSM -Funktionen, die der Unterstützung der Resilienz dienen, siehe [AWS CloudHSM Hochverfügbarkeit und Lastenausgleich im Cluster](#).

Sicherheit der Infrastruktur in AWS CloudHSM

Als verwalteter Service AWS CloudHSM ist er durch die AWS globalen Netzwerksicherheitsverfahren geschützt, die im Whitepaper [Amazon Web Services: Sicherheitsprozesse im Überblick](#) beschrieben sind.

Sie verwenden AWS veröffentlichte API-Aufrufe für den Zugriff AWS CloudHSM über das Netzwerk. Außerdem müssen Anforderungen mit einer Zugriffsschlüssel-ID und einem geheimen Zugriffsschlüssel signiert sein, der einem IAM-Prinzipal zugeordnet ist. Alternativ können Sie mit [AWS -Security-Token-Service](#) (AWS STS) temporäre Sicherheitsanmeldeinformationen erstellen, um die Anforderungen zu signieren.

Netzwerkisolierung

Eine Virtual Private Cloud (VPC) ist ein virtuelles Netzwerk in Ihrem eigenen logisch isolierten Bereich in der AWS Cloud. Sie können einen Cluster in einem privaten Subnetz der VPC erstellen. Sie können private Subnetze erstellen, wenn Sie eine VPC erstellen. Weitere Informationen finden Sie unter [Erstellen Sie eine virtuelle private Cloud \(VPC\) für AWS CloudHSM](#).

Wenn Sie ein HSM erstellen, AWS CloudHSM fügen Sie ein elastic network interface (ENI) in Ihr Subnetz ein, damit Sie mit Ihrem interagieren können. HSMs Weitere Informationen finden Sie unter [AWS CloudHSM Cluster-Architektur](#).

AWS CloudHSM erstellt eine Sicherheitsgruppe, die eingehende und ausgehende Kommunikation zwischen HSMs in Ihrem Cluster ermöglicht. Sie können diese Sicherheitsgruppe verwenden, um Ihren EC2-Instances die Kommunikation mit den HSMs in Ihrem Cluster zu ermöglichen. Weitere Informationen finden Sie unter [Konfigurieren Sie die Sicherheitsgruppen der Amazon EC2 EC2-Client-Instanz für AWS CloudHSM](#).

Autorisierung von Benutzern

Mit erfordern Operationen AWS CloudHSM, die auf dem HSM ausgeführt werden, die Anmeldeinformationen eines authentifizierten HSM-Benutzers. Weitere Informationen finden Sie unter [the section called “Benutzertypen”](#).

AWS CloudHSM und VPC-Endpunkte

Sie können eine private Verbindung zwischen Ihrer VPC herstellen und AWS CloudHSM einen VPC-Schnittstellen-Endpunkt erstellen. Schnittstellenendpunkte werden mit einer Technologie betrieben

[AWS PrivateLink](#), mit der Sie privat AWS CloudHSM APIs ohne Internet-Gateway, NAT-Gerät, VPN-Verbindung oder AWS Direct Connect Connect-Verbindung zugreifen können. Instances in Ihrer VPC benötigen keine öffentlichen IP-Adressen für die Kommunikation. AWS CloudHSM APIs Datenverkehr zwischen Ihrer VPC und AWS CloudHSM verlässt das Amazon-Netzwerk nicht.

Jeder Schnittstellenendpunkt wird durch eine oder mehrere [Elastic-Netzwerk-Schnittstellen](#) in Ihren Subnetzen dargestellt.

Weitere Informationen finden Sie unter [Interface VPC Endpoints \(AWS PrivateLink\)](#) im Amazon VPC-Benutzerhandbuch.

Überlegungen zu AWS CloudHSM VPC-Endpunkten

Bevor Sie einen Schnittstellen-VPC-Endpunkt für einrichten AWS CloudHSM, sollten Sie die [Eigenschaften und Einschränkungen der Schnittstellen-Endpunkte](#) im Amazon VPC-Benutzerhandbuch lesen.

- AWS CloudHSM unterstützt Aufrufe aller API-Aktionen von Ihrer VPC aus.

Erstellen eines Schnittstellen-VPC-Endpunkts für AWS CloudHSM

Sie können einen VPC-Endpunkt für den AWS CloudHSM Service entweder mit der Amazon VPC-Konsole oder mit AWS Command Line Interface ()AWS CLI erstellen. Weitere Informationen finden Sie unter [Erstellung eines Schnittstellenendpunkts](#) im Benutzerhandbuch für Amazon VPC.

Verwenden Sie den folgenden Dienstnamen AWS CloudHSM, um einen VPC-Endpunkt für zu erstellen:

```
com.amazonaws.<region>.cloudhsmv2
```

In der Region USA West (Oregon) (us-west-2) würde der Servicename wie folgt lauten:

```
com.amazonaws.us-west-2.cloudhsmv2
```

Um die Verwendung des VPC-Endpunkts zu vereinfachen, können Sie einen [privaten DNS-Hostnamen](#) für Ihren VPC-Endpunkt aktivieren. Wenn Sie die Option Privaten DNS-Namen aktivieren auswählen, werden die AWS CloudHSM Standard-DNS-Hostnamen (`https://cloudhsmv2.<region>.amazonaws.com` und `https://cloudhsmv2.<region>.api.aws`) zu Ihrem VPC-Endpunkt aufgelöst.

Diese Option vereinfacht die Verwendung des VPC-Endpunkts. Das AWS SDKs und AWS CLI verwendet standardmäßig den AWS CloudHSM Standard-DNS-Hostnamen, sodass Sie die VPC-Endpunkt-URL in Anwendungen und Befehlen nicht angeben müssen.

Weitere Informationen finden Sie unter [Zugriff auf einen Service über einen Schnittstellenendpunkt](#) im Benutzerhandbuch für Amazon VPC.

Erstellen einer VPC-Endpunktrichtlinie für AWS CloudHSM

Sie können eine Endpunktrichtlinie an Ihren VPC-Endpunkt anhängen, der den Zugriff auf AWS CloudHSM steuert. Die Richtlinie gibt die folgenden Informationen an:

- Prinzipal, der die Aktionen ausführen kann.
- Aktionen, die ausgeführt werden können
- Die Ressourcen, für die Aktionen ausgeführt werden können.

Weitere Informationen finden Sie unter [Steuerung des Zugriffs auf Services mit VPC-Endpunkten](#) im Amazon-VPC-Benutzerhandbuch.

Beispiel: VPC-Endpunktrichtlinie für Aktionen AWS CloudHSM

Im Folgenden finden Sie ein Beispiel für eine Endpunktrichtlinie für AWS CloudHSM. Wenn diese Richtlinie an einen Endpunkt angehängt ist, gewährt sie allen Prinzipalen auf allen Ressourcen Zugriff auf die aufgelisteten AWS CloudHSM Aktionen. [Identitäts- und Zugriffsmanagement für AWS CloudHSM](#) Weitere AWS CloudHSM Aktionen und die entsprechenden IAM-Berechtigungen finden Sie unter.

```
{
  "Statement": [
    {
      "Principal": "*",
      "Effect": "Allow",
      "Action": [
        "<cloudhsm>:<DescribeBackups>",
        "<cloudhsm>:<DescribeClusters>",
        "<cloudhsm>:<ListTags>",
      ],
      "Resource": "*"
    }
  ]
}
```

```
}
```

Aktualisieren Sie die Verwaltung in AWS CloudHSM

AWS verwaltet die Firmware. Die Firmware wird von einem Drittanbieter verwaltet und muss je nach HSM-Typ vom NIST auf die Einhaltung von FIPS 140-2 Level 3 oder FIPS 140-3 Level 3 geprüft werden. Es kann nur mit dem FIPS-Schlüssel (auf den AWS keinen Zugriff hat) kryptografisch signierte Firmware installiert werden.

Problembehebung AWS CloudHSM

Wenn Sie auf Probleme mit stoßen AWS CloudHSM, können Ihnen die folgenden Themen helfen, diese zu lösen.

Themen

- [AWS CloudHSM bekannte Probleme](#)
- [AWS CloudHSM Fehler bei der Schlüsselsynchronisierung des Client-SDK 3](#)
- [AWS CloudHSM Client SDK 3 verifiziert die HSM-Leistung mit dem pkpspeed-Tool](#)
- [AWS CloudHSM Der Client SDK 5-Benutzer enthält inkonsistente Werte](#)
- [AWS CloudHSM Fehler bei der Replikation von Client-SDK für 5 Benutzer](#)
- [AWS CloudHSM Fehler bei der Schlüsselreplikation des Client-SDK 5](#)
- [AWS CloudHSM Bei der Überprüfung der Schlüsselverfügbarkeit ist ein Fehler aufgetreten](#)
- [AWS CloudHSM Extrahieren von Schlüsseln mit JCE](#)
- [HSM-Drosselung](#)
- [Halten Sie HSM-Benutzer HSMs im gesamten Cluster synchron AWS CloudHSM](#)
- [Verbindung zum AWS CloudHSM Cluster verloren](#)
- [Fehlende AWS CloudHSM Audit-Logs CloudWatch](#)
- [Benutzerdefiniert IVs mit nicht konformer Länge für AES-Schlüsselumbruch AWS CloudHSM](#)
- [Behebung von Fehlern bei der AWS CloudHSM Clustererstellung](#)
- [AWS CloudHSM Client-Konfigurationsprotokolle abrufen](#)

AWS CloudHSM bekannte Probleme

AWS CloudHSM hat die folgenden bekannten Probleme. Wählen Sie ein Thema, um mehr zu erfahren.

Themen

- [Bekannte Probleme für alle HSM-Instances](#)
- [Bekannte Probleme für AWS CloudHSM hsm1.medium-Instances](#)
- [Bekannte Probleme für hsm2m.medium-Instances AWS CloudHSM](#)

- [Bekannte Probleme mit der PKCS #11 -Bibliothek für AWS CloudHSM](#)
- [Bekannte Probleme mit dem JCE SDK für AWS CloudHSM](#)
- [Bekannte Probleme mit der OpenSSL Dynamic Engine für AWS CloudHSM](#)
- [Bekannte Probleme beim Key Storage Provider \(KSP\) für AWS CloudHSM](#)
- [Bekannte Probleme für den OpenSSL Provider für AWS CloudHSM](#)
- [Bekannte Probleme für Amazon EC2 EC2-Instances, auf denen Amazon Linux 2 ausgeführt wird mit AWS CloudHSM](#)
- [Bekannte Probleme bei der Integration von Drittanbieteranwendungen mit AWS CloudHSM](#)
- [Bekannte Probleme bei der AWS CloudHSM Cluster-Modifikation](#)
- [Bekannte Probleme aufgrund von Betriebsfehlern bei Verwendung der AWS CloudHSM Client-Version 5.12.0 auf hsm2.medium](#)

Bekannte Probleme für alle HSM-Instances

Die folgenden Probleme betreffen alle AWS CloudHSM Benutzer, unabhängig davon, ob sie das Befehlszeilentool `key_mgmt_util`, das PKCS #11 -SDK, das JCE SDK oder das OpenSSL SDK verwenden.

Themen

- [Problem: Beim AES Key Wrapping wird anstelle der standardkonformen Implementierung der Schlüsselpackung ohne Padding das PKCS #5 Padding verwendet.](#)
- [Problem: Der Client-Daemon benötigt mindestens eine gültige IP-Adresse in seiner Konfigurationsdatei, um erfolgreich eine Verbindung mit dem Cluster herstellen zu können.](#)
- [Problem: Es gab eine Obergrenze von 16 KB für Daten, die AWS CloudHSM mit dem Client-SDK 3 gehasht und signiert werden können](#)
- [Problem: Importierte Schlüssel konnten nicht als nicht exportierbar angegeben werden](#)
- [Problem: Der Standardmechanismus für den `WrapKey` und die `unWrapKey` Befehle in `key\_mgmt\_util` wurde entfernt](#)
- [Problem: Befindet sich ein einzelnes HSM in Ihrem Cluster, funktioniert HSM-Failover nicht ordnungsgemäß.](#)
- [Problem: Wenn Sie innerhalb kurzer Zeit die Schlüsselkapazität von HSMs in Ihrem Cluster überschreiten, geht der Client in einen unbehandelten Fehlerstatus über](#)

- [Problem: Digest-Operationen mit HMAC-Schlüsseln mit einer Größe von mehr als 800 Bytes werden nicht unterstützt.](#)
- [Problem: Das im Client-SDK 3 enthaltene Tool client_info löscht den Inhalt des durch das optionale Ausgabeargument angegebenen Pfads](#)
- [Problem: Sie erhalten eine Fehlermeldung, wenn Sie das SDK-5-Configure Tool mit dem --cluster-id-Argument in containerisierten Umgebungen ausführen](#)
- [Problem: Sie erhalten die Fehlermeldung „Fehler beim Erstellen cert/key aus der bereitgestellten PFX-Datei. Fehler: NotPkcs 8“](#)
- [Problem: Die ECDSA-Signatur schlägt ab SDK 5.16 mit dem Fehler „Ungültiger Mechanismus“ fehl](#)
- [Problem: Beim Signieren mit vorgehashten Daten werden Sitzungstoken im interaktiven Modus nicht ordnungsgemäß gelöscht](#)
- [Problem: Das Standard-Client-Zertifikat der CloudHSM-Clientbibliothek läuft am 31. Januar 2026 ab](#)

Problem: Beim AES Key Wrapping wird anstelle der standardkonformen Implementierung der Schlüsselverpackung ohne Padding das PKCS #5 Padding verwendet.

Des Weiteren werden Schlüsselverpackungen ohne Padding und mit Zero Padding nicht unterstützt.

- **Auswirkung:** Das Ein- und Auspacken mit diesem Algorithmus hat keine Auswirkungen. AWS CloudHSM Schlüssel, die mit eingeschlossen sind, können jedoch AWS CloudHSM nicht in einer anderen Software HSMs oder Software entpackt werden, die die Einhaltung der No-Padding-Spezifikation voraussetzt. Dies liegt daran, dass während des standardkonformen Entpackens am Ende Ihrer Schlüsseldaten acht Bytes Padding-Daten hinzugefügt werden können. Extern verpackte Schlüssel können nicht ordnungsgemäß in eine Instanz entpackt werden. AWS CloudHSM
- **Problemumgehung:** Wenn Sie einen Schlüssel extern entpacken, der mit AES Key Wrapping und PKCS #5 Padding auf einer AWS CloudHSM-Instance verpackt wurde, entfernen Sie vor Verwendung des Schlüssels das zusätzliche Padding. Zu diesem Zweck können Sie die zusätzlichen Bytes in einem Datei-Editor abtrimmen oder nur die Schlüsselbytes in einen neuen Puffer in Ihrem Code kopieren.
- **Stand der Lösung:** Mit der Client- und Software-Version 3.1.0 bietet AWS CloudHSM standardkonforme Optionen für AES Key Wrapping. Weitere Informationen finden Sie unter [AES Key Wrapping](#).

Problem: Der Client-Daemon benötigt mindestens eine gültige IP-Adresse in seiner Konfigurationsdatei, um erfolgreich eine Verbindung mit dem Cluster herstellen zu können.

- **Auswirkung:** Wenn Sie alle HSM in Ihrem Cluster löschen und dann ein weiteres HSM hinzufügen, das eine neue IP-Adresse erhält, sucht der Client-Daemon weiterhin HSMs unter den ursprünglichen IP-Adressen nach Ihnen.
- **Umgehung:** Wenn Sie einen intermittierenden Workload ausführen, empfehlen wir, das `IpAddress` Argument in der [CreateHsm](#) Funktion zu verwenden, um das elastic network interface (ENI) auf seinen ursprünglichen Wert zu setzen. Beachten Sie, dass eine ENI spezifisch für eine Availability Zone (AZ) ist. Die Alternative ist, die Datei `/opt/cloudhsm/daemon/1/cluster.info` zu löschen und dann die Client-Konfiguration auf die IP-Adresse Ihres neuen HSM zurückzusetzen. Sie können den Befehl `client -a <IP address>` verwenden. Weitere Informationen finden [Sie unter Installation und Konfiguration des AWS CloudHSM Clients \(Linux\)](#) oder [Installation und Konfiguration des AWS CloudHSM Clients \(Windows\)](#).

Problem: Es gab eine Obergrenze von 16 KB für Daten, die AWS CloudHSM mit dem Client-SDK 3 gehasht und signiert werden können

- **Stand der Lösung:** Daten mit weniger als 16 KB werden weiterhin zum Versehen mit einem Hash-Wert an HSM gesendet. Daten zwischen 16 KB und 64 KB können nun auch lokal in der Software mit einem Hash-Wert versehen werden. Das Client-SDK 5 schlägt explizit fehl, wenn der Datenpuffer größer als 64 KB ist. Sie müssen Ihren Client und Ihre SDK (s) auf eine höhere Version als 5.0.0 oder höher aktualisieren, um von dem Fix zu profitieren.

Problem: Importierte Schlüssel konnten nicht als nicht exportierbar angegeben werden

- **Stand der Lösung:** Dieses Problem wurde behoben. Ihrerseits sind keine Maßnahmen erforderlich, um die Korrektur nutzen zu können.

Problem: Der Standardmechanismus für den WrapKey und die unWrapKey Befehle in key_mgmt_util wurde entfernt

- **Lösung:** Wenn Sie den WrapKey oder unWrapKey Befehle verwenden, müssen Sie die -m Option verwenden, um den Mechanismus anzugeben. Weitere Informationen finden Sie in den Beispielen im [WrapKey](#) oder in den [unWrapKey](#) Artikeln.

Problem: Befindet sich ein einzelnes HSM in Ihrem Cluster, funktioniert HSM-Failover nicht ordnungsgemäß.

- **Auswirkung:** Wenn die einzelne HSM-Instance im Cluster an Konnektivität verliert, stellt der Client keine erneute Verbindung zu ihr her – auch dann nicht, wenn die HSM-Instance zu einem späteren Zeitpunkt wiederhergestellt wird.
- **Problemumgehung:** Wir empfehlen mindestens zwei HSM-Instances pro Produktions-Cluster. Wenn Sie diese Konfiguration verwenden, werden Sie nicht von diesem Problem betroffen sein. Deaktivieren Sie bei Clustern mit nur einem HSM den Client-Daemon, um die Konnektivität wiederherzustellen.
- **Stand der Lösung:** Dieses Problem wurde in AWS CloudHSM -Client Version 1.1.2 behoben. Sie müssen auf diesen Client upgraden, um das Problem zu beheben.

Problem: Wenn Sie innerhalb kurzer Zeit die Schlüsselkapazität von HSMs in Ihrem Cluster überschreiten, geht der Client in einen unbehandelten Fehlerstatus über

- **Auswirkung:** Wenn der Client in den Status „Unbehandelter Fehler“ wechselt, hängt er sich auf und muss neu gestartet werden.
- **Problemumgehung:** (Problemumgehung) Testen Sie Ihren Durchsatz, um sicherzustellen, dass Sitzungsschlüssel nicht mit einer Rate erstellt werden, die der Client nicht verarbeiten kann. Sie können die Rate heruntersetzen, indem Sie dem Cluster ein HSM hinzufügen oder die Erstellung des Sitzungsschlüssels verzögern.
- **Stand der Lösung:** Dieses Problem wurde in AWS CloudHSM -Client Version 1.1.2 behoben. Sie müssen auf diesen Client upgraden, um das Problem zu beheben.

Problem: Digest-Operationen mit HMAC-Schlüsseln mit einer Größe von mehr als 800 Bytes werden nicht unterstützt.

- **Auswirkung:** HMAC-Schlüssel mit einer Größe von mehr als 800 Bytes können im HSM erstellt oder in das HSM importiert werden. Wenn Sie diesen größeren Schlüssel in einer Digest-Operation jedoch über die JCE oder `key_mgmt_util` verwenden, schlägt die Operation fehl. Beachten Sie, dass HMAC-Schlüssel bei Verwendung PKCS11 auf eine Größe von 64 Byte begrenzt sind.
- **Problemumgehung:** Wenn Sie HMAC-Schlüssel für Digest-Operationen im HSM verwenden, stellen Sie sicher, dass die Schlüsselgröße weniger als 800 Bytes beträgt.
- **Stand der Lösung:** Keine Angabe.

Problem: Das im Client-SDK 3 enthaltene Tool `client_info` löscht den Inhalt des durch das optionale Ausgabeargument angegebenen Pfads

- **Auswirkung:** Alle vorhandenen Dateien und Unterverzeichnisse unter dem angegebenen Ausgabepfad können dauerhaft verloren gehen.
- **Problemumgehung:** Verwenden Sie das optionale Argument `-output` *path* nicht, wenn Sie das `client_info`-Tool verwenden.
- **Stand der Lösung:** Dieses Problem wurde in [Client SDK 3.3.2](#) Version behoben. Sie müssen auf diesen Client upgraden, um das Problem zu beheben.

Problem: Sie erhalten eine Fehlermeldung, wenn Sie das SDK-5-Configure Tool mit dem `--cluster-id`-Argument in containerisierten Umgebungen ausführen

Sie erhalten den folgenden Fehler, wenn Sie das Argument `--cluster-id` mit dem Configure Tool verwenden:

```
No credentials in the property bag
```

Dieser Fehler wird durch ein Update auf Version 2 (IMDSv2) für den Instanz-Metadatendienst verursacht. Weitere Informationen finden Sie in der Dokumentation zu [IMDSv2](#).

- **Auswirkung:** Dieses Problem betrifft Benutzer, die das Configure Tool auf SDK-Versionen 5.5.0 und höher in containerisierten Umgebungen ausführen und EC2-Instance-Metadaten zur Bereitstellung von Anmeldeinformationen verwenden.

- Umgehung: Legen Sie das PUT-Antwort-Hop-Limit auf mindestens zwei fest. Eine Anleitung dazu finden Sie unter [Konfiguration der Optionen für Instanz-Metadaten](#).

Problem: Sie erhalten die Fehlermeldung „Fehler beim Erstellen cert/key aus der bereitgestellten PFX-Datei. Fehler: NotPkcs 8“

- Problemumgehung: Sie können den benutzerdefinierten privaten SSL-Schlüssel mit dem Befehl `openssl pkcs8 -topk8 -inform PEM -outform PEM -in ssl_private_key -out ssl_private_key_pkcs8`
- Lösungsstatus: Dieses Problem wurde in der [Client-SDK-Version 5.12.0](#) behoben. Sie müssen auf diese Client-Version oder eine neuere Version aktualisieren, um von dem Fix zu profitieren.

Problem: Die ECDSA-Signatur schlägt ab SDK 5.16 mit dem Fehler „Ungültiger Mechanismus“ fehl

- Auswirkung: ECDSA-Signaturvorgänge schlagen fehl, wenn Hashfunktionen verwendet werden, die schwächer als die Schlüsselstärke sind. Dieser Fehler tritt auf, weil [FIPS 186-5](#) erfordert, dass die Hash-Funktion mindestens so stark ist wie die Schlüsselstärke.

Möglicherweise wird in Ihren Client-Protokollen ein ähnlicher Fehler angezeigt:

```
[cloudhsm_provider::hsm1::session::ecdsa::sign::common][[]] Digest security strength (80) is weaker than the key security strength (128)
```

- Problemumgehung: Wenn Sie Ihre Hash-Funktionen nicht aktualisieren können, können Sie zu Nicht-FIPS-Clustern migrieren, bei denen die Anforderungen an die Hashstärke nicht eingehalten werden. Wir empfehlen jedoch, Ihre Hash-Funktionen zu aktualisieren, um die FIPS-Konformität aufrechtzuerhalten.

Als zusätzliche Problemumgehung haben wir eine Konfigurationsoption hinzugefügt, um diese Anforderung zu umgehen. Bitte beachten Sie, dass diese Option nicht empfohlen wird, da die Verwendung von ECDSA mit schwächeren Hash-Funktionen nicht den bewährten Sicherheitsmethoden entspricht. Um diese Option zu verwenden, führen Sie den folgenden Befehl aus (configure-`cli` setzen Sie ihn durch das Konfigurationstool für das verwendete SDK): [the section called “Syntax”](#)

```
sudo /opt/cloudhsm/bin/configure-cli --enable-ecdsa-with-weak-hash-function
```

- Lösung: Verwenden Sie eine Hash-Funktion, die mindestens so stark ist wie Ihr ECDSA-Schlüssel. Informationen zur Hash-Funktion und zu den wichtigsten Stärken von ECDSA finden Sie in den Tabellen 2 und 3 in [NIST SP 800-57](#) Teil 1 Rev 5.

Problem: Beim Signieren mit vorgehashten Daten werden Sitzungstoken im interaktiven Modus nicht ordnungsgemäß gelöscht

- Auswirkung: Wenn Sie die CloudHSM-CLI im interaktiven Modus mit SDK-Version 5.16.1 verwenden, können beim Signieren mit vorgehashten Daten Sitzungstoken nicht ordnungsgemäß gelöscht werden.
- Umgehung: Verwenden Sie den Einzelbefehlsmodus anstelle des interaktiven Modus, wenn Sie Signaturvorgänge mit vorgehashten Daten ausführen. Dadurch wird gewährleistet, dass das Token nach jedem Vorgang ordnungsgemäß bereinigt wird.
- Lösungsstatus: Dieses Problem wurde in CloudHSM SDK 5.16.2 behoben. Führen Sie ein Upgrade auf Version 5.16.2 oder höher durch, um von dem Update zu profitieren.

Problem: Das Standard-Client-Zertifikat der CloudHSM-Clientbibliothek läuft am 31. Januar 2026 ab

- Auswirkung: Nach dem 31. Januar 2026 gibt es keine Auswirkungen auf Kunden. [Die gesamte Kommunikation zwischen dem Client und HSM ist durch Client-HSM TLS gesichert, wie hier beschrieben.](#) Das Client-HSM TLS verwendet owned/managed Kundenzertifikate, die vom Kunden bei der Cluster-Initialisierung konfiguriert werden.
- Lösungsstatus: Gelöst.

Bekannte Probleme für AWS CloudHSM hsm1.medium-Instances

Die folgenden Probleme betreffen alle AWS CloudHSM hsm1.medium-Instances.

Themen

- [Problem: Das HSM kann nicht mehr als 250 Benutzer erstellen](#)

Problem: Das HSM kann nicht mehr als 250 Benutzer erstellen

- Problemumgehung: Dieses Problem wurde für die Instance-Typen AWS CloudHSM hsm2m.medium behoben.
- Stand der Lösung: Keine Angabe.

Bekannte Probleme für hsm2m.medium-Instances AWS CloudHSM

Die folgenden Probleme betreffen alle hsm2m.medium-Instanzen. AWS CloudHSM

Themen

- [Problem: Erhöhte Anmeldelatenz auf hsm2m.medium](#)
- [Problem: Erhöhte Latenz bei der Suche nach Schlüsseln auf hsm2m.medium](#)
- [Problem: Ein CO, der versucht, das vertrauenswürdige Attribut eines Schlüssels festzulegen, schlägt mit dem Client-SDK 5.12.0 und früher fehl](#)
- [Problem: Die ECDSA-Überprüfung schlägt mit Client SDK 5.12.0 und früher für Cluster im FIPS-Modus fehl](#)
- [Problem: Nur Zertifikate im PEM-Format können mit der CloudHSM-CLI als MTLS-Vertrauensanker registriert werden](#)
- [Problem: Kundenanwendungen beenden die Verarbeitung aller Anfragen, wenn mTLS mit einem durch Passphrase geschützten privaten Client-Schlüssel verwendet wird.](#)
- [Problem: Die Benutzerreplikation schlägt fehl, wenn die CloudHSM-CLI verwendet wird](#)
- [Problem: Während der Backup-Erstellung können Operationen fehlschlagen](#)
- [Problem: Das Client-SDK 5.8 und höher führen in einigen Szenarien auf hsm2m.medium keine automatischen Wiederholungsversuche für HSM-gedrosselte Operationen durch](#)
- [Problem: AES/CBC Unwrap-Operationen mit allen Zero-IV-Werten schlagen auf hsm2m.medium fehl](#)
- [Problem: Fehler beim Initialisieren der HSM-Verbindung bei Kaltstarts der Anwendung auf hsm2m.medium](#)

Problem: Erhöhte Anmeldelatenz auf hsm2m.medium

- Auswirkung: Die Anmeldung bei hsm2m.medium folgt einer zu strengen Interpretation der Compliance-Anforderungen, was zu einer erhöhten Latenz führt.

- Lösung: Wenn Sie vor dem 20. Dezember 2025 eine neue hsm2m.medium-Instance erstellt oder von hsm1.medium zu hsm2m.medium migriert haben, müssen Sie Ihr Passwort zurücksetzen, um die Leistungsverbesserungen nutzen zu können, die wir für Anmeldevorgänge implementiert haben. [Anweisungen finden Sie unter Passwort ändern.](#)

Problem: Erhöhte Latenz bei der Suche nach Schlüsseln auf hsm2m.medium

- Auswirkung: Die HSM-Instance hsm2m.medium hat eine verbesserte Fair-Share-Architektur, was im Vergleich zu hsm1.medium zu einer konsistenteren vorhersehbaren Leistung führt. Bei hsm1.medium können Kunden aufgrund der unregelmäßigen Nutzung von HSM-Ressourcen eine höhere Leistung bei der Suche nach Schlüsseln feststellen. Die Leistung von hsm1.medium für die Schlüsselsuche nimmt jedoch ab, wenn die HSM-Instanz gepatcht oder mit neuer Firmware aktualisiert wird. Dieses Problem betrifft Operationen wie in JCE. `KeyStore.getKey()`
- Lösung: Dieses Problem wurde behoben. Es hat sich bewährt, die Ergebnisse von Suchoperationen zwischenspeichern. Durch das Zwischenspeichern wird die Gesamtzahl der Schlüsselsuchvorgänge reduziert, da es sich um einen ressourcenintensiven Vorgang in HSM handelt. Implementieren Sie außerdem clientseitige Wiederholungsversuche mit exponentiellem Backoff und Jitter, um Ausfälle bei der HSM-Drosselung zu reduzieren.

Problem: Ein CO, der versucht, das vertrauenswürdige Attribut eines Schlüssels festzulegen, schlägt mit dem Client-SDK 5.12.0 und früher fehl

- Auswirkung: Jeder CO-Benutzer, der versucht, das vertrauenswürdige Attribut eines Schlüssels festzulegen, erhält eine entsprechende Fehlermeldung. `User type should be CO or CU`
- Lösung: Zukünftige Versionen des Client-SDK werden dieses Problem beheben. Updates werden in unseren Benutzerhandbüchern angekündigt [Dokumentverlauf](#).

Problem: Die ECDSA-Überprüfung schlägt mit Client SDK 5.12.0 und früher für Cluster im FIPS-Modus fehl

- Auswirkung: Der im FIPS-Modus ausgeführte ECDSA-Überprüfungsvorgang schlägt fehl. HSMs
- Lösungsstatus: Dieses Problem wurde in der Version 5.13.0 des [Client-SDK](#) behoben. Sie müssen auf diese Client-Version oder eine neuere Version aktualisieren, um von dem Fix zu profitieren.

Problem: Nur Zertifikate im PEM-Format können mit der CloudHSM-CLI als mTLS-Vertrauensanker registriert werden

- Auswirkung: Zertifikate im DER-Format können nicht als mTLS-Vertrauensanker mit der CloudHSM-CLI registriert werden.
- Umgehung: Sie können ein Zertifikat im DER-Format mit dem Befehl `openssl x509 -inform DER -outform PEM -in certificate.der -out certificate.pem` konvertieren.

Problem: Kundenanwendungen beenden die Verarbeitung aller Anfragen, wenn mTLS mit einem durch Passphrase geschützten privaten Client-Schlüssel verwendet wird.

- Auswirkung: Alle von der Anwendung ausgeführten Operationen werden angehalten und der Benutzer wird während der gesamten Lebensdauer der Anwendung mehrmals zur Eingabe der Passphrase auf der Standardeingabe aufgefordert. Bei Vorgängen kommt es zu einem Timeout und sie schlagen fehl, wenn die Passphrase nicht vor Ablauf der Zeitüberschreitungsdauer des Vorgangs angegeben wird.
- Problemumgehung: Mit Passphrase verschlüsselte private Schlüssel werden für mTLS nicht unterstützt. Entfernen Sie die Passphrase-Verschlüsselung aus dem privaten Schlüssel des Clients

Problem: Die Benutzerreplikation schlägt fehl, wenn die CloudHSM-CLI verwendet wird

- Auswirkung: Die Benutzerreplikation schlägt auf hsm2m.medium-Instances fehl, wenn die CloudHSM-CLI verwendet wird. Der `user replicate` Befehl funktioniert auf hsm1.medium-Instances erwartungsgemäß.
- Lösung: Dieses Problem wurde behoben.

Problem: Während der Backup-Erstellung können Operationen fehlschlagen

- Auswirkung: Vorgänge wie das Generieren von Zufallszahlen können auf hsm2m.medium-Instances fehlschlagen, während AWS CloudHSM ein Backup erstellt.
- Lösung: Implementieren Sie die folgenden bewährten Methoden, um Serviceunterbrechungen zu minimieren:

- Erstellen Sie einen Multi-HSM-Cluster
- Konfigurieren Sie Ihre Anwendungen so, dass Clustervorgänge wiederholt werden

Weitere Informationen zu bewährten Methoden, finden Sie unter [Bewährte Methoden für AWS CloudHSM](#).

Problem: Das Client-SDK 5.8 und höher führen in einigen Szenarien auf hsm2m.medium keine automatischen Wiederholungsversuche für HSM-gedrosselte Operationen durch

- **Auswirkung:** Beim Client-SDK 5.8 und höher werden einige HSM-gedrosselte Operationen nicht wiederholt
- **Problemumgehung:** Folgen Sie den bewährten Methoden, um Ihren Cluster so zu gestalten, dass er die Last bewältigt und Wiederholungsversuche auf Anwendungsebene implementiert. Wir arbeiten derzeit an einer Lösung. Updates werden in unseren Benutzerhandbüchern angekündigt [Dokumentverlauf](#).
- **Lösungsstatus:** Dieses Problem wurde im AWS CloudHSM Client SDK 5.16.2 behoben. Sie müssen ein Upgrade auf diese Client-Version oder eine neuere Version durchführen, um von dem Update zu profitieren.

Problem: AES/CBC Unwrap-Operationen mit allen Zero-IV-Werten schlagen auf hsm2m.medium fehl

- **Auswirkung:** Wenn der AES/CBC Mechanismus zum Entpacken von Schlüsseln mithilfe des AWS CloudHSM JCE-Anbieters verwendet wird, schlagen Operationen mit einer mit Null gefüllten 16-Byte-IV auf hsm2m.medium-Instances aufgrund einer zusätzlichen Validierungsprüfung fehl, die in hsm1.medium-Instances nicht durchgeführt wurde.
- **Status der Lösung:** Wir arbeiten an einer Lösung, die es ermöglicht, dass Null-Byte-Werte beim Entpacken akzeptiert werden. IVs AES/CBC

Problem: Fehler beim Initialisieren der HSM-Verbindung bei Kaltstarts der Anwendung auf hsm2m.medium

- **Auswirkung:** Dieses Problem betrifft Kaltstarts wie Bereitstellungen oder Neustarts von Client-Anwendungen. Die HSM-Instance hsm2m.medium verfügt über eine verbesserte Fair-

Share-Architektur, die eine konsistentere Leistung, Durchsatz und Latenz für alle Kunden gewährleistet. Derzeit stellen Sie auf hsm1.medium möglicherweise eine höhere Leistung als beabsichtigt bei der gleichzeitigen Initialisierung von HSM-Verbindungen fest. Die Leistung der Verbindungsinitialisierung von hsm1.medium hängt jedoch von den zugrunde liegenden Systemupdates ab.

- Lösung: Halten Sie sich an [bewährte Methoden und staffeln](#) Sie die Bereitstellungen und Neustarts von Clientanwendungen, um die Anzahl der Clientanwendungen zu begrenzen, die HSM-Verbindungen gleichzeitig initialisieren. Wir empfehlen außerdem, Wiederholungsversuche auf Anwendungsebene für die Initialisierung von Client-Anwendungen zu implementieren. Verwenden Sie außerdem das Konfigurationstool mit, `--cluster-id <cluster ID>` um der Client-Konfigurationsdatei alle HSM-IP-Adressen hinzuzufügen. Dieses Verhalten wurde in AWS CloudHSM Client SDK Version 5.17.1 und höher verbessert. Wir empfehlen ein Upgrade auf die neueste SDK-Version, um von dieser Verbesserung zu profitieren.

Bekannte Probleme mit der PKCS #11 -Bibliothek für AWS CloudHSM

Die folgenden Probleme betreffen die PKCS #11 -Bibliothek für. AWS CloudHSM

Themen

- [Problem: Der AES-Schlüsselumbruch in Version 3.0.0 der PKCS #11 -Bibliothek wird vor der Verwendung nicht validiert IVs](#)
- [Problem: PKCS #11 SDK 2.0.4 und frühere Versionen verwenden immer den Standard-IV von 0xA6A6A6A6A6A6A6A6 für AES-Schlüsselumbruch und -entpacken.](#)
- [Problem: Das Attribut CKA_DERIVE wurde nicht unterstützt und nicht verarbeitet.](#)
- [Problem: Das Attribut CKA_SENSITIVE wurde nicht unterstützt und nicht verarbeitet.](#)
- [Problem: Mehrteiliges Hashing und Signatur werden nicht unterstützt.](#)
- [Problem: C_GenerateKeyPair verarbeitet CKA_MODULUS_BITS oder CKA_PUBLIC_EXPONENT in der privaten Vorlage nicht konform zu den Standards.](#)
- [Problem: Puffer für die API-Operationen C_Encrypt und C_Decrypt dürfen bei Verwendung des CKM_AES_GCM-Mechanismus 16 KB nicht überschreiten.](#)
- [Problem: Elliptic-Curve Diffie-Hellman- \(ECDH\) Schlüsselableitung wird teilweise innerhalb des HSM ausgeführt.](#)
- [Problem: Die Überprüfung von secp256k1-Signaturen schlägt auf Plattformen wie Cent und RHEL 6 fehl EL6 OS6](#)

- [Problem: Eine falsche Reihenfolge von Funktionsaufrufen führt zu undefinierten Ergebnissen, anstatt dass sie fehlschlagen](#)
- [Problem: Die schreibgeschützte Sitzung wird in SDK 5 nicht unterstützt](#)
- [Problem: Die cryptoki.h-Header-Datei ist nur für Windows verfügbar](#)

Problem: Der AES-Schlüsselumbruch in Version 3.0.0 der PKCS #11 -Bibliothek wird vor der Verwendung nicht validiert IVs

Wenn Sie einen IV angeben, der kürzer als 8 Byte ist, wird er vor der Verwendung mit unvorhersehbaren Bytes aufgefüllt.

 Note

Dies wirkt sich nur C_WrapKey mit CKM_AES_KEY_WRAP-Mechanismus aus.

- **Auswirkung:** Wenn Sie in der Version 3.0.0 der PKCS #11-Bibliothek eine IV angeben, die kürzer als 8 Byte ist, können Sie den Schlüssel möglicherweise nicht entpacken.
- **Problemumgehungen:**
 - Wir empfehlen Ihnen dringend, auf die Version 3.0.1 oder höher der PKCS #11-Bibliothek zu aktualisieren, die die IV-Länge während des AES-Schlüsselumbruchs korrekt durchsetzt. Ändern Sie Ihren Umbruchcode, um einen NULL-IV zu übergeben, oder geben Sie den Standard-IV von 0xA6A6A6A6A6A6A6A6 an. Weitere Informationen finden Sie unter [Benutzerdefiniert IVs mit nicht kompatibler Länge für AES-Schlüsselumbruch](#).
 - Wenn Sie mit der Version 3.0.0 der PKCS #11-Bibliothek Schlüssel mit einer IV von weniger als 8 Byte umgewandelt haben, wenden Sie sich bitte an uns, um [Unterstützung](#) zu erhalten.
- **Auflösungsstatus:** Dieses Problem wurde in Version 3.0.1 der PKCS #11-Bibliothek behoben. Um Schlüssel mit AES-Schlüsselumbruch zu umschließen, geben Sie einen IV an, der NULL oder 8 Byte lang ist.

Problem: PKCS #11 SDK 2.0.4 und frühere Versionen verwenden immer den Standard-IV von **0xA6A6A6A6A6A6A6A6** für AES-Schlüsselumbruch und -entpacken.

Vom Benutzer bereitgestellte Informationen IVs wurden stillschweigend ignoriert.

 Note

Dies wirkt sich nur C_WrapKey mit CKM_AES_KEY_WRAP-Mechanismus aus.

- Auswirkung:
 - Wenn Sie PKCS #11 SDK 2.0.4 oder eine frühere Version und ein vom Benutzer bereitgestelltes IV verwendet haben, werden Ihre Schlüssel mit dem Standard-IV von 0xA6A6A6A6A6A6A6A6 umgebrochen.
 - Wenn Sie PKCS #11 SDK 3.0.0 oder höher und einen vom Benutzer bereitgestellten IV verwendet haben, werden Ihre Schlüssel mit dem vom Benutzer bereitgestellten IV umschlossen.
- Problemumgehungen:
 - Um Schlüssel zu entpacken, die mit PKCS #11 SDK 2.0.4 oder früher umschlossen sind, verwenden Sie den Standard-IV von 0xA6A6A6A6A6A6A6A6.
 - Um Schlüssel zu entpacken, die mit PKCS #11 SDK 3.0.0 oder höher umschlossen sind, verwenden Sie den vom Benutzer bereitgestellten IV.
- Auflösungsstatus: Wir empfehlen dringend, dass Sie Ihren Umbruch- und Entpackungscode so ändern, dass ein NULL-IV übergeben wird, oder geben Sie den Standard-IV von 0xA6A6A6A6A6A6A6A6 an.

Problem: Das Attribut **CKA_DERIVE** wurde nicht unterstützt und nicht verarbeitet.

- Stand der Lösung: Wir haben Fehlerbehebungen implementiert, um CKA_DERIVE zu akzeptieren, wenn es auf FALSE festgelegt ist. Wenn CKA_DERIVE auf TRUE eingestellt ist, wird es erst dann unterstützt, wenn wir die Schlüsselableitungsfunktion AWS CloudHSM hinzufügen. Sie müssen Ihren Client und Ihr(e) SDK(s) auf Version 1.1.1 oder höher aktualisieren, um dieses Problem zu beheben.

Problem: Das Attribut **CKA_SENSITIVE** wurde nicht unterstützt und nicht verarbeitet.

- Stand der Lösung: Wir haben Korrekturen implementiert, mit denen das Attribut CKA_SENSITIVE akzeptiert und berücksichtigt wird. Sie müssen Ihren Client und Ihr(e) SDK(s) auf Version 1.1.1 oder höher aktualisieren, um dieses Problem zu beheben.

Problem: Mehrteiliges Hashing und Signatur werden nicht unterstützt.

- Auswirkung: `C_DigestUpdate` und `C_DigestFinal` werden nicht implementiert. `C_SignFinal` ist ebenfalls nicht implementiert und schlägt mit `CKR_ARGUMENTS_BAD` für einen Nicht-NULL-Puffer fehl.
- Umgehung: Hashen Sie Ihre Daten innerhalb Ihrer Anwendung und verwenden Sie die AWS CloudHSM nur zum Signieren des Hashs.
- Status der Lösung: Wir sind dabei, den Client zu reparieren und das mehrteilige Hashing korrekt SDKs zu implementieren. Aktualisierungen werden im AWS CloudHSM -Forum und auf der Seite des Versionsverlaufs bekanntgegeben.

Problem: `C_GenerateKeyPair` verarbeitet `CKA_MODULUS_BITS` oder `CKA_PUBLIC_EXPONENT` in der privaten Vorlage nicht konform zu den Standards.

- Auswirkung: `C_GenerateKeyPair` sollte `CKA_TEMPLATE_INCONSISTENT` zurückgeben, wenn die private Vorlage `CKA_MODULUS_BITS` oder `CKA_PUBLIC_EXPONENT` enthält. Stattdessen generiert es einen privaten Schlüssel, für den alle Nutzungsfelder auf `FALSE` gesetzt sind. Der Schlüssel kann nicht verwendet werden.
- Problemumgehung: Wir empfehlen, dass Ihre Anwendung die Nutzungsfeldwerte zusätzlich zum Fehlercode auswertet.
- Stand der Lösung: Wir implementieren Korrekturen, um die richtige Fehlermeldung zurückzugeben, wenn eine fehlerhafte private Schlüsselvorlage verwendet wird. Die aktualisierte PKCS#11-Bibliothek wird auf der Seite des Versionsverlaufs bekanntgegeben.

Problem: Puffer für die API-Operationen `C_Encrypt` und `C_Decrypt` dürfen bei Verwendung des `CKM_AES_GCM`-Mechanismus 16 KB nicht überschreiten.

AWS CloudHSM unterstützt keine mehrteilige AES-GCM-Verschlüsselung.

- Auswirkung: Sie können den `CKM_AES_GCM`-Mechanismus nicht zum Verschlüsseln von Daten verwenden, die größer als 16 KB sind.
- Problemumgehung: Sie können einen alternativen Mechanismus wie `CKM_AES_CBC` oder `CKM_AES_CBC_PAD` verwenden, oder Sie können Ihre Daten in Teile aufteilen und jedes Teil einzeln verschlüsseln. Wenn Sie `AES_GCM` verwenden, müssen Sie die Aufteilung Ihrer Daten und die anschließende Verschlüsselung verwalten. AWS CloudHSM führt für Sie

keine mehrteilige AES-GCM-Verschlüsselung durch. Beachten Sie, dass FIPS erfordert, dass der Initialisierungsvektor (IV) auf dem HSM generiert wird. AES-GCM. Deshalb unterscheidet sich der IV für jeden Teil Ihrer mit AES-GCM verschlüsselten Daten.

- **Stand der Lösung:** Wir korrigieren das SDK, sodass es explizit fehlschlägt, wenn der Datenpuffer zu groß ist. Wir geben für die `C_EncryptUpdate`- und `C_DecryptUpdate`-API-Operationen `CKR_MECHANISM_INVALID` zurück. Wir werten Alternativen zur Unterstützung größerer Puffer aus, ohne dabei eine mehrteilige Verschlüsselung verwenden zu müssen. Updates werden im AWS CloudHSM Forum und auf der Seite mit dem Versionsverlauf angekündigt.

Problem: Elliptic-Curve Diffie-Hellman- (ECDH) Schlüsselableitung wird teilweise innerhalb des HSM ausgeführt.

Der private EC-Schlüssel verbleibt zu jeder Zeit innerhalb des HSM, der Prozess zur Schlüsselableitung wird aber in mehreren Schritten durchgeführt. Dies hat zur Folge, dass auf dem Client Zwischenergebnisse eines jeden Schritts verfügbar sind.

- **Auswirkung:** In Client SDK 3 ist der mithilfe des `CKM_ECDH1_DERIVE` Mechanismus abgeleitete Schlüssel zunächst auf dem Client verfügbar und wird dann in das HSM importiert. Ein Schlüssel-Handle wird dann an Ihre Anwendung zurückgegeben.
- **Umgehung:** Wenn Sie SSL/TLS Offload in implementieren AWS CloudHSM, stellt diese Einschränkung möglicherweise kein Problem dar. Wenn Ihre Anwendung vorschreibt, dass Ihr Schlüssel zu jeder Zeit innerhalb einer FIPS-Begrenzung verbleibt, sollten Sie mit einem alternativen Protokoll ohne erforderliche ECDH-Schlüsselableitung arbeiten.
- **Status der Lösung:** SDK 5.16 unterstützt jetzt ECDH mit Key Derivation, die vollständig innerhalb des HSM ausgeführt wird.

Problem: Die Überprüfung von secp256k1-Signaturen schlägt auf Plattformen wie CentOS und RHEL 6 fehl. EL6 OS6

Der Grund hierfür ist, dass die PKCS #11-Bibliothek von CloudHSM während der Initialisierung des Überprüfungsvorgangs einen Netzwerkaufruf vermeidet, indem sie OpenSSL zur Überprüfung von EC-Kurvendaten verwendet. Da Secp256k1 vom Standard-OpenSSL-Paket auf EL6 Plattformen nicht unterstützt wird, schlägt die Initialisierung fehl.

- **Auswirkung:** Die Überprüfung der SecP256k1-Signatur schlägt auf Plattformen fehl. EL6 Der Überprüfungsaufruf schlägt fehl und es wird eine `CKR_HOST_MEMORY`-Fehlermeldung angezeigt.

- Umgehung: Wir empfehlen, entweder Amazon Linux 1 oder eine andere EL7 Plattform zu verwenden, wenn Ihre PKCS #11 -Anwendung secp256k1-Signaturen verifizieren muss. Alternativ können Sie ein Upgrade auf eine Version des OpenSSL-Pakets durchführen, das die secp256k1-Kurve unterstützt.
- Stand der Lösung: Wir implementieren zurzeit Korrekturen, die es ermöglichen, auf das HSM zurückzugreifen, wenn die lokale Kurvenvalidierung nicht verfügbar ist. Die aktualisierte PKCS #11-Bibliothek wird auf der Seite des [Versionsverlaufs](#) bekanntgegeben.

Problem: Eine falsche Reihenfolge von Funktionsaufrufen führt zu undefinierten Ergebnissen, anstatt dass sie fehlschlagen

- Auswirkung: Wenn Sie eine falsche Reihenfolge von Funktionen aufrufen, ist das Endergebnis falsch, obwohl die einzelnen Funktionsaufrufen erfolgreich sind. Beispielsweise stimmen entschlüsselte Daten möglicherweise nicht mit dem ursprünglichen Klartext überein oder Signaturen lassen sich möglicherweise nicht verifizieren. Dieses Problem betrifft sowohl einteilige als auch mehrteilige Operationen.

Beispiele für falsche Funktionssequenzen:

- C_EncryptInit/C_EncryptUpdate gefolgt von C_Encrypt
- C_DecryptInit/C_DecryptUpdate gefolgt von C_Decrypt
- C_SignInit/C_SignUpdate gefolgt von C_Sign
- C_VerifyInit/C_VerifyUpdate gefolgt von C_Verify
- C_FindObjectsInit gefolgt von C_FindObjectsInit
- Umgehung: Ihre Anwendung sollte gemäß der PKCS #11 -Spezifikation die richtige Reihenfolge von Funktionsaufrufen sowohl für einteilige als auch für mehrteilige Operationen verwenden. Ihre Anwendung sollte sich nicht darauf verlassen, dass die CloudHSM PKCS #11-Bibliothek unter diesen Umständen einen Fehler zurückgibt.

Problem: Die schreibgeschützte Sitzung wird in SDK 5 nicht unterstützt

- Problem: SDK 5 unterstützt nicht das Öffnen von Nur-Lesesitzungen mit C_OpenSession.
- Auswirkung: Wenn Sie versuchen, einen Aufruf an C_OpenSession ohne Angabe von CKF_RW_SESSION zu tätigen, schlägt der Aufruf mit der folgenden Fehlermeldung CKR_FUNCTION_FAILED fehl.

- Problemumgehung: Wenn Sie eine Sitzung öffnen, müssen Sie die CKF_SERIAL_SESSION | CKF_RW_SESSION-Flags an den C_OpenSession-Funktionsaufruf übergeben.

Problem: Die **cryptoki.h**-Header-Datei ist nur für Windows verfügbar

- Problem: Bei den AWS CloudHSM Client SDK 5-Versionen 5.0.0 bis 5.4.0 unter Linux ist die Header-Datei /opt/cloudhsm/include/pkcs11/cryptoki.h nur mit Windows-Betriebssystemen kompatibel.
- Auswirkung: Unter Linux-basierten Betriebssystemen können Probleme auftreten, wenn Sie versuchen, diese Header-Datei in Ihre Anwendung aufzunehmen.
- Lösungsstatus: Führen Sie ein Upgrade auf AWS CloudHSM Client SDK 5 Version 5.4.1 oder höher durch, die eine Linux-kompatible Version dieser Header-Datei enthält.

Bekannte Probleme mit dem JCE SDK für AWS CloudHSM

Die folgenden Probleme betreffen das JCE SDK für AWS CloudHSM

Themen

- [Problem: Bei der Arbeit mit asymmetrischen Schlüsselpaaren wird die belegte Schlüsselkapazität auch dann angezeigt, wenn Sie Schlüssel nicht explizit erstellen oder importieren.](#)
- [Problem: Das JCE KeyStore ist schreibgeschützt](#)
- [Problem: Puffer für die AES-GCM-Verschlüsselung dürfen nicht größer sein als 16.000 Byte.](#)
- [Problem: Elliptic-Curve Diffie-Hellman- \(ECDH\) Schlüsselableitung wird teilweise innerhalb des HSM ausgeführt.](#)
- [Problem: KeyGenerator und interpretiert den Schlüsselgrößenparameter KeyAttribute fälschlicherweise als Anzahl von Bytes statt als Bits](#)
- [Problem: Das Client-SDK 5 gibt die Warnung „Es ist ein illegaler Reflective Access-Vorgang aufgetreten“ aus](#)
- [Problem: Der JCE-Sitzungspool ist erschöpft](#)
- [Problem: Speicherverlust im Client-SDK 5 bei GetKey-Vorgängen](#)

Problem: Bei der Arbeit mit asymmetrischen Schlüsselpaaren wird die belegte Schlüsselkapazität auch dann angezeigt, wenn Sie Schlüssel nicht explizit erstellen oder importieren.

- **Auswirkung:** Dieses Problem kann dazu führen HSMs , dass Ihnen unerwartet der Schlüsselspeicher ausgeht. Es tritt auf, wenn Ihre Anwendung statt eines Objekts ein Standard-JCE-Schlüsselobjekt für Kryptooperationen verwendet. `CaviumKey` Wenn Sie ein JCE-Standardschlüsselobjekt verwenden, importiert `CaviumProvider` diesen Schlüssel implizit in das HSM, da ein Sitzungsschlüssel diesen Schlüssel erst löscht, wenn die Anwendung beendet wird. Infolgedessen sammeln sich Schlüssel an, während die Anwendung ausgeführt wird, und dies kann dazu führen, dass Ihnen HSMs der freie Schlüsselspeicher ausgeht, sodass Ihre Anwendung einfriert.
- **Problemumgehung:** Wenn Sie die Klasse `CaviumSignature`, `CaviumCipher`, `CaviumMac` oder `CaviumKeyAgreement` verwenden, sollten Sie den Schlüssel als `CaviumKey` und nicht als JCE-Standardschlüsselobjekt bereitstellen.

Sie können mithilfe der [ImportKey](#)-Klasse einen normalen Schlüssel manuell in einen `CaviumKey` umwandeln und den Schlüssel dann nach Abschluss des Vorgangs manuell löschen.

- **Stand der Lösung:** Wir aktualisieren den `CaviumProvider`, um implizite Importe ordnungsgemäß zu verwalten. Diese Fehlerbehebung wird auf der Seite des Versionsverlaufs bekanntgegeben, sobald sie verfügbar ist.

Problem: Das JCE KeyStore ist schreibgeschützt

- **Auswirkungen:** Derzeit können im JCE KeyStore nur Objekttypen gespeichert werden, die von HSM unterstützt werden. Insbesondere ist es nicht möglich, Zertifikate im Schlüsselspeicher zu speichern. Dadurch wird die Interoperabilität mit Tools wie `jarsigner`, die im Schlüsselspeicher ein Zertifikat erwarten, verhindert.
- **Problemumgehung:** Sie können Änderungen am Code vornehmen, sodass Zertifikate anstatt aus dem Schlüsselspeicher aus lokalen Dateien oder aus einem S3-Bucket-Speicherort geladen werden.
- **Lösungsstatus:** Sie können den AWS CloudHSM Keystore zum Speichern von Zertifikaten verwenden.

Problem: Puffer für die AES-GCM-Verschlüsselung dürfen nicht größer sein als 16.000 Byte.

Es wird keine mehrteilige AES-GCM-Verschlüsselung unterstützt.

- **Auswirkung:** Sie können AES-GCM nicht zum Verschlüsseln von Daten verwenden, die größer als 16.000 Byte sind.
- **Problemumgehung:** Sie können einen alternativen Mechanismus verwenden, wie beispielsweise AES-CBC, oder Sie können Ihre Daten unterteilen und jeden Teil einzeln verschlüsseln. Wenn Sie die Daten unterteilen, müssen Sie den unterteilten Verschlüsselungstext und dessen Entschlüsselung verwalten. Da FIPS erfordert, dass der Initialisierungsvektor (IV) für AES-GCM auf dem HSM generiert wird, ist der IV für jedes Datenelement unterschiedlich. AES-GCM-encrypted
- **Stand der Lösung:** Wir korrigieren das SDK, sodass es explizit fehlschlägt, wenn der Datenpuffer zu groß ist. Wir werten Alternativen zur Unterstützung größerer Puffer aus, ohne eine mehrteilige Verschlüsselung verwenden zu müssen. Aktualisierungen werden im AWS CloudHSM -Forum und auf der Seite des Versionsverlaufs bekanntgegeben.

Problem: Elliptic-Curve Diffie-Hellman- (ECDH) Schlüsselableitung wird teilweise innerhalb des HSM ausgeführt.

Der private EC-Schlüssel verbleibt zu jeder Zeit innerhalb des HSM, der Prozess zur Schlüsselableitung wird aber in mehreren Schritten durchgeführt. Dies hat zur Folge, dass auf dem Client Zwischenergebnisse eines jeden Schritts verfügbar sind. Ein ECDH-Schlüsselableitungsbeispiel ist in den [Java-Codebeispielen](#) verfügbar.

- **Auswirkung:** Das Client-SDK 3 erweitert das JCE um ECDH-Funktionalität. Wenn Sie die `KeyAgreement` Klasse verwenden, um eine abzuleiten `SecretKey`, ist sie zunächst auf dem Client verfügbar und wird dann in das HSM importiert. Ein Schlüssel-Handle wird dann an Ihre Anwendung zurückgegeben.
- **Problemumgehung:** Wenn Sie SSL/TLS Offload in implementieren AWS CloudHSM, ist diese Einschränkung möglicherweise kein Problem. Wenn Ihre Anwendung vorschreibt, dass Ihr Schlüssel zu jeder Zeit innerhalb einer FIPS-Begrenzung verbleibt, sollten Sie mit einem alternativen Protokoll ohne erforderliche ECDH-Schlüsselableitung arbeiten.
- **Status der Lösung:** SDK 5.16 unterstützt jetzt ECDH mit Key Derivation, die vollständig innerhalb des HSM ausgeführt wird.

Problem: KeyGenerator und interpretiert den Schlüsselgrößenparameter KeyAttribute fälschlicherweise als Anzahl von Bytes statt als Bits

Beim Generieren eines Schlüssels mithilfe der `init` Funktion der [KeyGenerator Klasse](#) oder des `SIZE` Attributs der [AWS CloudHSM KeyAttribute Enumeration](#) erwartet die API fälschlicherweise, dass das Argument die Anzahl der Schlüsselbytes ist, obwohl es stattdessen die Anzahl der Schlüsselbits sein sollte.

- **Auswirkung:** Die Client-SDK-Versionen 5.4.0 bis 5.4.2 erwarten fälschlicherweise, dass die Schlüsselgröße in Byte angegeben APIs wird.
- **Umgehung:** Wenn Sie die Client-SDK-Versionen 5.4.0 bis 5.4.2 verwenden, konvertieren Sie die Schlüsselgröße von Bits in Byte, bevor Sie die KeyGenerator Klasse oder KeyAttribute Enum verwenden, um Schlüssel mithilfe des AWS CloudHSM JCE-Anbieters zu generieren.
- **Lösungsstatus:** Führen Sie ein Upgrade Ihrer Client-SDK-Version auf Version 5.5.0 oder höher durch. Dies beinhaltet einen Fix, mit dem Schlüsselgrößen in Bits korrekt erwartet werden, wenn Sie die KeyGenerator Klasse oder die Enumeration zum Generieren von Schlüsseln verwenden. KeyAttribute

Problem: Das Client-SDK 5 gibt die Warnung „Es ist ein illegaler Reflective Access-Vorgang aufgetreten“ aus

Bei Verwendung von Client-SDK 5 mit Java 11 gibt CloudHSM die folgende Java-Warnung aus:

```
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
```

Dieses Problem wurde in der Client-SDK-Version 5.8 und höher behoben.

Problem: Der JCE-Sitzungspool ist erschöpft

Auswirkung: Sie können möglicherweise keine Operationen in JCE ausführen, wenn die folgende Meldung angezeigt wird:

```
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
```

Problemumgehungen:

- Starten Sie Ihre JCE-Anwendung neu, falls Probleme auftreten.
- Wenn Sie einen Vorgang ausführen, müssen Sie den JCE-Vorgang möglicherweise abschließen, bevor Sie den Bezug zu dem Vorgang verlieren.

Note

Je nach Vorgang ist möglicherweise eine Abschlussmethode erforderlich.

Operation	Vervollständigungsmethode
Verschlüsselungsverfahren	<code>doFinal()</code> im Verschlüsselungs- oder Entschlüsselungsmodus <code>wrap()</code> im Wrap-Modus <code>unwrap()</code> im Unwrap-Modus
KeyAgreement	<code>generateSecret()</code> oder <code>generateSecret(String)</code>
KeyPairGenerator	<code>generateKeyPair()</code> , <code>genKeyPair()</code> oder <code>reset()</code>
KeyStore	Keine Methode erforderlich
MAC	<code>doFinal()</code> oder <code>reset()</code>

Operation	Vervollständigungsmethode
MessageDigest	<code>digest()</code> oder <code>reset()</code>
SecretKeyFactory	Keine Methode erforderlich
SecureRandom	Keine Methode erforderlich
Signatur	<code>sign()</code> im Signiermodus <code>verify()</code> im Überprüfungsmodus

Lösungsstatus: Wir haben dieses Problem in Client SDK 5.9.0 und höher behoben. Um dieses Problem zu beheben, aktualisieren Sie Ihr Client-SDK auf eine dieser Versionen.

Problem: Speicherverlust im Client-SDK 5 bei GetKey-Vorgängen

- **Auswirkung:** Bei der `getKey` API-Operation ist in JCE in den Client-SDK-Versionen 5.10.0 und früher ein Speicherverlust aufgetreten. Wenn Sie die `getKey` API in Ihrer Anwendung mehrfach verwenden, führt dies zu einem erhöhten Speicherwachstum und damit zu einer Erhöhung des Speicherbedarfs in Ihrer Anwendung. Im Laufe der Zeit kann dies zu Drosselungsfehlern führen oder einen Neustart der Anwendung erfordern.
- **Problemumgehung:** Wir empfehlen ein Upgrade auf Client SDK 5.11.0. Wenn dies nicht möglich ist, empfehlen wir, die `getKey` API in Ihrer Anwendung nicht mehrmals aufzurufen. Verwenden Sie stattdessen den zuvor zurückgegebenen Schlüssel aus der vorherigen `getKey` Operation so oft wie möglich wieder.
- **Lösungsstatus:** Aktualisieren Sie Ihre Client-SDK-Version auf 5.11.0 oder höher, was eine Lösung für dieses Problem beinhaltet.

Bekannte Probleme mit der OpenSSL Dynamic Engine für AWS CloudHSM

Dies sind die bekannten Probleme für OpenSSL Dynamic Engine for AWS CloudHSM.

Themen

- [Problem: Sie können AWS CloudHSM OpenSSL Dynamic Engine nicht auf RHEL 6 und Cent installieren OS6](#)
- [Problem: Standardmäßig wird nur die RSA-Auslagerung zum HSM unterstützt.](#)

- [Problem: RSA-Ver- und Entschlüsselung mit OAEP-Padding unter Verwendung eines Schlüssels im HSM wird nicht unterstützt.](#)
- [Problem: Nur die Generierung privaten Schlüssel der RSA- und ECC-Schlüssel wird in das HSM ausgelagert.](#)
- [Problem: Sie können OpenSSL Dynamic Engine für Client-SDK 3 nicht auf RHEL 8, CentOS 8 oder Ubuntu 18.04 LTS installieren](#)
- [Problem: SHA-1 Sign and Verify ist auf RHEL 9 \(9.2+\) als veraltet markiert](#)
- [Problem: AWS CloudHSM OpenSSL Dynamic Engine ist nicht mit dem FIPS-Anbieter für OpenSSL v3.x kompatibel](#)
- [Problem: Der SSL/TLS Offload schlägt mit ECDSA Cipher Suites in TLS 1.0 und TLS 1.1 ab SDK 5.16 fehl](#)

Problem: Sie können AWS CloudHSM OpenSSL Dynamic Engine nicht auf RHEL 6 und CentOS 6 installieren

- Auswirkung: Die OpenSSL Dynamic Engine [unterstützt nur OpenSSL 1.0.2\[f+\]](#). Standardmäßig werden RHEL 6 und CentOS 6 mit OpenSSL 1.0.1 ausgeliefert.
- Problemumgehung: Aktualisieren Sie die OpenSSL-Bibliothek auf RHEL 6 und CentOS 6 auf Version 1.0.2[f+].

Problem: Standardmäßig wird nur die RSA-Auslagerung zum HSM unterstützt.

- Auswirkung: Zur Maximierung der Leistung ist der SDK nicht für die Auslagerung zusätzlicher Funktionen, wie z. B. Zufallszahlengenerierung oder EC-DH-Operationen, konfiguriert.
- Problemumgehung: Bitte kontaktieren Sie uns mittels eines Supportfalls, wenn Sie zusätzliche Operationen auslagern möchten.
- Stand der Lösung: Wir arbeiten daran, den SDK um Unterstützung für das Konfigurieren von Auslagerungsoptionen über eine Konfigurationsdatei zu erweitern. Die Aktualisierung wird auf der Seite des Versionsverlaufs bekanntgegeben, sobald sie verfügbar ist.

Problem: RSA-Ver- und Entschlüsselung mit OAEP-Padding unter Verwendung eines Schlüssels im HSM wird nicht unterstützt.

- **Auswirkung:** Jeder Aufruf zur RSA-Verschlüsselung und -Entschlüsselung mit OAEP-Padding schlägt mit einem Fehler fehl. `divide-by-zero` Der Grund besteht darin, dass die dynamische OpenSSL-Engine die Operation lokal unter Verwendung der Fake-PEM-Datei aufruft, statt die Operation in das HSM auszulagern.
- **Problemumgehung:** Sie können diesen Vorgang in der [PKCS #11 -Bibliothek für AWS CloudHSM Client SDK 5](#) oder in der [JCE-Anbieter für AWS CloudHSM Client SDK 5](#) ausführen.
- **Stand der Lösung:** Wird fügen eine Unterstützung des SDK hinzu, damit diese Operation ordnungsgemäß ausgelagert wird. Die Aktualisierung wird auf der Seite des Versionsverlaufs bekanntgegeben, sobald sie verfügbar ist.

Problem: Nur die Generierung privaten Schlüssel der RSA- und ECC-Schlüssel wird in das HSM ausgelagert.

Für jeden anderen Schlüsseltyp wird die AWS CloudHSM OpenSSL-Engine nicht für die Anruferverarbeitung verwendet. Stattdessen wird die lokale OpenSSL-Engine verwendet. Dies generiert einen Schlüssel lokal in der Software.

- **Auswirkungen:** Da das Failover stillschweigend erfolgt, gibt es keinen Hinweis darauf, dass Sie keinen Schlüssel erhalten haben, der sicher im HSM generiert wurde. Sie sehen einen Ausgabeablauf, der die Zeichenkette "`.....++++++`" enthält, wenn der Schlüssel lokal von OpenSSL in der Software generiert wurde. Diese Abfolge ist nicht vorhanden, wenn die Operation in das HSM ausgelagert wurde. Da der Schlüssel nicht im HSM erstellt oder gespeichert, steht er für die zukünftige Nutzung nicht zur Verfügung.
- **Problemumgehung:** Verwenden Sie die OpenSSL-Engine nur für Schlüsseltypen, die sie unterstützt. Verwenden Sie für alle anderen Schlüsseltypen PKCS #11 oder JCE in Anwendungen oder `key_mgmt_util` in der CLI.

Problem: Sie können OpenSSL Dynamic Engine für Client-SDK 3 nicht auf RHEL 8, CentOS 8 oder Ubuntu 18.04 LTS installieren

- **Auswirkung:** RHEL 8, CentOS 8 und Ubuntu 18.04 LTS liefern standardmäßig eine Version von OpenSSL, die nicht mit OpenSSL Dynamic Engine für Client-SDK 3 kompatibel ist.

- Problemumgehung: Verwenden Sie eine Linux-Plattform, die OpenSSL Dynamic Engine unterstützt. Weitere Informationen zu unterstützten Plattformen finden Sie unter [Unterstützte Plattformen](#).
- Auflösungsstatus: AWS CloudHSM unterstützt diese Plattformen mit OpenSSL Dynamic Engine für Client SDK 5. Weitere Informationen finden Sie unter [Unterstützte Plattformen](#) und [OpenSSL Dynamic Engine](#).

Problem: SHA-1 Sign and Verify ist auf RHEL 9 (9.2+) als veraltet markiert

- Auswirkung: Die Verwendung des SHA-1 Message Digest für kryptografische Zwecke ist in RHEL 9 (9.2+) veraltet. Infolgedessen schlagen Signier- und Verifizierungsvorgänge mit SHA-1 unter Verwendung der OpenSSL Dynamic Engine fehl.
- Umgehung: [Wenn Ihr Szenario die Verwendung von SHA-1 für signing/verifying bestehende kryptografische Signaturen oder kryptografische Signaturen von Drittanbietern erfordert, finden Sie weitere Informationen unter Enhancing RHEL Security: Understanding SHA-1 Deprecation on RHEL 9 \(9.2+\) und RHEL 9 \(9.2+\) Versionshinweise.](#)

Problem: AWS CloudHSM OpenSSL Dynamic Engine ist nicht mit dem FIPS-Anbieter für OpenSSL v3.x kompatibel

- Auswirkung: Sie erhalten eine Fehlermeldung, wenn Sie versuchen, die AWS CloudHSM OpenSSL Dynamic Engine zu verwenden, wenn der FIPS-Anbieter für OpenSSL-Versionen 3.x aktiviert ist.
- Umgehung: Um die AWS CloudHSM OpenSSL Dynamic Engine mit OpenSSL Version 3.x zu verwenden, stellen Sie sicher, dass der „Standard“-Anbieter konfiguriert ist. Lesen Sie mehr über den Standardanbieter auf der [OpenSSL-Website](#).

Problem: Der SSL/TLS Offload schlägt mit ECDSA Cipher Suites in TLS 1.0 und TLS 1.1 ab SDK 5.16 fehl

- Auswirkung: [Verbindungsversuche mit TLS 1.0 oder TLS 1.1 schlagen fehl, weil diese Versionen SHA-1 zum Signieren verwenden, was die FIPS 186-5-Anforderungen nicht erfüllt.](#)
- Umgehung: Wenn Sie TLS-Versionen nicht sofort aktualisieren können, können Sie zu Nicht-FIPS-Clustern migrieren, die die Anforderungen an die Hashstärke nicht durchsetzen. Wir empfehlen jedoch, ein Upgrade auf TLS 1.2 oder TLS 1.3 durchzuführen, um die FIPS-Konformität und die bewährten Sicherheitsmethoden aufrechtzuerhalten.

- Lösung: Führen Sie ein Upgrade Ihrer Implementierung auf TLS 1.2 oder TLS 1.3 durch. Die Internet Engineering Task Force (IETF) hat TLS 1.0 und TLS 1.1 aus Sicherheitsgründen als [veraltet](#) eingestuft.

Bekannte Probleme beim Key Storage Provider (KSP) für AWS CloudHSM

Dies sind die bekannten Probleme für Key Storage Provider (KSP) für AWS CloudHSM

Themen

- [Problem: Die Überprüfung eines Zertifikatsspeichers schlägt fehl](#)
- [Problem: Inkonsistenz der Containernamen im Zertifikatsspeicher bei Verwendung des SDK3 Kompatibilitätsmodus für Client SDK 5](#)

Problem: Die Überprüfung eines Zertifikatsspeichers schlägt fehl

Wenn Sie die Client-SDK-Versionen 5.14 und 5.15 verwenden, wird beim Aufrufen der `certutil -store my CERTIFICATE_SERIAL_NUMBER` folgende Fehler ausgegeben:

```
ERROR: Could not verify certificate public key against private key
```

- Auswirkung: Sie können einen mit dem Client-SDK 5 erstellten Zertifikatsspeicher nicht `certutil` zur Validierung verwenden.
- Umgehung: Überprüfen Sie das mit dem Zertifikat verknüpfte key pair, indem Sie eine Datei mit dem privaten Schlüssel signieren und die Signatur mit dem öffentlichen Schlüssel überprüfen. Dies kann mit Microsoft erfolgen, SignTool indem Sie die [hier](#) angegebenen Schritte ausführen.
- Status der Lösung: Wir arbeiten daran, Unterstützung für die Überprüfung von Zertifikaten mit `certutil` hinzuzufügen. Diese Fehlerbehebung wird auf der Seite des Versionsverlaufs bekanntgegeben, sobald sie verfügbar ist.

Problem: Inkonsistenz der Containernamen im Zertifikatsspeicher bei Verwendung des SDK3 Kompatibilitätsmodus für Client SDK 5

Wenn Sie den `certutil -store my CERTIFICATE_SERIAL_NUMBER` Befehl verwenden, um Zertifikate anzuzeigen, deren Schlüsselreferenzdateien mit dem Befehl [generate-file](#) in AWS CLI 5.16.0 generiert wurden, tritt der folgende Fehler auf:


```
ERROR: Container name inconsistent: CONTAINER_NAME
```

Dieser Fehler tritt auf, weil es eine Diskrepanz zwischen dem im Zertifikat gespeicherten Container-Namen und dem von der CloudHSM-CLI generierten Schlüsselreferenzdatei gibt.

- **Auswirkung:** Trotz dieses Fehlers sind die Zertifikate und die zugehörigen Schlüssel weiterhin voll funktionsfähig. Alle Anwendungen, die diese Zertifikate verwenden, funktionieren weiterhin normal.
- **Problemumgehung:** Um diesen Fehler zu beheben, benennen Sie den Dateinamen der Schlüsselreferenz in Simple oder Unique Container Name um. Sehen Sie sich die folgende Beispielausgabe des Befehls `certutil -store my`

```
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
```

Standardmäßig werden die wichtigsten Referenzdateien in gespeichert C:\Users\Default\AppData\Roaming\Microsoft\Crypto\CaviumKSP\GlobalPartition

1. Benennen Sie die Schlüsselreferenzdatei in den einfachen Container-Namen um.
 2. Reparieren Sie den Zertifikatsspeicher mit dem neuen Namen des Schlüsselcontainers. Weitere Informationen finden Sie in den Schritten 12 bis 14 unter [KSP-Migration](#).
- **Lösungsstatus:** Dieses Problem wurde in der Client-SDK-Version 5.16.1 behoben. Um dieses Problem zu beheben, aktualisieren Sie Ihr Client-SDK auf Version 5.16.1 oder höher.

Bekannte Probleme für den OpenSSL Provider für AWS CloudHSM

Dies sind die bekannten Probleme für OpenSSL Provider for AWS CloudHSM.

Themen

- [Problem: Fehler in der OpenSSL-CLI bei Verwendung mit OpenSSL Provider](#)

Problem: Fehler in der OpenSSL-CLI bei Verwendung mit OpenSSL Provider

- Auswirkung: Die Integration mit OpenSSL CLI wird derzeit vom AWS CloudHSM OpenSSL Provider nicht unterstützt. Informationen zu unterstützten [the section called “Offload auf Linux mit OpenSSL Provider”](#) Integrationen finden Sie unter.
- Lösung: Verwenden Sie die unter aufgeführten unterstützten Integrationen. [the section called “Offload auf Linux mit OpenSSL Provider”](#) Alle Aktualisierungen der OpenSSL-CLI-Unterstützung werden auf der Seite mit dem Versionsverlauf angekündigt.

Bekannte Probleme für Amazon EC2 EC2-Instances, auf denen Amazon Linux 2 ausgeführt wird mit AWS CloudHSM

Die folgenden Probleme betreffen AWS CloudHSM Amazon EC2 EC2-Instances, die auf Amazon Linux 2 ausgeführt werden.

Problem: Amazon Linux 2 Version 2018.07 verwendet ein aktualisiertes `ncurses` Paket (Version 6), das derzeit nicht kompatibel ist mit AWS CloudHSM SDKs

[Beim Ausführen von AWS CloudHSM `cloudhsm\_mgmt\_util` oder `key\_mgmt\_util` wird der folgende Fehler angezeigt:](#)

```
/opt/cloudhsm/bin/cloudhsm_mgmt_util: error while loading shared libraries:  
libncurses.so.5: cannot open shared object file: No such file or directory
```

- Auswirkung: Instances, die auf Amazon Linux 2 Version 2018.07 ausgeführt werden, können nicht alle AWS CloudHSM Dienstprogramme verwenden.
- Problemumgehung: Geben Sie den folgenden Befehl an Ihre Amazon-Linux-2-EC2-Instances aus, um das unterstützte `ncurses`-Paket (Version 5) zu installieren:

```
sudo yum update && yum install ncurses-compat-libs
```

- Stand der Lösung: Dieses Problem wurde in AWS CloudHSM -Client Version 1.1.2 behoben. Sie müssen auf diesen Client upgraden, um das Problem zu beheben.

Bekannte Probleme bei der Integration von Drittanbieteranwendungen mit AWS CloudHSM

Die folgenden Probleme wirken sich auf AWS CloudHSM die Integration mit Drittanbieteranwendungen aus.

Problem: Client-SDK 3 unterstützt nicht das Setzen des PKCS #11-Attributs **CKA_MODIFIABLE** durch Oracle während der Hauptschlüsselerzeugung

Dieses Limit ist in der PKCS #11-Bibliothek definiert. Weitere Informationen finden Sie unter Anmerkung 1 zu [Unterstützte PKCS #11-Attribute](#).

- **Auswirkung:** Die Erstellung des Oracle-Masterschlüssels schlägt fehl.
- **Problemumgehung:** Setzen Sie die spezielle Umgebungsvariable `CLOUDHSM_IGNORE_CKA_MODIFIABLE_FALSE` auf `TRUE`, wenn Sie einen neuen Masterschlüssel erstellen. Diese Umgebungsvariable wird nur für die Generierung des Masterschlüssels benötigt. Sie müssen diese Umgebungsvariable für nichts anderes verwenden. Sie würden diese Variable beispielsweise für den ersten Masterschlüssel verwenden, den Sie erstellen, und würden dann diese Umgebungsvariable nur erneut verwenden, wenn Sie Ihre Masterschlüssel-Edition rotieren möchten. Weitere Informationen finden Sie unter [Generieren des Oracle TDE-Master-Verschlüsselungsschlüssels](#).
- **Behebungsstatus:** Wir verbessern die HSM-Firmware, um das Attribut `CKA_MODIFIABLE` vollständig zu unterstützen. Updates werden im AWS CloudHSM Forum und auf der Seite mit dem Versionsverlauf angekündigt

Bekannte Probleme bei der AWS CloudHSM Cluster-Modifikation

Die folgenden Probleme betreffen Kunden, die versuchen, die Modify-Cluster-API zu verwenden, um den HSM-Typ eines Clusters zu ändern.

Themen

- [Problem: Die Anmeldelatenz nimmt aufgrund der Anzahl der Iterationen zu PBKDF2](#)
- [Problem: Der HSM-Typ konnte aufgrund der Erstellung des Token-Schlüssels nicht geändert werden](#)

Problem: Die Anmeldelatenz nimmt aufgrund der Anzahl der Iterationen zu PBKDF2

- **Auswirkung:** Bei Clustern mit einer großen Anzahl von Benutzern wird es zu einer längeren Migrationsphase kommen. Dies ist auf Änderungen im Backup-Wiederherstellungsprozess zurückzuführen, bei dem PBKDF2 Operationen pro Benutzer ausgeführt werden, wenn ein hsm1.medium-Backup zum ersten Mal auf hsm2m.medium wiederhergestellt wird.
- **Status der Lösung:** Wenn Sie vor dem 20. Dezember 2025 eine neue hsm2m.medium-Instanz erstellt oder von hsm1.medium zu hsm2m.medium migriert haben, müssen Sie Ihr Passwort zurücksetzen, um die Leistungsverbesserungen nutzen zu können, die wir für Anmeldevorgänge implementiert haben. [Anweisungen finden Sie unter Passwort ändern.](#)

Problem: Der HSM-Typ konnte aufgrund der Erstellung des Token-Schlüssels nicht geändert werden

- **Auswirkung:** Kunden, die auf Tokenschlüsseln basierende Workloads ausführen, können ihre Migration nicht starten. Dies liegt daran, dass das HSM in einen eingeschränkten Schreibmodus versetzt wird, um Datenverlustszenarien während der Änderung des HSM-Typs zu verhindern.
- **Umgehung:** Beenden Sie das Erstellen und Löschen von Token-Schlüsseln und warten Sie dann 7 Tage. Alternativ können Sie sich auch an den Support wenden, wenn
 - Kann das Blockieren von Tokenschlüssel-Migrationen nicht handhaben und kann keine Bereitstellung durchführen. blue/green
 - Kann das Blockieren von Token-Schlüsselvorgängen für die Dauer der Migration bewältigen, kann aber nicht die gesamten 7 Tage warten.
- **Lösungsstatus:** Dieses Problem wurde behoben. Kunden, die auf Tokenschlüsseln basierende Workloads ausführen, können jetzt mit der Migration beginnen. Das Erstellen und Löschen von Token-Schlüsseln wird für die Dauer der Migration blockiert.

Bekannte Probleme aufgrund von Betriebsfehlern bei Verwendung der AWS CloudHSM Client-Version 5.12.0 auf hsm2.medium

Die folgenden Probleme wirken AWS CloudHSM sich auf die Verwendung der Client-Version 5.12.0 aus AWS CloudHSM

Problem: Fehler beim Get-Attribute-Vorgang

Wenn Sie von hsm1.medium zu hsm2m.medium migrieren und das CloudHSM Client SDK 5.12.0 verwenden, können Fehler im Zusammenhang mit der Attributbehandlung auftreten.

In den Client-Protokollen wird möglicherweise die folgende Fehlermeldung angezeigt: `Error in deserialization of data: Invalid integer conversion`

Auswirkung: Die folgenden Operationen schlagen bei Verwendung der Client-Version 5.12.0 fehl

- Im PKCS #11 -SDK schlagen Aufrufe von `C_` fehl `GetAttributeValue`
- In der CloudHSM-CLI zeigt der Befehl `key list` keine Attribute in der Ausgabe an
- In der CloudHSM-CLI schlägt die Schlüsselgenerierungsdatei für Schlüssel, die mit hsm1.medium generiert wurden, möglicherweise fehl

Lösung: Wir empfehlen ein Upgrade auf die neueste Version des SDK, mit der dieses Problem behoben wird.

AWS CloudHSM Fehler bei der Schlüsselsynchronisierung des Client-SDK 3

Wenn die clientseitige Synchronisation fehlschlägt, versucht Client SDK 3 nach besten Kräften, AWS CloudHSM alle unerwünschten Schlüssel zu bereinigen, die möglicherweise erstellt wurden (und jetzt unerwünscht sind). Bei diesem Vorgang wird unerwünschtes Schlüsselmaterial sofort entfernt oder unerwünschtes Material für ein späteres Entfernen markiert. In beiden Fällen sind für die Lösung keine Maßnahmen Ihrerseits erforderlich. In den seltenen Fällen, in denen unerwünschtes Schlüsselmaterial AWS CloudHSM nicht entfernt oder markiert werden kann, müssen Sie das Schlüsselmaterial löschen.

Problem: Sie versuchen, einen Token-Schlüssel zu generieren, zu importieren oder zu entpacken, und es werden Fehler angezeigt, die darauf hindeuten, dass Tombstone nicht korrekt ausgeführt wurde.

```
2018-12-24T18:28:54Z liquidSecurity ERR: print_node_ts_status:
[create_object_min_nodes]Key: 264617 failed to tombstone on node:1
```

Ursache: AWS CloudHSM Das Entfernen und Markieren von unerwünschtem Schlüsselmaterial war nicht erfolgreich.

Lösung: Ein HSM in Ihrem Cluster enthält unerwünschtes Schlüsselmateriale, das nicht als unerwünscht markiert ist. Sie müssen das Schlüsselmateriale manuell entfernen. Um unerwünschtes Schlüsselmateriale manuell zu löschen, verwenden Sie `key_mgmt_util` (KMU) oder eine API aus der PKCS #11-Bibliothek oder dem JCE-Anbieter. Weitere Informationen hierzu finden Sie unter [deleteKey](#) oder [Kunde SDKs](#).

Um Tokenschlüssel langlebiger zu machen, AWS CloudHSM schlägt Schlüsselerstellungsvorgänge fehl, die bei der Mindestanzahl der in den clientseitigen HSMs Synchronisierungseinstellungen angegebenen Mindestanzahl nicht erfolgreich sind. Weitere Informationen finden Sie unter [Schlüsselsynchronisierung in AWS CloudHSM](#).

AWS CloudHSM Client SDK 3 verifiziert die HSM-Leistung mit dem pkpspeed-Tool

In diesem Thema wird beschrieben, wie Sie die Leistung des AWS CloudHSM Hardware-Sicherheitsmoduls (HSM) mit dem Client SDK 3 überprüfen können.

Um die Leistung von HSMs in Ihrem AWS CloudHSM Cluster zu überprüfen, können Sie das Tool `pkpspeed` (Linux) oder `pkpspeed_blocking` (Windows) verwenden, das im Client SDK 3 enthalten ist. Das `pkpspeed`-Tool wird unter idealen Bedingungen ausgeführt und ruft das HSM direkt auf, um Operationen auszuführen, ohne ein SDK wie z. B. PKCS11. Wir empfehlen, Ihre Anwendung unabhängig voneinander unter Last zu testen, um Ihre Skalierungsanforderungen zu ermitteln. Wir raten davon ab, die folgenden Tests durchzuführen: Random (I), ModExp (R) und EC point mul (Y).

Weitere Informationen zum Installieren des Clients auf einer Linux EC2 Instance finden Sie unter [Installieren und konfigurieren Sie den AWS CloudHSM Client für CMU \(Linux\)](#). Weitere Informationen zum Installieren des Clients auf einer Windows-Instance finden Sie unter [Installieren und konfigurieren Sie den AWS CloudHSM Client für CMU \(Windows\)](#).

Nachdem Sie den AWS CloudHSM Client installiert und konfiguriert haben, führen Sie den folgenden Befehl aus, um ihn zu starten.

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

- Für Windows-Client 1.1.2 und höher:

```
C:\Program Files\Amazon\CloudHSM>net.exe start AWSCloudHSMClient
```

- Für Windows-Clients 1.1.1 und früher:

```
C:\Program Files\Amazon\CloudHSM>start "cloudhsm_client" cloudhsm_client.exe C:  
\ProgramData\Amazon\CloudHSM\data\cloudhsm_client.cfg
```

Wenn Sie die Client-Software bereits installiert haben, müssen Sie möglicherweise die neueste Version herunterladen und installieren, um „pkpspeed“ abzurufen. Sie finden das Tool pkpspeed

unter Linux in `/opt/cloudhsm/bin/pkpspeed` und unter Windows in `C:\Program Files\Amazon\CloudHSM\`.

Um `pkpspeed` zu verwenden, führen Sie den Befehl `pkpspeed` oder `pkpspeed_blocking.exe` aus. Geben Sie dabei den Benutzernamen und das Passwort eines Crypto-Benutzers (CU) im HSM an. Legen Sie dann die Optionen unter Beachtung der folgenden Empfehlungen fest.

Empfehlungen testen

- Wählen Sie zum Testen der Leistung von RSA-Signatur- und Prüfoperationen die Verschlüsselung `RSA_CRT` unter Linux oder Option B unter Windows. Wählen Sie nicht RSA (Option A unter Windows). Die Verschlüsselungen sind zwar gleichwertig, `RSA_CRT` ist jedoch leistungsoptimiert.
- Beginnen Sie mit einer geringen Anzahl von Threads. Zum Testen der AES-Leistung genügt in der Regel ein Thread, um eine maximale Leistung zu demonstrieren. Zum Testen der RSA-Leistung (`RSA_CRT`) sind drei oder vier Threads in der Regel ausreichend.

Konfigurierbare Optionen für das Tool `pkpspeed`

- FIPS-Modus: AWS CloudHSM befindet sich immer im FIPS-Modus ([AWS CloudHSM FAQs](#) Einzelheiten finden Sie unter). Dies kann überprüft werden, indem Sie die im AWS CloudHSM Benutzerhandbuch dokumentierten CLI-Tools verwenden und den [Rufen Sie Hardwareinformationen für jedes HSM in einem AWS CloudHSM Cluster mit CMU ab](#) Befehl ausführen, der den Status des FIPS-Modus anzeigt.
- Testtyp (blockierend versus nicht blockierend): Dies gibt an, wie Operationen in Thread-Form ausgeführt werden. Sie werden höchstwahrscheinlich bessere Zahlen erhalten, wenn Sie nicht blockierend verwenden. Das liegt daran, dass sie Threads und Parallelität nutzen.
- Anzahl der Threads: Anzahl der Threads, mit denen der Test ausgeführt werden soll.
- Zeit in Sekunden für die Ausführung des Tests (max. = 600): `pkpspeed` erzeugt Ergebnisse, die in „Operationen/Sekunde“ gemessen werden, und meldet diesen Wert für jede Sekunde, in der der Test ausgeführt wird. Wenn der Test beispielsweise 5 Sekunden lang ausgeführt wird, kann die Ausgabe wie die folgenden Beispielwerte aussehen:
 - OPERATIONS/second 821/1
 - OPERATIONS/second 833/1
 - OPERATIONS/second 845/1
 - OPERATIONS/second 835/1

- OPERATIONS/second 837/1

Tests, die mit dem Tool pkpspeed ausgeführt werden können

- AES GCM: Testet die Verschlüsselung im AES-GCM-Modus.
- Basic 3DES CBC: Testet die Verschlüsselung im 3DES CBC-Modus. Eine bevorstehende Änderung finden Sie im Hinweis [1](#) unten.
- Basic AES: Testet die CBC/ECB AES-Verschlüsselung.
- Digest: Testet den Hash-Digest.
- ECDSA-Zeichen: Testet das ECDSA-Zeichen.
- ECDSA-Verifizierung: Testet ECDSA-Verifizierung.
- FIPS Random: Testet die Generierung einer FIPS-konformen Zufallszahl (Hinweis: Diese kann nur im Sperrmodus verwendet werden).
- HMAC: Testet HMAC.
- Random: Dieser Test ist nicht relevant, da wir FIPS 140-2 HSMs verwenden.
- RSA non-CRT versus RSA_CRT: Testet RSA-Signier- und Verifizierungsvorgänge.
- RSA OAEP Enc: Testet die RSA OAEP-Verschlüsselung.
- RSA OAEP Dec: Testet die RSA OAEP-Entschlüsselung.
- RSA Private Dec Non-CRT: Testet die Verschlüsselung mit privatem RSA-Schlüssel (nicht optimiert).
- RSA Private Key dec CRT: Testet die Verschlüsselung mit privatem RSA-Schlüssel (optimiert).
- RSA PSS Sign: Testet das RSA PSS-Zeichen.
- RSA PSS Verify: Teste RSA PSS verifizieren.
- RSA Public Key Enc: Testet die Verschlüsselung mit öffentlichem RSA-Schlüssel.

Bei der Verschlüsselung mit öffentlichem RSA-Schlüssel, der Entschlüsselung mit privatem RSA-Schlüssel (nicht CRT) und der Entschlüsselung mit privatem RSA-Schlüssel (CRT) wird der Benutzer außerdem aufgefordert, die folgenden Fragen zu beantworten:

Do you want to use static key [y/n]

Wenn y eingegeben wird, wird ein vorberechneter Schlüssel in das HSM importiert.

Wenn n eingegeben wird, wird ein neuer Schlüssel generiert.

[1] Gemäß den NIST-Richtlinien ist dies für Cluster im FIPS-Modus nach 2023 nicht zulässig. Für Cluster im Nicht-FIPS-Modus ist dies auch nach 2023 zulässig. Details dazu finden Sie unter [FIPS-140-Konformität: Mechanismus 2024 nicht mehr unterstützt](#).

Beispiele

Die folgenden Beispiele zeigen die Optionen, die Sie mit pkpspeed (Linux) oder pkpspeed_blocking (Windows) zum Testen der HSM-Leistung für RSA- und AES-Operationen auswählen können.

Example– Verwenden von pkpspeed zum Testen der RSA-Leistung

Sie können dieses Beispiel unter Windows, Linux und kompatiblen Betriebssystemen ausführen.

Linux

Verwenden Sie diese Anweisungen für Linux und kompatible Betriebssysteme.

```
/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– Verwenden von pkpspeed zum Testen der AES-Leistung

Linux

Verwenden Sie diese Anweisungen für Linux und kompatible Betriebssysteme.

```
/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 Der Client SDK 5-Benutzer enthält inkonsistente Werte

Der `user list` Befehl in AWS CloudHSM Client SDK 5 gibt eine Liste aller Benutzer und Benutzereigenschaften in Ihrem Cluster zurück. Wenn eine der Eigenschaften eines Benutzers den Wert „inconsistent“ hat, wird dieser Benutzer nicht in Ihrem gesamten Cluster synchronisiert. Das bedeutet, dass der Benutzer mit unterschiedlichen Eigenschaften auf verschiedenen HSMs Clustern existiert. Je nachdem, welche Eigenschaft inkonsistent ist, können unterschiedliche Reparaturschritte unternommen werden.

Die folgende Tabelle enthält Schritte zum Beheben von Inkonsistenzen für einen einzelnen Benutzer. Wenn ein einzelner Benutzer mehrere Inkonsistenzen aufweist, lösen Sie diese, indem Sie die folgenden Schritte von oben nach unten ausführen. Wenn mehrere Benutzer Inkonsistenzen aufweisen, arbeiten Sie diese Liste für jeden Benutzer durch und beheben Sie die Inkonsistenzen für diesen Benutzer vollständig, bevor Sie mit dem nächsten fortfahren.

Note

Um diese Schritte durchzuführen, sollten Sie idealerweise als Administrator angemeldet sein. Wenn Ihr Administratorkonto nicht konsistent ist, gehen Sie wie folgt vor, melden Sie sich beim Administrator an und wiederholen Sie die Schritte, bis alle Eigenschaften

konsistent sind. Sobald Ihr Administratorkonto konsistent ist, können Sie diesen Administrator verwenden, um andere Benutzer im Cluster zu synchronisieren.

Inkonsistentes Merkmal	Beispielausgabe einer Benutzerliste	Auswirkung	Wiederherstellungsmethode
Die „Rolle“ des Benutzers ist „inkonsistent“	<pre>{ "username": "test_user", "role": "inconsistent ", "locked": "false", "mfa": [], "cluster-coverage": "full" }</pre>	Dieser Benutzer ist CryptoUser bei einigen ein HSMs und bei anderen ein Admin HSMs. Dies kann passieren, wenn zwei Personen gleichzeitig SDKs versuchen, denselben Benutzer mit unterschiedlichen Rollen zu erstellen. Sie müssen diesen Benutzer entfernen und ihn mit der gewünschten Rolle neu erstellen.	1. Melden Sie sich als Administrator an. 2. Löschen Sie den Benutzer auf allen: HSMs <pre>user delete --username <user's name> -- role admin user delete --username <user's name> -- role crypto-user</pre> 3. Erstellen Sie den Benutzer mit der gewünschten Rolle: <pre>user create --username <user's name> --role <desired role></pre>
Der Benutzer „Cluster-Abdeckung“ ist „inkonsistent“	<pre>{ "username": "test_user",</pre>	Dieser Benutzer ist in einer Teilmenge von HSMs im Cluster vorhanden. Dies kann	Wenn der Benutzer nicht existieren sollte, gehen Sie wie folgt vor:

Inkonsistentes Merkmal	Beispielausgabe einer Benutzerliste	Auswirkung	Wiederherstellungsmethode
	<pre>"role": "crypto-user", "locked": "false", "mfa": [], "cluster-coverage": "inconsistent "</pre>	passieren, wenn ein Vorgang <code>user create</code> teilweise erfolgreich war, oder wenn ein Benutzer teilweise erfolgreich war. <code>user delete</code> Sie müssen Ihren vorherigen Vorgang beenden, indem Sie diesen Benutzer entweder erstellen oder aus Ihrem Cluster entfernen.	1. Melden Sie sich als Administrator an. 2. Führen Sie diesen Befehl aus: <pre>user delete -- username<user's name> --role admin</pre> 3. Führen Sie den Befehl jetzt aus: <pre>user delete -- username<user's name> --role crypto-user</pre> Wenn der Benutzer existieren sollte, gehen Sie wie folgt vor: 1. Melden Sie sich als Administrator an. 2. Führen Sie den folgenden Befehl aus: <pre>user create --username <user's name> --role <desired role></pre>

Inkonsistentes Merkmal	Beispielausgabe einer Benutzerliste	Auswirkung	Wiederherstellungsmethode
Der Parameter „Gesperrt“ des Benutzers ist „inkonsistent“ oder „wahr“	<pre>{ "username": "test_user", "role": "crypto-user", "locked" : inconsistent , "mfa": [], "cluster-coverage": "full" }</pre>	Dieser Benutzer ist für eine Teilmenge von gesperrt. HSMs Dies kann passieren , wenn ein Benutzer das falsche Passwort verwendet und sich nur mit einer Teilmenge von HSMs im Cluster verbindet. Sie müssen die Anmeldeinformationen des Benutzers ändern, damit sie im gesamten Cluster einheitlich sind.	Wenn der Benutzer MFA aktiviert hat, gehen Sie wie folgt vor: 1. Melden Sie sich als Administrator an. 2. Verwenden Sie den folgenden Befehl, um MFA vorübergehend zu deaktivieren: <pre>user change-mfa token-sig n --username <user's name> --role <desired role> --disable</pre> 3. Ändern Sie das Passwort des Benutzers, sodass er sich bei allen HSMs anmelden kann: <pre>user change-password --username e <user's name> --role <desired role></pre> Wenn MFA für den Benutzer aktiv sein

Inkonsistentes Merkmal	Beispielausgabe einer Benutzerliste	Auswirkung	Wiederherstellungsmethode
			soll, befolgen Sie diese Schritte: 1. Lassen Sie den Benutzer sich anmelden und MFA erneut aktivieren (dazu muss er Tokens signieren und seinen öffentlichen Schlüssel in einer PEM-Datei angeben): <pre>user change-mfa token-sign --username <user's name> --role <desired role> --token <File></pre>

Inkonsistentes Merkmal	Beispielausgabe einer Benutzerliste	Auswirkung	Wiederherstellungsmethode
Der MFA-Status ist „inkonsistent“	<pre>{ "username": "test_user", "role": "crypto-u ser", "locked": "false", "mfa": [{ "strategy": "token-sign", "status": "inconsistent " }], "cluster- coverage": "full" }</pre>	Dieser Benutzer hat unterschiedliche MFA-Flags auf HSMs verschiedenen Clustern. Dies kann passieren , wenn ein MFA-Vorgang nur für eine Teilmenge von abgeschlossen wurde. HSMs Sie müssen das Passwort des Benutzers zurücksetzen und ihm erlauben, MFA erneut zu aktivieren.	Wenn der Benutzer MFA aktiviert hat, gehen Sie wie folgt vor: 1. Melden Sie sich als Administrator an. 2. Verwenden Sie den folgenden Befehl, um MFA vorübergehend zu deaktivieren: <pre>user change- mfa token-sig n --username <user's name> --role <desired role> --disable</pre> 3. Sie müssen dann auch das Passwort des Benutzers ändern, damit er sich bei allen anmelden kann: HSMs <pre>user change-pa ssword --usernam e <user's name> --role <desired role></pre>

Inkonsistentes Merkmal	Beispielausgabe einer Benutzerliste	Auswirkung	Wiederherstellungsmethode
			Wenn MFA für den Benutzer aktiv sein soll, befolgen Sie diese Schritte: 1. Lassen Sie den Benutzer sich anmelden und MFA erneut aktivieren (dazu muss er Tokens signieren und seinen öffentlichen Schlüssel in einer PEM-Datei angeben): <pre>user change- mfa token-sign --username <user's name> --role <desired role> --token <File></pre>

AWS CloudHSM Fehler bei der Replikation von Client-SDK für 5 Benutzer

Der `user replicate` Befehl in der CloudHSM-CLI repliziert einen Benutzer zwischen AWS geklonten CloudHSM-Clustern. In diesem Handbuch werden Fehler behandelt, die auf Benutzerinkonsistenzen innerhalb des Quell-Clusters oder zwischen Quell- und Zielclustern zurückzuführen sind. Das Benutzerreplikat überprüft anhand der folgenden Attribute, ob Benutzer konsistent sind:

- Rolle des Benutzers

- Status der Kontosperrung
- Quorum-Status
- Status der Multi-Factor Authentication (MFA)

Problem: Der ausgewählte Benutzer ist nicht im gesamten Cluster synchronisiert

Der Benutzerreplikationsprozess überprüft die Benutzersynchronisierung im gesamten Quellcluster. Wenn das Attribut eines Benutzers den Wert „inconsistent“ hat, bedeutet dies, dass der Benutzer nicht im gesamten Cluster synchronisiert ist. Die Benutzerreplikation schlägt mit der folgenden Fehlermeldung fehl:

```
{
  "error_code": 1,
  "data": "Specified user is inconsistent across the cluster"
}
```

So überprüfen Sie die Desynchronisierung der Benutzer im Quellcluster:

- Führen Sie den `user list` Befehl in der CloudHSM-CLI aus.

```
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"
  }
]
}
```

Lösung: Synchronisieren Sie Benutzerattribute im gesamten Quellcluster

- Informationen zum Synchronisieren von Benutzerinformationen im gesamten Quellcluster finden Sie in den folgenden Abschnitten: [AWS CloudHSM Der Client SDK 5-Benutzer enthält inkonsistente Werte](#).

Problem: Auf dem Zielcluster ist ein Benutzer mit unterschiedlichen Attributen vorhanden

Wenn ein Benutzer mit derselben Referenz bereits in einem oder mehreren HSMs im Zielcluster vorhanden ist, aber unterschiedliche Benutzerattribute hat, kann der folgende Fehler auftreten:

```
{
  "error_code": 1,
  "data": "User replicate failed on 1 of 3 connections"
}
```

Auflösung

1. Ermitteln Sie, welche Version des Benutzers beibehalten werden soll.
2. Löschen Sie den unerwünschten Benutzer im entsprechenden Cluster, indem Sie den `user delete` Befehl ausführen. Weitere Informationen finden Sie unter [Löschen Sie einen AWS CloudHSM Benutzer mit CloudHSM CLI](#).

3. Replizieren Sie den Benutzer, indem Sie den `user replicate` Befehl ausführen.

Problem: Die Benutzerreplikation von hsm2m.medium nach hsm1.medium schlägt fehl

Die Benutzerreplikation von hsm2m.medium auf hsm1.medium wird nicht unterstützt. Wenn Sie einen Benutzer von einem hsm2m.medium-Quellcluster auf einen hsm1.medium-Zielcluster replizieren, tritt der folgende Fehler auf:

```
{
  "error_code": 1,
  "data": "User replicate failed on 1 of 1 connections"
}
```

Auflösung

- Verwenden Sie die [Benutzerverwaltung](#) mit CloudHSM CLI, um die fehlenden Benutzer manuell neu zu erstellen.

AWS CloudHSM Fehler bei der Schlüsselreplikation des Client-SDK 5

Der `key replicate` Befehl in der CloudHSM-CLI repliziert einen Schlüssel von einem AWS CloudHSM Quellcluster zu einem Zielcluster. AWS CloudHSM In diesem Handbuch werden Fehler behandelt, die durch Inkonsistenzen innerhalb des Quell-Clusters oder zwischen dem Quell- und dem Ziel-Cluster verursacht werden.

Problem: Der ausgewählte Schlüssel ist nicht im gesamten Cluster synchronisiert

Der Schlüsselreplikationsprozess überprüft die Schlüsselsynchronisierung im gesamten Quellcluster. Wenn Schlüsselinformationen oder Attribute den Wert „inkonsistent“ haben, bedeutet dies, dass der Schlüssel nicht im gesamten Cluster synchronisiert ist. Die Schlüsselreplikation schlägt mit der folgenden Fehlermeldung fehl:

```
{
```

```

    "error_code": 1,
    "data": "The selected key is not synchronized throughout the cluster"
  }

```

So überprüfen Sie, ob die Schlüssel im Quellcluster desynchronisiert wurden:

1. Führen Sie den `key list` Befehl in der CloudHSM-CLI aus.
2. Verwenden Sie das `--filter` Flag, um den Schlüssel anzugeben.
3. Fügen Sie das `--verbose` Kennzeichen hinzu, um die vollständige Ausgabe mit wichtigen Informationen zur Abdeckung zu sehen.

```

aws-cloudhsm > key list --filter attr.label=example-desynchronized-key-label --verbose
{
  "error_code": 0,
  "data": {
    "matched_keys": [
      {
        "key-reference": "0x00000000000048000f",
        "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
}
}

```

Lösung: Synchronisieren Sie wichtige Informationen und Attribute im gesamten Quellcluster

So synchronisieren Sie Schlüsselinformationen und Attribute im gesamten Quellcluster:

1. Bei inkonsistenten Schlüsselattributen: Verwenden Sie den `key set-attribute` Befehl, um das gewünschte Attribut für den spezifischen Schlüssel festzulegen.
2. Bei inkonsistenter Abdeckung gemeinsam genutzter Benutzer: Passen Sie mit den `key unshare` Befehlen `key share` oder die Tastenbelegung mit den gewünschten Benutzern an.

Problem: Ein Schlüssel mit derselben Referenz ist im Zielcluster mit unterschiedlichen Informationen oder Attributen vorhanden

Wenn ein Schlüssel mit derselben Referenz im Zielcluster vorhanden ist, aber unterschiedliche Informationen oder Attribute hat, kann der folgende Fehler auftreten:

```
{
```



```
"error_code": 1,  
"data": "Key replicate failed on 1 of 3 connections"  
}
```

Auflösung

1. Ermitteln Sie, welche Version des Schlüssels beibehalten werden soll.
2. Löschen Sie die unerwünschte Schlüsselversion mithilfe des `key delete` Befehls im entsprechenden Cluster.
3. Replizieren Sie den Schlüssel aus dem Cluster mit der richtigen Version.

AWS CloudHSM Bei der Überprüfung der Schlüsselvefügbarkeit ist ein Fehler aufgetreten

Problem: Ein AWS CloudHSM Hardware-Sicherheitsmodul (HSM) gibt den folgenden Fehler zurück:

```
Key <KEY HANDLE> does not meet the availability requirements - The key must be  
available on at least 2 HSMs before being used.
```

Ursache: Bei der Überprüfung der Verfügbarkeit von Schlüsseln wird nach Schlüsseln gesucht, die in seltenen, aber möglichen Fällen verloren gehen könnten. Dieser Fehler tritt normalerweise in Clustern mit nur einem HSM oder in Clustern mit zwei HSMs während eines Zeitraums auf, in dem eines von ihnen ersetzt wird. In diesen Situationen haben die folgenden Kundenvorgänge wahrscheinlich zu dem oben genannten Fehler geführt:

- Ein neuer Schlüssel wurde mit einem Befehl wie [Die Kategorie „Symmetrisch generieren“ in CloudHSM CLI](#) oder [Die generate-asymmetric-pair Kategorie in CloudHSM CLI](#) generiert.
- Eine [Schlüssel für einen Benutzer mit CloudHSM CLI auflisten](#)-Operation wurde gestartet.
- Eine neue Instanz des SDK wurde gestartet.

Note

OpenSSL forkt häufig neue Instanzen des SDK.

Lösung/Empfehlung: Wählen Sie aus den folgenden Aktionen, um das Auftreten dieses Fehlers zu verhindern:

- Verwenden Sie den `--disable-key-availability-check`-Parameter, um die Schlüsselverfügbarkeit in der Konfigurationsdatei Ihres [Configure Tools](#) auf „false“ zu setzen. Weitere Informationen finden Sie im Abschnitt [AWS CloudHSM Konfigurationsparameter für das Client-SDK 5](#) des Configure Tools.
- Wenn Sie einen Cluster mit zwei verwenden HSMs, vermeiden Sie es, die Operationen zu verwenden, die zu dem Fehler geführt haben, außer während des Initialisierungscode.
- Erhöhen Sie die Anzahl von HSMs in Ihrem Cluster auf mindestens drei.

AWS CloudHSM Extrahieren von Schlüsseln mit JCE

Verwenden Sie die folgenden Abschnitte, um Probleme beim Extrahieren von AWS CloudHSM Schlüsseln mit JCE zu beheben.

`getEncoded` oder `getPrivateExponent` GetS gibt null zurück

`getEncoded`, `getPrivateExponent` und `getS` geben Null zurück, da sie standardmäßig deaktiviert sind. Informationen zum Aktivieren finden Sie unter [Schlüsselextraktion mit JCE für AWS CloudHSM](#).

Wenn `getEncoded`, `getPrivateExponent` und `getS` nach der Aktivierung Null zurückgeben, erfüllt Ihr Schlüssel nicht die richtigen Voraussetzungen. Weitere Informationen finden Sie unter [Schlüsselextraktion mit JCE für AWS CloudHSM](#).

`getEncoded` oder `getS` geben Schlüsselbytes außerhalb des HSM zurück `getPrivateExponent`

Sie oder jemand mit Zugriff auf Ihr System haben die Extraktion von klaren Schlüsseln aktiviert. Auf den folgenden Seiten finden Sie weitere Informationen, unter anderem, wie Sie diese Konfiguration auf den deaktivierten Standardstatus zurücksetzen können.

- [Schlüsselextraktion mit JCE für AWS CloudHSM](#)
- [Schlüssel aus einem HSM schützen und extrahieren](#)

HSM-Drosselung

Wenn Ihre Arbeitslast die Kapazität des Hardware-Sicherheitsmoduls (HSM) Ihres AWS CloudHSM Clusters überschreitet, erhalten Sie Fehlermeldungen, die besagen, dass sie ausgelastet oder

gedrosselt HSMs sind. In diesem Fall kann es zu einem verringerten Durchsatz oder einer erhöhten Anzahl von Ablehnungsanfragen von kommen. HSMs Darüber hinaus werden HSMs möglicherweise die folgenden Auslastungsfehler gesendet.

Für Client-SDK 5

- In PKCS11, ausgelastete Fehler werden zugeordnet auf `CKR_FUNCTION_FAILED`. Dieser Fehler kann aus mehreren Gründen auftreten. Wenn dieser Fehler jedoch durch HSM-Drosselung verursacht wird, werden die folgenden Protokollzeilen in Ihrem Protokoll angezeigt:
 - `[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`
- In JCE werden Busy-Fehler `com.amazonaws.cloudhsm.jce.jni.exception.InternalException: Unexpected error with the Provider: The HSM could not queue the request for processing.` zugeordnet.
- Bei SDKs anderen Auslastungsfehlern wird die folgende Meldung ausgegeben: `Received error response code from Server. Response Code: 187.`

Für Client-SDK 3

- Bei „PKCS11Ausgelastet“ werden `CKR_OPERATION_ACTIVE` Fehler zugeordnet.
- In JCE werden Busy-Fehler `CFM2Exception` mit dem Status `0xBB` (187) zugeordnet. Anwendungen können die `getStatus()`-Funktion auf `CFM2Exception` verwenden, um zu überprüfen, welcher Status vom HSM zurückgegeben wird.
- Bei anderen SDKs Auslastungsfehlern wird die folgende Meldung ausgedruckt: `HSM Error: HSM is already busy generating the keys(or random bytes) for another request.`

Auflösung

Sie können diese Probleme beheben, indem Sie mindestens eine der folgenden Maßnahmen ergreifen:

- Fügen Sie Wiederholungsbefehle für abgelehnte HSM-Operationen in Ihrer Anwendungsebene hinzu. Stellen Sie vor der Aktivierung von Wiederholungsbefehlen sicher, dass Ihr Cluster ausreichend dimensioniert ist, um Spitzenlasten zu bewältigen.

 Note

Für Client-SDK 5.8.0 und höher sind Wiederholungsbefehle standardmäßig aktiviert. Einzelheiten zur Konfiguration der Wiederholungsbefehle der einzelnen SDKs finden Sie unter [Erweiterte Konfigurationen für das Client-SDK-5-Configure-Tool](#).

- Fügen Sie Ihrem Cluster weitere HSMs hinzu, indem Sie den Anweisungen unter folgen [Skalierung HSMs in einem AWS CloudHSM Cluster](#).

 Important

Wir empfehlen, Ihren Cluster einem Belastungstest zu unterziehen, um die zu erwartende Spitzenlast zu ermitteln, und dann ein weiteres HSM hinzuzufügen, um eine hohe Verfügbarkeit zu gewährleisten.

Halten Sie HSM-Benutzer HSMs im gesamten Cluster synchron AWS CloudHSM

Um die [Benutzer Ihres HSM zu verwalten, verwenden Sie ein Befehlszeilentool namens `cloudhsm\_mgmt\_util`](#). AWS CloudHSM Es kommuniziert nur mit denen, die in der Konfigurationsdatei des Tools enthalten sind. HSMs Andere HSMs im Cluster, die nicht in der Konfigurationsdatei enthalten sind, werden nicht erkannt.

AWS CloudHSM synchronisiert die Schlüssel auf Ihrem Computer mit HSMs allen anderen Schlüsseln HSMs im Cluster, synchronisiert jedoch nicht die Benutzer oder Richtlinien des HSM. Wenn Sie `cloudhsm_mgmt_util` zur [Verwaltung von HSM-Benutzern](#) verwenden, wirken sich diese Benutzeränderungen möglicherweise nur auf einige der Cluster aus, d. h. auf diejenigen, die sich in der Konfigurationsdatei `cloudhsm_mgmt_util` befinden. HSMs Dies kann zu Problemen führen, wenn Schlüssel innerhalb des Clusters AWS CloudHSM synchronisiert werden, da die Benutzer, denen die Schlüssel gehören, möglicherweise nicht auf allen HSMs im Cluster vorhanden sind. HSMs

Um diese Probleme zu vermeiden, bearbeiten Sie die `cloudhsm_mgmt_util`-Konfigurationsdatei, bevor Sie Benutzer verwalten. Weitere Informationen finden Sie unter [???](#).

Verbindung zum AWS CloudHSM Cluster verloren

Bei [der Konfiguration des AWS CloudHSM Clients](#) haben Sie die IP-Adresse des ersten HSM in Ihrem Cluster angegeben. Diese IP-Adresse wird in der Konfigurationsdatei für den AWS CloudHSM Client gespeichert. Wenn der Client gestartet wird, wird versucht, eine Verbindung zu dieser IP-Adresse einzurichten. Wenn das nicht möglich ist – weil beispielsweise der HSM fehlgeschlagen ist oder Sie ihn gelöscht haben – wird möglicherweise eine der folgenden Fehlermeldungen angezeigt:

```
LIQUIDSECURITY: Daemon socket connection error
```

```
LIQUIDSECURITY: Invalid Operation
```

Um diese Fehler zu beheben, aktualisieren Sie die Konfigurationsdatei mit der IP-Adresse eines aktiven, verfügbaren HSM im Cluster.

Um die Konfigurationsdatei für den AWS CloudHSM Client zu aktualisieren

1. Verwenden Sie eine der folgenden Methoden, um die IP-Adresse eines aktiven HSM in Ihrem Cluster zu ermitteln.
 - Sehen Sie sich die HSMsRegisterkarte auf der Seite mit den Cluster-Details in der [AWS CloudHSM Konsole](#) an.
 - Verwenden Sie AWS Command Line Interface (AWS CLI), um den [describe-clusters](#) Befehl auszuführen.

Sie benötigen diese IP-Adresse in einem späteren Schritt.

2. Verwenden Sie den folgenden Befehl, um den Client zu beenden.

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

- Für Windows-Client 1.1.2 und höher:

```
C:\Program Files\Amazon\CloudHSM>net.exe stop AWSCloudHSMClient
```

- Für Windows-Clients 1.1.1 und früher:

Verwenden Sie Strg + C in dem Befehlsfenster, in dem Sie den AWS CloudHSM Client gestartet haben.

3. Verwenden Sie den folgenden Befehl ein, um die Konfigurationsdatei des Clients zu aktualisieren, und geben Sie dazu die IP-Adresse an, die Sie in einem vorherigen Schritt erhalten haben.

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

4. Verwenden Sie den folgenden Befehl, um den -Client zu starten.

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

- Für Windows-Client 1.1.2 und höher:

```
C:\Program Files\Amazon\CloudHSM>net.exe start AWSCloudHSMClient
```

- Für Windows-Clients 1.1.1 und früher:

```
C:\Program Files\Amazon\CloudHSM>start "cloudhsm_client" cloudhsm_client.exe  
C:\ProgramData\Amazon\CloudHSM\data\cloudhsm_client.cfg
```

Fehlende AWS CloudHSM Audit-Logs CloudWatch

Wenn Sie vor dem 20. Januar 2018 einen AWS CloudHSM Cluster erstellt haben, müssen Sie eine [dienstverknüpfte Rolle](#) manuell konfigurieren, um die Übermittlung der Audit-Logs dieses Clusters zu ermöglichen. Anweisungen zum Aktivieren einer serviceverknüpften Rolle in einem HSM-Cluster finden Sie unter [Grundlegendes zu serviceverknüpften Rollen](#) und unter [Erstellen einer serviceverknüpften Rolle](#) im IAM-Benutzerleitfaden.

Benutzerdefiniert IVs mit nicht konformer Länge für AES-Schlüsselumbruch AWS CloudHSM

Mithilfe dieses Themas zur Problembehandlung können Sie feststellen, ob Ihre Anwendung unwiederbringliche verpackte Schlüssel generiert. Wenn Sie von diesem Problem betroffen sind, verwenden Sie dieses Thema, um das Problem zu lösen.

Topics

- [Stellen Sie fest, ob Ihr Code unwiederbringliche verpackte Schlüssel generiert](#)
- [Maßnahmen, die Sie ergreifen müssen, wenn Ihr Code unwiederbringliche verpackte Schlüssel generiert](#)

Stellen Sie fest, ob Ihr Code unwiederbringliche verpackte Schlüssel generiert

Sie sind nur betroffen, wenn Sie alle folgenden Bedingungen erfüllen:

Bedingung	Woher soll ich das wissen?
Ihre Anwendung verwendet die PKCS #11-Bibliothek	Die PKCS #11-Bibliothek wird als <code>libpkcs11.so</code> -Datei in Ihrem <code>/opt/cloudhsm/lib</code> -Ordner installiert. In der Sprache C geschriebene Anwendungen verwenden im Allgemeinen direkt die PKCS #11-Bibliothek, während in Java geschriebene Anwendungen die Bibliothek möglicherweise indirekt über eine Java-Abstraktionsschicht verwenden. Wenn Sie Windows verwenden, sind Sie davon NICHT betroffen, da die PKCS #11-Bibliothek derzeit nicht für Windows verfügbar ist.
Ihre Anwendung verwendet speziell Version 3.0.0 der PKCS #11-Bibliothek	Wenn Sie eine E-Mail vom AWS CloudHSM Team erhalten haben, verwenden Sie wahrscheinlich Version 3.0.0 der PKCS #11 -Bibliothek. Verwenden Sie diesen Befehl, um die Softwareversion auf Ihren Anwendungs-Instances zu überprüfen: <pre data-bbox="829 1199 1507 1276">rpm -qa grep ^cloudhsm</pre>
Schlüssel werden mithilfe von AES-Schlüsselumbruch umgebrochen	AES-Schlüsselumbruch bedeutet, dass Sie einen AES-Schlüssel verwenden, um einen anderen Schlüssel zu umschließen. Der Name des entsprechenden Mechanismus lautet <code>CKM_AES_KEY_WRAP</code>. Er wird zusammen mit der Funktion <code>C_WrapKey</code> verwendet. Andere AES-basierte Wrapping-Mechanismen, die Initialisierungsvektoren (IVs) verwenden, wie z. B. <code>CKM_AES_GCM</code> und <code>CKM_CLOUDHSM_AES_GCM</code>, sind von diesem Problem

Bedingung	Woher soll ich das wissen?
	nicht betroffen. Erfahren Sie mehr über Funktionen und Mechanismen.
Sie geben eine benutzerdefinierte IV an, wenn Sie AES Key Wrapping aufrufen, und die Länge dieser IV ist kürzer als 8	Der AES-Schlüsselumbruch wird im Allgemeinen mit einer CK_MECHANISM Struktur wie folgt initialisiert: <pre>CK_MECHANISM mech = {CKM_AES_KEY_WRAP, IV_POINTER, IV_LENGTH};</pre> Dieses Problem trifft auf Sie nur in folgenden Fällen zu: • IV_POINTER ist nicht NULL • IV_LENGTH ist weniger als 8 Byte

Wenn Sie nicht alle oben genannten Bedingungen erfüllen, können Sie jetzt mit dem Lesen aufhören. Ihre verpackten Schlüssel können ordnungsgemäß ausgepackt werden, und dieses Problem hat keine Auswirkungen auf Sie. Andernfalls lesen Sie unter [the section called “Maßnahmen, die Sie ergreifen müssen, wenn Ihr Code unwiederbringliche verpackte Schlüssel generiert”](#) weiter.

Maßnahmen, die Sie ergreifen müssen, wenn Ihr Code unwiederbringliche verpackte Schlüssel generiert

Sie sollten die folgenden drei Schritte ausführen:

1. Aktualisieren Sie Ihre PKCS #11-Bibliothek sofort auf eine neuere Version
 - [Aktuelle PKCS #11-Bibliothek für Amazon Linux, CentOS 6 und RHEL 6](#)
 - [Aktuelle PKCS #11-Bibliothek für Amazon Linux 2, CentOS 7 und RHEL 7](#)
 - [Aktuelle PKCS #11-Bibliothek für Ubuntu 16.04 LTS](#)
2. Aktualisieren Sie Ihre Software, um eine standardkonforme IV zu verwenden

Wir empfehlen Ihnen dringend, unserem Beispielcode zu folgen und einfach eine NULL IV anzugeben, sodass das HSM die standardkonforme Standard-IV verwendet. Alternativ können Sie

die IV explizit als 0xA6A6A6A6A6A6A6A6 mit einer entsprechenden IV-Länge von 8 angeben. Es wird nicht empfohlen, andere IV für das Umschließen von AES-Schlüsseln zu verwenden, und wir werden in einer future Version der PKCS #11 -Bibliothek das benutzerdefinierte IVs Umschließen von AES-Schlüsseln ausdrücklich deaktivieren.

Der Beispielcode für die korrekte Angabe der IV befindet sich in [aes_wrapping.c](#) unter. GitHub

3. Identifizieren Sie vorhandene verpackte Schlüssel und stellen Sie sie wieder her

Sie sollten alle Schlüssel identifizieren, die Sie mit Version 3.0.0 der PKCS #11 -Bibliothek verpackt haben, und sich dann an den Support wenden, um Unterstützung (<https://aws.amazon.com/support>) bei der Wiederherstellung dieser Schlüssel zu erhalten.

Important

Dieses Problem betrifft nur Schlüssel, die mit Version 3.0.0 der PKCS #11-Bibliothek umschlossen wurden. Sie können Schlüssel mithilfe früherer Versionen (2.0.4 und Pakete mit niedrigeren Nummern) oder neueren Versionen (3.0.1 und Pakete mit höheren Nummern) der PKCS #11-Bibliothek umbrechen.

Behebung von Fehlern bei der AWS CloudHSM Clustererstellung

Wenn Sie einen Cluster erstellen, AWS CloudHSM wird die dienstverknüpfte AWSService RoleForCloud HSM-Rolle erstellt, sofern die Rolle noch nicht vorhanden ist. Wenn die dienstverknüpfte Rolle AWS CloudHSM nicht erstellt werden kann, schlägt Ihr Versuch, einen Cluster zu erstellen, möglicherweise fehl.

In diesem Thema wird erklärt, wie Sie die am häufigsten auftretenden Probleme lösen und so erfolgreich einen Cluster erstellen. Sie müssen diese Rolle nur einmal erstellen. Sobald die servicegebundene Rolle in Ihrem Konto erstellt wurde, können Sie mit jeder der unterstützten Methoden weitere Cluster erstellen und verwalten.

Die folgenden Abschnitte halten Ratschläge bereit, mit der Fehler bei der Cluster-Erstellung behoben werden können, falls diese im Zusammenhang mit der servicegebundenen Rolle stehen. Wenn Sie nach Ausführung dieser Ratschläge immer noch nicht in der Lage sind, Cluster zu erstellen, wenden Sie sich an den [Support](#). Weitere Informationen zur dienstbezogenen AWSService RoleForCloud HSM-Rolle finden Sie unter. [Mit Diensten verknüpfte Rollen für AWS CloudHSM](#)

Themen

- [Fügen Sie die fehlende Berechtigung hinzu](#)
- [Manuelles Erstellen der serviceverknüpften Rolle](#)
- [Verwenden Sie einen Benutzer, der nicht an einen Verbund angeschlossen ist](#)

Fügen Sie die fehlende Berechtigung hinzu

Um eine servicegebundene Rolle erstellen zu können, muss der Benutzer über die `iam:CreateServiceLinkedRole`-Berechtigung verfügen. Wenn der IAM-Benutzer, der den Cluster erstellt, nicht über diese Berechtigung verfügt, schlägt der Clustererstellungsprozess fehl, wenn versucht wird, die dienstverknüpfte Rolle in Ihrem Konto zu erstellen. AWS

Wenn eine fehlende Berechtigung den Fehler verursacht, enthält die Fehlermeldung den folgenden Text.

```
This operation requires that the caller have permission to call
iam:CreateServiceLinkedRole to create the CloudHSM Service Linked Role.
```

Um diesen Fehler zu beheben, geben Sie dem IAM-Benutzer, der den Cluster erstellt, die `AdministratorAccess`-Berechtigung oder fügen Sie der IAM-Richtlinie des Benutzers die `iam:CreateServiceLinkedRole`-Berechtigung hinzu. Weitere Anleitungen finden Sie unter [Hinzufügen von Berechtigungen zu einem neuen oder vorhandenen Benutzer \(Konsole\)](#).

Versuchen Sie anschließend erneut, [den Cluster zu erstellen](#).

Manuelles Erstellen der serviceverknüpften Rolle

Sie können die IAM-Konsole, CLI oder API verwenden, um die dienstverknüpfte `AWSServiceRoleForCloudHSM`-Rolle zu erstellen. Weitere Informationen finden Sie unter [Erstellen einer serviceverknüpfte Rolle](#) im IAM-Leitfaden.

Verwenden Sie einen Benutzer, der nicht an einen Verbund angeschlossen ist

Verbundbenutzer, deren Anmeldeinformationen außerhalb von stammen AWS, können viele der Aufgaben eines Benutzers ohne Verbundzugriff ausführen. Mit AWS können Benutzer jedoch keine

API-Aufrufe zum Erstellen einer servicegebundenen Rolle von einem verbundenen Endpunkt aus ausführen.

Um dieses Problem zu lösen, [erstellen Sie einen nicht verbundenen Benutzer](#) mit der `iam:CreateServiceLinkedRole`-Berechtigung oder erteilen Sie einem vorhandenen nicht verbundenen Benutzer die `iam:CreateServiceLinkedRole`-Berechtigung. Lassen Sie den Benutzer anschließend in der AWS CLI [einen Cluster erstellen](#). Dadurch wird die servicegebundene Rolle in Ihrem Konto erstellt.

Sobald die serviceverknüpfte Rolle erstellt wurde, können Sie auf Wunsch den Cluster löschen, den der nicht verbundene Benutzer erstellt hat. Das Löschen des Clusters hat keine Auswirkungen auf die Rolle. Danach kann jeder Benutzer mit den erforderlichen Berechtigungen, einschließlich Verbundbenutzer, AWS CloudHSM Cluster in Ihrem Konto erstellen.

Um zu überprüfen, ob die Rolle erstellt wurde, öffnen Sie die IAM-Konsole unter <https://console.aws.amazon.com/iam/> und wählen Sie Rollen aus. Oder verwenden Sie den [get-role](#)-Befehl in der AWS CLI.

```
$ aws iam get-role --role-name AWSServiceRoleForCloudHSM
{
  "Role": {
    "Description": "Role for CloudHSM service operations",
    role policy statement
    "RoleId": "AR0AJ4I6WN5QVGG5G7CBY",
    "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 Client-Konfigurationsprotokolle abrufen

AWS CloudHSM bietet Tools für Client SDK 3 und Client SDK 5, mit denen Sie Informationen über Ihre Umgebung sammeln können, damit der AWS Support Probleme beheben kann.

Themen

- [AWS CloudHSM Support-Tool für das Client-SDK 5](#)
- [AWS CloudHSM Tool zur Unterstützung von Client-SDK 3](#)

AWS CloudHSM Support-Tool für das Client-SDK 5

Das Skript für AWS CloudHSM Client SDK 5 extrahiert die folgenden Informationen:

- Die Konfigurationsdatei für die Client-SDK 5-Komponente
- Verfügbare Protokolldateien
- Aktuelle Version des Betriebssystems
- Paketinformationen

Das Info-Tool für Client-SDK 5 wird ausgeführt

Das Client-SDK 5 umfasst ein Client-Support-Tool für jede Komponente, aber alle Tools funktionieren gleich. Führen Sie das Tool aus, um eine Ausgabedatei mit allen erfassten Informationen zu erstellen.

Die Tools verwenden eine Syntax wie diese:

```
[ pkcs11 | dyn | jce | ksp | cli ]_info
```

Um beispielsweise Informationen zur Unterstützung von einem Linux-Host zu sammeln, auf dem die PKCS #11-Bibliothek läuft, und das System in das Standardverzeichnis schreiben zu lassen, würden Sie diesen Befehl ausführen:

```
/opt/cloudhsm/bin/pkcs11_info
```

Das Tool erstellt die Ausgabedatei innerhalb des /tmp-Verzeichnisses.

AWS CloudHSM CLI

Um Supportdaten für AWS CloudHSM CLI unter Linux zu sammeln

- Verwenden Sie das Support-Tool, um Daten zu sammeln.

```
/opt/cloudhsm/bin/cli_info
```

Um Supportdaten für Windows AWS CloudHSM zu sammeln

- Verwenden Sie das Support-Tool, um Daten zu sammeln.

```
PS C:\> & "C:\Program Files\Amazon\CloudHSM\bin\cli_info.exe"
```

PKCS #11 library

Um Unterstützungsdaten für die PKCS #11-Bibliothek unter Linux zu sammeln

- Verwenden Sie das Support-Tool, um Daten zu sammeln.

```
/opt/cloudhsm/bin/pkcs11_info
```

Um Unterstützungsdaten für die PKCS #11-Bibliothek unter Windows zu sammeln

- Verwenden Sie das Support-Tool, um Daten zu sammeln.

```
PS C:\> & "C:\Program Files\Amazon\CloudHSM\bin\pkcs11_info.exe"
```

OpenSSL Dynamic Engine

Um Unterstützungsdaten für OpenSSL Dynamic Engine unter Linux zu sammeln

- Verwenden Sie das Support-Tool, um Daten zu sammeln.

```
/opt/cloudhsm/bin/dyn_info
```

JCE provider

Um Supportdaten für den JCE-Anbieter unter Linux zu sammeln

- Verwenden Sie das Support-Tool, um Daten zu sammeln.

```
/opt/cloudhsm/bin/jce_info
```

Um Supportdaten für den JCE-Anbieter unter Windows zu sammeln

- Verwenden Sie das Support-Tool, um Daten zu sammeln.

```
PS C:\> & "C:\Program Files\Amazon\CloudHSM\bin\jce_info.exe"
```

Key Storage Provider

Um Supportdaten für Key Storage Provider unter Windows zu sammeln

- Verwenden Sie das Support-Tool, um Daten zu sammeln.

```
PS C:\> & "C:\Program Files\Amazon\CloudHSM\bin\ksp_info.exe"
```

Abrufen von Protokollen aus einer Serverless-Umgebung

Für die Konfiguration für serverlose Umgebungen wie Fargate oder Lambda empfehlen wir Ihnen, Ihren AWS CloudHSM Protokolltyp auf zu konfigurieren. term Nach der Konfiguration auf term kann in der serverlosen Umgebung Daten ausgegeben werden. CloudWatch

Informationen zum Abrufen der Client-Logs finden Sie unter [Arbeiten mit Protokollgruppen und Protokollstreams](#) im Amazon CloudWatch Logs-Benutzerhandbuch. CloudWatch

AWS CloudHSM Tool zur Unterstützung von Client-SDK 3

Das Skript für das AWS CloudHSM Client SDK 3 extrahiert die folgenden Informationen:

- Betriebssystem und seine aktuelle Version
- Clientkonfigurationsinformationen aus `cloudhsm_client.cfg`-, `cloudhsm_mgmt_util.cfg`-, und `application.cfg`-Dateien
- Clientprotokolle von dem für die Plattform spezifischen Speicherort
- Cluster- und HSM-Informationen mithilfe von `cloudhsm_mgmt_util`
- OpenSSL-Informationen
- Aktuelle Client- und Build-Version
- Installationsprogrammversion

Das Info-Tool für Client-SDK 3 wird ausgeführt

Das Skript erstellt eine Ausgabedatei mit allen erfassten Informationen. Das Skript erstellt die Ausgabedatei innerhalb des /tmp-Verzeichnisses.

Linux: /opt/cloudhsm/bin/client_info

Windows: C:\Program Files\Amazon\CloudHSM\client_info

Warning

Dieses Skript hat ein bekanntes Problem für die Client-SDK 3-Versionen 3.1.0 bis 3.3.1. Wir empfehlen dringend, auf Version 3.3.2 zu aktualisieren, die eine Lösung für dieses Problem enthält. Weitere Informationen finden Sie auf der Seite [Bekannte Probleme](#), die Sie lesen sollten, bevor Sie dieses Tool verwenden.

AWS CloudHSM Kontingente

Kontingente, früher als Grenzwerte bezeichnet, sind die zugewiesenen Werte für AWS Ressourcen. Die folgenden Kontingente gelten für Ihre AWS CloudHSM Ressourcen pro AWS Region und AWS Konto. Das Standardkontingent ist der ursprüngliche Wert AWS, der von angewendet wird. Diese Werte sind in der folgenden Tabelle aufgeführt. Ein anpassbares Kontingent kann über das Standardkontingent hinaus erhöht werden.

Servicekontingente

Ressource	Standardkontingent	Anpassbar?
Cluster	4	Ja
HSMs	6	Ja
HSMs pro Cluster	28	Nein

Die empfohlene Möglichkeit, eine Kontingenterhöhung anzufordern, besteht darin, die [Servicekontingente-Konsole](#) zu öffnen. Wählen Sie in der Konsole Ihren Service und Ihr Kontingent aus, und senden Sie Ihre Anfrage. Weitere Informationen finden Sie in der [Dokumentation zu Servicekontingenten](#).

Die Kontingente in der folgenden Systemkontingentetabelle sind nicht anpassbar.

Systemkontingente

Ressource	Kontingent für hsm1.medium	Kontingent für hsm2m.medium
Maximale Schlüssel pro Cluster	3.300	Insgesamt 16.666 Schlüssel , wobei asymmetrische Schlüssel ein Maximum von 3.333 haben
Maximale Anzahl Benutzer pro Cluster	250	1,024
Maximale Länge eines Benutzernamens	31 Zeichen	31 Zeichen

Ressource	Kontingent für hsm1.medium	Kontingent für hsm2m.medium
Erforderlich Passwortlänge	8 bis 32 Zeichen	8 bis 32 Zeichen
Maximale Anzahl von gleichzeitigen Client-Verbindungen pro Cluster ¹	900	900
Maximale Anzahl von PKCS #11-Sitzungen pro Anwendung	1,024	1,024

[1] Eine Client-Verbindung für Client-SDK 3 ist ein Client-Daemon. Für das Client-SDK 5 ist eine Client-Verbindung eine Anwendung.

Downloads für das AWS CloudHSM Client-SDK

Die folgenden Themen bieten Downloads für AWS CloudHSM Kunden SDKs.

Note

Informationen darüber, welche Plattformen von den einzelnen Client-SDKs unterstützt werden, finden Sie unter [AWS CloudHSM Client SDK 5 unterstützte Plattformen](#) und [AWS CloudHSM Unterstützte Plattformen vom Client SDK 3](#).

Themen

- [AWS CloudHSM neueste Version des Client-SDK](#)
- [AWS CloudHSM frühere Client-SDK-Versionen](#)
- [AWS CloudHSM veraltete Client-SDK-Versionen](#)
- [AWS CloudHSM end-of-life SDK-Versionen für Kunden](#)

AWS CloudHSM neueste Version des Client-SDK

Im März 2021 wurde die Client-SDK-Version 5.0.0 AWS CloudHSM veröffentlicht, die ein völlig neues Client-SDK mit unterschiedlichen Anforderungen, Funktionen und Plattformunterstützung einführt.

Client SDK 5 wird für Produktionsumgebungen vollständig unterstützt und bietet dieselben Komponenten und denselben Support wie Client SDK 3. Weitere Informationen finden Sie unter [Vergleichen Sie die Unterstützung von AWS CloudHSM Client SDK-Komponenten](#).

Note

Ab CloudHSM SDK 5.17 aktualisieren wir unser Service Level Agreement, sodass es bis zu 3 frühere Nebenversionen und ein Jahr ab dem Veröffentlichungsdatum einer SDK-Version unterstützt. Wir werden neue Versionen des CloudHSM SDK veröffentlichen, um neue Funktionen bereitzustellen und Verbesserungen in Bezug auf Sicherheit, Stabilität und Leistung zu erzielen. Wir werden Download-Links für ältere und nicht unterstützte Versionen des CloudHSM SDK deaktivieren, sobald neue Versionen verfügbar sind.

Dieser Abschnitt enthält die neueste Version des Client-SDK. Informationen zu früheren Versionen finden Sie unter [Frühere SDK-Versionen](#). Informationen zu veralteten Versionen finden Sie unter [Veraltete SDK-Versionen](#).

Version 5 des Client-SDK: Version 5.17.1

Amazon Linux 2023

Laden Sie die Version 5.17.1 der Software für Amazon Linux 2023 auf der x86_64-Architektur herunter:

- [PKCS-#11-Bibliothek](#) (SHA256 checksum
e5d37660312c8d5b9c36a9b74555222a755276e2c1f360418c34125b2b715b97)
- [OpenSSL Dynamic Engine](#) (SHA256 checksum
9b84d756e56f19b830adbd46aa5ac5096ae28944e34dd8dac3bd534e487d7aee)
- [OpenSSL-Anbieter](#) (SHA256 checksum 5df0647ea5fe6da453bbf9a9578de9674ba93b722ec17dfe79825006211dc7a9)
- [JCE-Anbieter](#) (SHA256 checksum ac820e95618bf6cecf76b6ae0c5fb59198041f1d583178105a506d255d886fc9)
 - [Javadocs für AWS CloudHSM](#) (SHA256 checksum
89b5c2232c1739d61813d16f6d3177984bfff66754367883e2d7e997a363f30)
- [CloudHSM-CLI](#) (SHA256 checksum 3b2f966c05690c14c325ea37cf9799bad477b2573f1077a5319d92933af9e590)

Laden Sie die Version 5.17.1 der Software für Amazon Linux 2023 auf ARM64 Architektur herunter:

- [PKCS-#11-Bibliothek](#) (SHA256 checksum
8cf4f7e9bed86ed15198e0870075fa01cb8243d3646ae9a1417ee3eba0cb3111)
- [OpenSSL Dynamic Engine](#) (SHA256 checksum
831c7565caaea18c9c239b060347ec1b0e5121ec94d5bfe4cecdad0fd4b1aa9ab)
- [OpenSSL-Anbieter](#) (SHA256 checksum 73e91db43c98fa1eac6ce4d39f211d0dafa8445941b9f3adf77876cbeb443561)
- [JCE-Anbieter](#) (SHA256 checksum 1cc6bd9226ac98ebf898498f3b51b41b3ed58304f0290bd2045ab472e3bf3e14)
 - [Javadocs für AWS CloudHSM](#) (SHA256 checksum
89b5c2232c1739d61813d16f6d3177984bfff66754367883e2d7e997a363f30)
- [CloudHSM-CLI](#) (SHA256 checksum d716108dfc8f0e89e082603dbd5c95379b9f593628ab30bc611c37c169fae8d0)

Amazon Linux 2

Laden Sie die Version 5.17.1 der Software für Amazon Linux 2 auf der x86_64-Architektur herunter:

- [PKCS-#11-Bibliothek](#) (SHA256 checksum
c22875d9a81b2a5814d949b06d1467d4fbb111e67a6b29d83e961d95bdebd9e9)
- [OpenSSL Dynamic Engine](#) (SHA256 checksum
f54d264ede8624867c3010d6b3979c51cd05a34d90a4169208400013c8de76c5)
- [JCE-Anbieter](#) (SHA256 checksum c46297760a1d306541498e6a0295d7143273286dbb798c1c65bee38799b68e6b)
 - [Javadocs für AWS CloudHSM](#) (SHA256 checksum
89b5c2232c1739d61813d16f6d3177984bfff66754367883e2d7e997a363f30)
- [CloudHSM-CLI](#) (SHA256 checksum 87162212c93d8536d9247632221d855e841637a786c405eb47bc1410d62987d9)

Laden Sie die Version 5.17.1 der Software für Amazon Linux 2 auf ARM64 Architektur herunter:

- [PKCS-#11-Bibliothek](#) (SHA256 checksum
0abe258732376b20730c2d2e089458e83eff4e751df8e60c0c70d3d92e75d7a4)
- [OpenSSL Dynamic Engine](#) (SHA256 checksum
4f8101debd678327a26246e50df10fe79663fc264fed49deef63e036221fc0b)
- [JCE-Anbieter](#) (SHA256 checksum dfc83b8f2db9e97eab98bff431c3374181a0051884abe7e9424621fcd9a9c8c5)
 - [Javadocs für AWS CloudHSM](#) (SHA256 checksum
89b5c2232c1739d61813d16f6d3177984bfff66754367883e2d7e997a363f30)
- [CloudHSM-CLI](#) (SHA256 checksum bc43b221610d1c220ed402ee14bb0275ad79fb21a342639d1266579881da953a)

RHEL 10 (10.0+)

Laden Sie die Softwareversion 5.17.1 für RHEL 10 auf der x86_64-Architektur herunter:

- [PKCS-#11-Bibliothek](#) (SHA256 checksum
11076479568454ad1083bb6a81953a73f9175f6d8626e247e1e16439a5c105ec)
- [OpenSSL-Anbieter](#) (SHA256 checksum 6c3d7a58b38ef39c7b5310958eb684feed20edf6731603a8b727b4bd978f465f)
- [JCE-Anbieter](#) (SHA256 checksum 921883948344a06716d6b3b30f67708dedff550978efce6d5817b36c83e83b35)
 - [Javadocs für AWS CloudHSM](#) (SHA256 checksum
89b5c2232c1739d61813d16f6d3177984bfff66754367883e2d7e997a363f30)

- [CloudHSM-CLI](#) (SHA256 checksum 4d69481cc4e725e65811e031fe344cb178016cd268759bb910acdb14df9be413)

Laden Sie die Softwareversion 5.17.1 für RHEL 10 unter Architektur herunter: ARM64

- [PKCS-#11-Bibliothek](#) (SHA256 checksum 7a20af9fbe7a9874132c1314841ef2ead0daef92ced3ba2289938a88d0ee929e)
- [OpenSSL-Anbieter](#) (SHA256 checksum e762bb21086131725e9ca13b7d25e6fcb474888ae49317e1b045ddf826a8338c)
- [JCE-Anbieter](#) (SHA256 checksum c8bb96ded1abdfb54cd710fade3c91ce6030f5dc9ddeea639cc0a0cac5c26bb1)
 - [Javadocs für AWS CloudHSM](#) (SHA256 checksum 89b5c2232c1739d61813d16f6d3177984bffffb66754367883e2d7e997a363f30)
- [CloudHSM-CLI](#) (SHA256 checksum d48ccaecccc9d4247f0cf4c81a6f092704630933dbc89e0b0ba9a75629fd863b7)

RHEL 9 (9.2+)

Laden Sie die Version 5.17.1 der Software für RHEL 9 auf der x86_64-Architektur herunter:

- [PKCS-#11-Bibliothek](#) (SHA256 checksum df4f1de8e68e363b351770dd74782d204f04d4db646c554fa96182b429006618)
- [OpenSSL Dynamic Engine](#) (SHA256 checksum 06d2e65adb076e65b81b51bcea34b64f718b6aa29c173eebcacaf277de2c8d03ec)
- [OpenSSL-Anbieter](#) (SHA256 checksum 802e608484ab748eccdce8da0854dc668e90578d4c2bc6ede4caf4121d10171f)
- [JCE-Anbieter](#) (SHA256 checksum 1a4fdfe82cc79a4de85a19d6645916b6b388c919a71a22efae7bafaba195db0f)
 - [Javadocs für AWS CloudHSM](#) (SHA256 checksum 89b5c2232c1739d61813d16f6d3177984bffffb66754367883e2d7e997a363f30)
- [CloudHSM-CLI](#) (SHA256 checksum daab5b758c61f49b34b6f7e288debb121adc3d63b9652db086855d160c988570)

Laden Sie die Softwareversion 5.17.1 für RHEL 9 auf Architektur herunter: ARM64

- [PKCS-#11-Bibliothek](#) (SHA256 checksum 2fe6549339b743200c6be91b8abf8017c3f67c90f7db25ffd0ab1915b91d0f78)
- [OpenSSL Dynamic Engine](#) (SHA256 checksum a0a964dcb817251faafcd432e6260dff1d6e1bcdd7e7ffbd54a691cbe366b58)
- [OpenSSL-Anbieter](#) (SHA256 checksum bbcc9b3a2455328e6e7b11ff64b7e7653757960a74bfd9d6b3cc998b2a955fb2)
- [JCE-Anbieter](#) (SHA256 checksum f9c74e2ededb9d2fdb7076789223c4cde7a0845b6f999b006fa741e9428195c7)

- [Javadocs für AWS CloudHSM](#) (SHA256 checksum
89b5c2232c1739d61813d16f6d3177984bfff66754367883e2d7e997a363f30)
- [CloudHSM-CLI](#) (SHA256 checksum 0c4c1c7f4e06ba3b5b5dfe619abb0fa9d4729c87c668361fdb99b2a62f4b00fc)

RHEL 8 (8.3+)

Laden Sie die Version 5.17.1 der Software für RHEL 8 auf der x86_64-Architektur herunter:

- [PKCS-#11-Bibliothek](#) (SHA256 checksum
0f8465ab44f7388980b22bc1f8c6c3c3e09e809ba2712170c85db86918817803)
- [OpenSSL Dynamic Engine](#) (SHA256 checksum
8fe4f9fb6088d2a32e210696775d395b918ee867dede8b2152ff37d0d99cc8f3)
- [JCE-Anbieter](#) (SHA256 checksum 948d37f435e7b03ec8409ee48f89c265009c7e7a76e7a18921c5ef88f3504391)
 - [Javadocs für AWS CloudHSM](#) (SHA256 checksum
89b5c2232c1739d61813d16f6d3177984bfff66754367883e2d7e997a363f30)
- [CloudHSM-CLI](#) (SHA256 checksum 2168377badf708047ee14d2bc57b9234f01ce2c9f42568dbd1185723a63eeeb8)

Laden Sie die Version 5.17.1 der Software für RHEL 8 auf Architektur herunter: ARM64

- [PKCS-#11-Bibliothek](#) (SHA256 checksum 69db8b832edba06fe055ce9772bfad645a86e5e530d20e8f14738fc7cb06a5c7)
- [OpenSSL Dynamic Engine](#) (SHA256 checksum
ac7ad7b11e63d1fb7af57ff1e61b421aa0fccbca99ed15b0ddec1602fea2acd3)
- [JCE-Anbieter](#) (SHA256 checksum 7520a95076a7a794219677489d2c764502a6dcfeea484f10059e283f1aee6975)
 - [Javadocs für AWS CloudHSM](#) (SHA256 checksum
89b5c2232c1739d61813d16f6d3177984bfff66754367883e2d7e997a363f30)
- [CloudHSM-CLI](#) (SHA256 checksum 596acc3164695a262b15edbb92eda89ffd4b8d06b91617ce91058c2d99ae2a53)

Ubuntu 24.04 LTS

Laden Sie die Version 5.17.1 der Software für Ubuntu 24.04 LTS auf der x86_64-Architektur herunter:

- [PKCS-#11-Bibliothek](#) (SHA256 checksum
bc3146d5dc7aabe5359c41d2bd0fd97ed4bc2f389be6a4b94423065af9d60bd0)

- [OpenSSL Dynamic Engine](#) (SHA256 checksum 05d6afa19949b69af4325f55d285a89255ab2c7487aba898c74cbd4ee8e30fdf)
- [OpenSSL-Anbieter](#) (SHA256 checksum 3deba48c356478fdbb0236ad435cee8f7f8cfe3cec1a894aea3f79d80d398cce)
- [JCE-Anbieter](#) (SHA256 checksum ebacb95d8f92528e236a5342cb5c1ecbf74c228428fb4a080c0c972b697292b5)
 - [Javadocs für AWS CloudHSM](#) (SHA256 checksum 89b5c2232c1739d61813d16f6d3177984bfff66754367883e2d7e997a363f30)
- [CloudHSM-CLI](#) (SHA256 checksum 575f54610d26561bb98b69548de2f61a17b5b93ed49be13821be536d76ffdb75)

Laden Sie die Version 5.17.1 der Software für Ubuntu 24.04 LTS auf die Architektur herunter:
ARM64

- [PKCS-#11-Bibliothek](#) (SHA256 checksum b0c918575e611f486902ebbcc3ee84e15217247f5577b06764cfc620660c6ec4)
- [OpenSSL Dynamic Engine](#) (SHA256 checksum 57f0afb66c34a91e96bef4964dd3fb487fe5b9b06d0818ee9a2c90ad60068321)
- [OpenSSL-Anbieter](#) (SHA256 checksum dcf5128c6913e43e0a304d0221c243fd5228f235b78fac5ffd43478bb0d96062)
- [JCE-Anbieter](#) (SHA256 checksum 21d43992ace3aec9ec3711e7bd64ec716d3281f627e78a6f7a328a8e533beec4)
 - [Javadocs für AWS CloudHSM](#) (SHA256 checksum 89b5c2232c1739d61813d16f6d3177984bfff66754367883e2d7e997a363f30)
- [CloudHSM-CLI](#) (SHA256 checksum 096844ef80a7d573843682cfb3797a0c45608d004639ab0e671065f3964c06e3)

Ubuntu 22.04 LTS

Laden Sie die Version 5.17.1 der Software für Ubuntu 22.04 LTS auf der x86_64-Architektur herunter:

- [PKCS-#11-Bibliothek](#) (SHA256 checksum ec8c275832f9936e698106454e930268d7f2f310faffa2b7d03c3c03c6e297e1)
- [OpenSSL Dynamic Engine](#) (SHA256 checksum 316c3929363ad1d764f356bbf56dff121b3bccefa5060b94c5d9d7955f828ff4)
- [JCE-Anbieter](#) (SHA256 checksum 4c53b61bcdada0a684b605c0630f52b6b3bf6e9ed64f4c623f35fa4283e9770d)
 - [Javadocs für AWS CloudHSM](#) (SHA256 checksum 89b5c2232c1739d61813d16f6d3177984bfff66754367883e2d7e997a363f30)
- [CloudHSM-CLI](#) (SHA256 checksum 6b4ba5d3d06e5234b64b7c60310d5b477c2d8732fbbd3e9c60dcb56f2ef69a6e)

Laden Sie die Version 5.17.1 der Software für Ubuntu 22.04 LTS auf die Architektur herunter:
ARM64

- [PKCS-#11-Bibliothek](#) (SHA256 checksum d26d10425572f0624d2205e1409b88ccc5d793bcfeef6cd26fdb5c41f5c8129)
- [OpenSSL Dynamic Engine](#) (SHA256 checksum e3b6cd3e93f2645d9a23add1a514e314ad3d48a93b253e6b984577b8e502e1f0)
- [JCE-Anbieter](#) (SHA256 checksum 7ab3ea04d34f92c585121b110f620b632616a5585a4ed153fbb61fedef667b06)
 - [Javadocs für AWS CloudHSM](#) (SHA256 checksum 89b5c2232c1739d61813d16f6d3177984bfff66754367883e2d7e997a363f30)
- [CloudHSM-CLI](#) (SHA256 checksum 9cf4a05ebf2bce4a0c9e57dd2eef87ba47ac05c4a303d4465cbf41536736c685)

Windows Server 2025

Laden Sie die Version 5.17.1 der Software für Windows Server 2025 auf der x86_64-Architektur herunter:

- [PKCS-#11-Bibliothek](#) (SHA256 checksum 145ea74ff3ab70c0f5f1ccd8ced90b86f0b4f3ee029d2d9452e9b0724ae2f64e)
- [JCE-Anbieter](#) (SHA256 checksum 074bc0a990d51debea16e8691171c1cfc4e164795eb839be1d311858b157b0a6)
 - [Javadocs für AWS CloudHSM](#) (SHA256 checksum 89b5c2232c1739d61813d16f6d3177984bfff66754367883e2d7e997a363f30)
- [CloudHSM-CLI](#) (SHA256 checksum ddc11eee7b884ddaa203ded85f36553a9a636d35bf9dd4b6557c23a167ed7c4a)
- [Wichtiger Speicheranbieter \(KSP\)](#) (SHA256 checksum 5bef4732d23861795a8463c3a3511e0a235a29972d584c2f65384e87ba4bb48c)

Windows Server 2022

Laden Sie die Version 5.17.1 der Software für Windows Server 2022 auf der x86_64-Architektur herunter:

- [PKCS-#11-Bibliothek](#) (SHA256 checksum 145ea74ff3ab70c0f5f1ccd8ced90b86f0b4f3ee029d2d9452e9b0724ae2f64e)
- [JCE-Anbieter](#) (SHA256 checksum 074bc0a990d51debea16e8691171c1cfc4e164795eb839be1d311858b157b0a6)
 - [Javadocs für AWS CloudHSM](#) (SHA256 checksum 89b5c2232c1739d61813d16f6d3177984bfff66754367883e2d7e997a363f30)
- [CloudHSM-CLI](#) (SHA256 checksum ddc11eee7b884ddaa203ded85f36553a9a636d35bf9dd4b6557c23a167ed7c4a)

- [Wichtiger Speicheranbieter \(KSP\)](#) (SHA256 checksum 5bef4732d23861795a8463c3a3511e0a235a29972d584c2f65384e87ba4bb48c)

Windows Server 2019

Laden Sie die Version 5.17.1 der Software für Windows Server 2019 auf der x86_64-Architektur herunter:

- [PKCS-#11-Bibliothek](#) (SHA256 checksum 145ea74ff3ab70c0f5f1ccd8ced90b86f0b4f3ee029d2d9452e9b0724ae2f64e)
- [JCE-Anbieter](#) (SHA256 checksum 074bc0a990d51debea16e8691171c1cfc4e164795eb839be1d311858b157b0a6)
 - [Javadocs für AWS CloudHSM](#) (SHA256 checksum 89b5c2232c1739d61813d16f6d3177984bfff66754367883e2d7e997a363f30)
- [CloudHSM-CLI](#) (SHA256 checksum ddc11eee7b884ddaa203ded85f36553a9a636d35bf9dd4b6557c23a167ed7c4a)
- [Wichtiger Speicheranbieter \(KSP\)](#) (SHA256 checksum 5bef4732d23861795a8463c3a3511e0a235a29972d584c2f65384e87ba4bb48c)

Windows Server 2016

Laden Sie die Version 5.17.1 der Software für Windows Server 2016 auf der x86_64-Architektur herunter:

- [PKCS-#11-Bibliothek](#) (SHA256 checksum 145ea74ff3ab70c0f5f1ccd8ced90b86f0b4f3ee029d2d9452e9b0724ae2f64e)
- [JCE-Anbieter](#) (SHA256 checksum 074bc0a990d51debea16e8691171c1cfc4e164795eb839be1d311858b157b0a6)
 - [Javadocs für AWS CloudHSM](#) (SHA256 checksum 89b5c2232c1739d61813d16f6d3177984bfff66754367883e2d7e997a363f30)
- [CloudHSM-CLI](#) (SHA256 checksum ddc11eee7b884ddaa203ded85f36553a9a636d35bf9dd4b6557c23a167ed7c4a)
- [Wichtiger Speicheranbieter \(KSP\)](#) (SHA256 checksum 5bef4732d23861795a8463c3a3511e0a235a29972d584c2f65384e87ba4bb48c)

Das Client SDK 5.17.1 enthält Fehlerkorrekturen und Verbesserungen.

AWS CloudHSM frühere Client-SDK-Versionen

In diesem Abschnitt werden frühere Client-SDK-Versionen aufgeführt.

Version 5.17.0

Amazon Linux 2023

Laden Sie die Version 5.17.0 der Software für Amazon Linux 2023 auf der x86_64-Architektur herunter:

- [PKCS-#11-Bibliothek](#) (SHA256 checksum e0656d606c1f75b8ecf11df79f2d4aa17c0b68983bb20d271dabe02f7b97ca47)
- [OpenSSL Dynamic Engine](#) (SHA256 checksum b8917abe48799035018bc044fbc2e9db8d5568ebf29663ecc12d53ec2be5efd2)
- [OpenSSL-Anbieter](#) (SHA256 checksum 61e49cb0f1f9eb7bcf76f5ef36f01cf1fadfa56bf5eab94b05efdd93214c9cf7)
- [JCE-Anbieter](#) (SHA256 checksum 4ba2548fa27f3662c56fc140386366cfb40cde899695412c6d4731a7d0a70e43)
 - [Javadocs für AWS CloudHSM](#) (SHA256 checksum a89e6d7e4e1f4049fefa64c765e97928fc54c366249f315ac6a839704e1ffba8)
- [CloudHSM-CLI](#) (SHA256 checksum 0a9de06b3f15d38f3aa02dd0288d3a9ad41f525f692f539664168b9e6b95e18d)

Laden Sie die Version 5.17.0 der Software für Amazon Linux 2023 auf ARM64 Architektur herunter:

- [PKCS-#11-Bibliothek](#) (SHA256 checksum 9bd40e13bf777a4651721ebe5613cf2e4f0f339679d50f06b81f7f1b73c1fef5)
- [OpenSSL Dynamic Engine](#) (SHA256 checksum e592419bc0449664f0707281a46ebb775091386ac5b297e35ff0e4160468cbe0)
- [OpenSSL-Anbieter](#) (SHA256 checksum 46358ef68388928702d780564811814aa826c143ccc0624a2a45d8995226d33e)
- [JCE-Anbieter](#) (SHA256 checksum db5bd1494c742a64ca3f2bc57253b17655ad38ecc27d45422965466c1caf740f)
 - [Javadocs für AWS CloudHSM](#) (SHA256 checksum a89e6d7e4e1f4049fefa64c765e97928fc54c366249f315ac6a839704e1ffba8)
- [CloudHSM-CLI](#) (SHA256 checksum e3d21b4460a421058f65a597b0bb30a9fdda890363ebee107b124a1212d5a042)

Amazon Linux 2

Laden Sie die Version 5.17.0 der Software für Amazon Linux 2 auf der x86_64-Architektur herunter:

- [PKCS-#11-Bibliothek](#) (SHA256 checksum 17d96d5f4ad25de74fadcc7272fac825d43c496b0b7912e4d32b1ccfb67250bc)
- [OpenSSL Dynamic Engine](#) (SHA256 checksum 3d3de61bd16ef849e53827709b8c8bd9eaf89bd1b7d56b1a02455bca38fe1eb6)

- [JCE-Anbieter](#) (SHA256 checksum 5c268313ebf16cf762619a1e75f1d44efdbd6eaf5ba4bbce67ecfe96bbbed9775)
- [Javadocs für AWS CloudHSM](#) (SHA256 checksum a89e6d7e4e1f4049fefa64c765e97928fc54c366249f315ac6a839704e1ffba8)
- [CloudHSM-CLI](#) (SHA256 checksum 43b678eff0225590cc59c02645332f4459c8161a6742e562eb95902d1bec9134)

Laden Sie die Version 5.17.0 der Software für Amazon Linux 2 auf ARM64 Architektur herunter:

- [PKCS-#11-Bibliothek](#) (SHA256 checksum 7910ec55f8b106d804290b77e6d13cefb85ef79cbc728f69bb21eebeb7245c6b)
- [OpenSSL Dynamic Engine](#) (SHA256 checksum 399c16e0fbfa372710a83836517cfeaae0895513931427a458e7994ce91a321d)
- [JCE-Anbieter](#) (SHA256 checksum 1678581e31fcfe58625ca1e3c0762c4b96095065a868a5b7f0872214e5ab9e8)
 - [Javadocs für AWS CloudHSM](#) (SHA256 checksum a89e6d7e4e1f4049fefa64c765e97928fc54c366249f315ac6a839704e1ffba8)
- [CloudHSM-CLI](#) (SHA256 checksum 52257b580d73d902e9fd29f2c334f88cabee484c7eec8af28f7b2aa7864466e7)

RHEL 10 (10.0+)

Laden Sie die Softwareversion 5.17.0 für RHEL 10 auf der x86_64-Architektur herunter:

- [PKCS-#11-Bibliothek](#) (SHA256 checksum 037102ee2b7f9c244d647cfb06da282b99349e208d27b7b0aa80bd32e382a2d4)
- [OpenSSL-Anbieter](#) (SHA256 checksum 3d6d97df03a003bae579dabbf9abeefc1bd2fa6d799a6db01c3ef98d02145393)
- [JCE-Anbieter](#) (SHA256 checksum 4e69ec89acfefc19f922c68988d4674718d49d5cda2e0a288f7f0343ebfc270c)
 - [Javadocs für AWS CloudHSM](#) (SHA256 checksum a89e6d7e4e1f4049fefa64c765e97928fc54c366249f315ac6a839704e1ffba8)
- [CloudHSM-CLI](#) (SHA256 checksum 38a67191949cc8cb0c69ef2fe4694514b2ca6af4cf5f8763bd7f6860c501675a)

Laden Sie die Softwareversion 5.17.0 für RHEL 10 auf Architektur herunter: ARM64

- [PKCS-#11-Bibliothek](#) (SHA256 checksum bc383aff236d4b2a0fdbc5fd81907cf9212ad8f3cfc91fadeb6cdc14b9d3d581)
- [OpenSSL-Anbieter](#) (SHA256 checksum d9ead84f8d2e6a09044646d62fed293ca082431e5e0338bcbbf8acd023ecc0)
- [JCE-Anbieter](#) (SHA256 checksum c61fdb5f05cf486619deb3e23ecb171ce4485bb64c891bf4480fbe6a4276c3bc)
 - [Javadocs für AWS CloudHSM](#) (SHA256 checksum a89e6d7e4e1f4049fefa64c765e97928fc54c366249f315ac6a839704e1ffba8)

- [CloudHSM-CLI](#) (SHA256 checksum fbd4387ef3fab177de8c888d98d1b24bc042d23ea2f93834dfb5ee918518f406)

RHEL 9 (9.2+)

Laden Sie die Version 5.17.0 der Software für RHEL 9 auf der x86_64-Architektur herunter:

- [PKCS-#11-Bibliothek](#) (SHA256 checksum 7fc68025493fc4a0bc4d3b2fd11b17da4481b6bee51fc86a40f95fe4c206aa60)
- [OpenSSL Dynamic Engine](#) (SHA256 checksum 3e786e9218f657c93e1bf1b54c81f0f573e39d10147cf99485da73677b0d67ae)
- [OpenSSL-Anbieter](#) (SHA256 checksum c52b366f9c6c960d697794bd7840888bc51e1ee5aa654b8d2c83ec1626797c56)
- [JCE-Anbieter](#) (SHA256 checksum 427e9c68815188b59a0f9b2fd14793791d9a4ce4d3e5ddc34fce916af18a45f1)
 - [Javadocs für AWS CloudHSM](#) (SHA256 checksum a89e6d7e4e1f4049fefa64c765e97928fc54c366249f315ac6a839704e1ffba8)
- [CloudHSM-CLI](#) (SHA256 checksum 49d128639eea0e588c95281635a33b0e62837cf5a9d992e0bacb5b598a0037ed)

Laden Sie die Softwareversion 5.17.0 für RHEL 9 auf Architektur herunter: ARM64

- [PKCS-#11-Bibliothek](#) (SHA256 checksum 3ae8f5068abb3d9ca75ac21b160acc7ee84c7ded6107a889602448af51d5bd82)
- [OpenSSL Dynamic Engine](#) (SHA256 checksum 64c5feaf86fc40e28616156f17942cc46d8ac93babdd8af59dd98a1f2e73117d)
- [OpenSSL-Anbieter](#) (SHA256 checksum 24df4296ba0ae15f6b7060ccffeca0d1a24f58d9e8da8eb6a4afa3a6a50770fd)
- [JCE-Anbieter](#) (SHA256 checksum 648d7948ab8a1271d7cf0eddc2ca32e960979be481b99a1ce2ce8b9fa3a2b2ed)
 - [Javadocs für AWS CloudHSM](#) (SHA256 checksum a89e6d7e4e1f4049fefa64c765e97928fc54c366249f315ac6a839704e1ffba8)
- [CloudHSM-CLI](#) (SHA256 checksum edd1f3707aeea285a55cf24d2e2385489458a69e5d7dab4014783f5cdc232d77)

RHEL 8 (8.3+)

Laden Sie die Version 5.17.0 der Software für RHEL 8 auf der x86_64-Architektur herunter:

- [PKCS-#11-Bibliothek](#) (SHA256 checksum 85e070ed3273ead9698711dfc2c3e590dda1a01f99074943459ddc5ac052872d)
- [OpenSSL Dynamic Engine](#) (SHA256 checksum d0c8e5603f4c99d0ba547c0ba40361c15977f3349ac120d514afed59abf32c83)

- [JCE-Anbieter](#) (SHA256 checksum 4567b9a236d0dfd0410863beaced6d5d4e71acb06b2b136c7c40ce6e704f3698)
- [Javadocs für AWS CloudHSM](#) (SHA256 checksum a89e6d7e4e1f4049fefa64c765e97928fc54c366249f315ac6a839704e1ffba8)
- [CloudHSM-CLI](#) (SHA256 checksum f8a090752849fac9bf99ef67b59f571e476908dccec1285b466c25e6cc71793c)

Laden Sie die Version 5.17.0 der Software für RHEL 8 auf Architektur herunter: ARM64

- [PKCS-#11-Bibliothek](#) (SHA256 checksum 71ade413435947eb2093727cb0e7a5c5f084ca859cb38d728a8c3d23f3b49bd0)
- [OpenSSL Dynamic Engine](#) (SHA256 checksum 02210da06b3fd22813fce8b89b753d150c8f321df405c40dd9f0172dc03968a7)
- [JCE-Anbieter](#) (SHA256 checksum 80ac5dbaad1f644d6b9551e6a1d95d4b4a51ae82d34e45593be77b1068f7c910)
- [Javadocs für AWS CloudHSM](#) (SHA256 checksum a89e6d7e4e1f4049fefa64c765e97928fc54c366249f315ac6a839704e1ffba8)
- [CloudHSM-CLI](#) (SHA256 checksum c3c56c3b9fed1dc51761d9412007c727d3b608377a089bd637a44bc374744d5c)

Ubuntu 24.04 LTS

Laden Sie die Version 5.17.0 der Software für Ubuntu 24.04 LTS auf der x86_64-Architektur herunter:

- [PKCS-#11-Bibliothek](#) (SHA256 checksum b3869a6fe9f2031cb5f36ac8d2a5a5280451159e7a68f0d6f91f98b5f0100939)
- [OpenSSL Dynamic Engine](#) (SHA256 checksum 3e46443a3f8425f62cec8d8ca414c4b5446b86b42012877ebe44a17303403413)
- [OpenSSL-Anbieter](#) (SHA256 checksum 69c2e61894e954ac1d0a785025902af5aa1922a7f6011a847f2111e153db706e)
- [JCE-Anbieter](#) (SHA256 checksum 663da1a7b13a1055bb90fd7b168b69fcfb000eb91c1c3ce6062d8cc8db607fe4)
- [Javadocs für AWS CloudHSM](#) (SHA256 checksum a89e6d7e4e1f4049fefa64c765e97928fc54c366249f315ac6a839704e1ffba8)
- [CloudHSM-CLI](#) (SHA256 checksum 0d57b4f08bd768ed595978d2a6d3fe6bbbf8398a784c66c4e30c19ecf6730b5d)

Laden Sie die Version 5.17.0 der Software für Ubuntu 24.04 LTS auf die Architektur herunter: ARM64

- [PKCS-#11-Bibliothek](#) (SHA256 checksum 94d9a36a3d3f2f04c10539713b98ddad68654c739572064e4c123005a9caf575)

- [OpenSSL Dynamic Engine](#) (SHA256 checksum
a442950055a7051f50b4cdfa7ec347c1b207b0bbf7e159b31a3bf96896b66146)
- [OpenSSL-Anbieter](#) (SHA256 checksum dcc4b5a370da176a86e47683a78703a5604d6b7b49ba687377460f48c3ddff26)
- [JCE-Anbieter](#) (SHA256 checksum 1fe1fa59a3e5c32edd5668604e43f7ce152473514b25ee736dfb0e9cf87c92c9)
 - [Javadocs für AWS CloudHSM](#) (SHA256 checksum
a89e6d7e4e1f4049fefa64c765e97928fc54c366249f315ac6a839704e1ffba8)
- [CloudHSM-CLI](#) (SHA256 checksum 2bf2c13bf1520eaf2a45eb55aaba3334fa54da686d22d1a03005ec8f62ea8895)

Ubuntu 22.04 LTS

Laden Sie die Version 5.17.0 der Software für Ubuntu 22.04 LTS auf der x86_64-Architektur herunter:

- [PKCS-#11-Bibliothek](#) (SHA256 checksum 2dc1f00952f69b59b0498e85c2f2009f80c9a2f9f28104995b0291da5f45d805)
- [OpenSSL Dynamic Engine](#) (SHA256 checksum
df0e7ff8852a93fb4c5807317d3438ab96ffe69f19ba144d872991cbcb4172f6)
- [JCE-Anbieter](#) (SHA256 checksum 77a3cc37e066ae0ad5655ddc3dd38a0513bdcbd3c9d3d780e2973ac42a75035e)
 - [Javadocs für AWS CloudHSM](#) (SHA256 checksum
a89e6d7e4e1f4049fefa64c765e97928fc54c366249f315ac6a839704e1ffba8)
- [CloudHSM-CLI](#) (SHA256 checksum 88ca1614178771991649b8117ddf27ffb99c6842ee89f68e77c6df3470e6ba78)

Laden Sie die Version 5.17.0 der Software für Ubuntu 22.04 LTS auf die Architektur herunter:
ARM64

- [PKCS-#11-Bibliothek](#) (SHA256 checksum 0e1f62783c16deb9f188c9cf0085fe972671e5ea9bf0fc2fc3255f908885949c)
- [OpenSSL Dynamic Engine](#) (SHA256 checksum
109d25583e2d5ed0fe821a77098d8f2c815011f7873168e0110aeb857dc903ab)
- [JCE-Anbieter](#) (SHA256 checksum 9151661a39f6394aa3ace682bd7964095ed5300244582d3f6b1f81255ebf22cf)
 - [Javadocs für AWS CloudHSM](#) (SHA256 checksum
a89e6d7e4e1f4049fefa64c765e97928fc54c366249f315ac6a839704e1ffba8)
- [CloudHSM-CLI](#) (SHA256 checksum 66e408cea800a93becf378b8611595ad8cd1bf51e0ae7feebc548fe3099b477c)

Windows Server 2025

Laden Sie die Version 5.17.0 der Software für Windows Server 2025 auf der x86_64-Architektur herunter:

- [PKCS-#11-Bibliothek](#) (SHA256 checksum
77dd4ad5da0d4a21c82ff2ea03c09a6b8c2005e27b6aba6724e4bec75606b0e7)
- [JCE-Anbieter](#) (SHA256 checksum c2c4766b67f02a93cca22b166e63fe6f4c71bfd9f7b45aa2da6a603d825e5c1a)
 - [Javadocs für AWS CloudHSM](#) (SHA256 checksum
a89e6d7e4e1f4049fefa64c765e97928fc54c366249f315ac6a839704e1ffba8)
- [CloudHSM-CLI](#) (SHA256 checksum 4feb0aeb7554ddd919eb7df301a4da537fffb0577a89a053fba802a2d3657a2a)
- [Wichtiger Speicheranbieter \(KSP\)](#) (SHA256 checksum
3766d4d4ab1bd7afa6b72e74bc80d8e5054481c8535f5a5c841cc320617c4e1e)

Windows Server 2022

Laden Sie die Softwareversion 5.17.0 für Windows Server 2022 auf der x86_64-Architektur herunter:

- [PKCS-#11-Bibliothek](#) (SHA256 checksum
77dd4ad5da0d4a21c82ff2ea03c09a6b8c2005e27b6aba6724e4bec75606b0e7)
- [JCE-Anbieter](#) (SHA256 checksum c2c4766b67f02a93cca22b166e63fe6f4c71bfd9f7b45aa2da6a603d825e5c1a)
 - [Javadocs für AWS CloudHSM](#) (SHA256 checksum
a89e6d7e4e1f4049fefa64c765e97928fc54c366249f315ac6a839704e1ffba8)
- [CloudHSM-CLI](#) (SHA256 checksum 4feb0aeb7554ddd919eb7df301a4da537fffb0577a89a053fba802a2d3657a2a)
- [Wichtiger Speicheranbieter \(KSP\)](#) (SHA256 checksum
3766d4d4ab1bd7afa6b72e74bc80d8e5054481c8535f5a5c841cc320617c4e1e)

Windows Server 2019

Laden Sie die Softwareversion 5.17.0 für Windows Server 2019 auf der x86_64-Architektur herunter:

- [PKCS-#11-Bibliothek](#) (SHA256 checksum
77dd4ad5da0d4a21c82ff2ea03c09a6b8c2005e27b6aba6724e4bec75606b0e7)
- [JCE-Anbieter](#) (SHA256 checksum c2c4766b67f02a93cca22b166e63fe6f4c71bfd9f7b45aa2da6a603d825e5c1a)

- [Javadocs für AWS CloudHSM](#) (SHA256 checksum
a89e6d7e4e1f4049fefa64c765e97928fc54c366249f315ac6a839704e1ffba8)
- [CloudHSM-CLI](#) (SHA256 checksum 4feb0aeb7554ddd919eb7df301a4da537fffb0577a89a053fba802a2d3657a2a)
- [Wichtiger Speicheranbieter \(KSP\)](#) (SHA256 checksum
3766d4d4ab1bd7afa6b72e74bc80d8e5054481c8535f5a5c841cc320617c4e1e)

Windows Server 2016

Laden Sie die Softwareversion 5.17.0 für Windows Server 2016 auf der x86_64-Architektur herunter:

- [PKCS-#11-Bibliothek](#) (SHA256 checksum
77dd4ad5da0d4a21c82ff2ea03c09a6b8c2005e27b6aba6724e4bec75606b0e7)
- [JCE-Anbieter](#) (SHA256 checksum c2c4766b67f02a93cca22b166e63fe6f4c71bfd9f7b45aa2da6a603d825e5c1a)
 - [Javadocs für AWS CloudHSM](#) (SHA256 checksum
a89e6d7e4e1f4049fefa64c765e97928fc54c366249f315ac6a839704e1ffba8)
- [CloudHSM-CLI](#) (SHA256 checksum 4feb0aeb7554ddd919eb7df301a4da537fffb0577a89a053fba802a2d3657a2a)
- [Wichtiger Speicheranbieter \(KSP\)](#) (SHA256 checksum
3766d4d4ab1bd7afa6b72e74bc80d8e5054481c8535f5a5c841cc320617c4e1e)

Das Client SDK 5.17.0 bietet neue Funktionen und Verbesserungen für mehrere Komponenten.

Plattformunterstützung

- RHEL 8-Unterstützung für Architektur hinzugefügt. ARM64

JCE

- Unterstützung für OpenJDK 2.5 auf allen Plattformen hinzugefügt
- Client SDK 5.17.0 ist die letzte Version, die OpenJDK 8 unterstützt.

Neue Features

- Einführung des neuen OpenSSL Provider SDK für verbesserte OpenSSL 3.2+-Kompatibilität und zusätzliche kryptografische Operationen.
- Unterstützung für Ed25519-Kurven zur EC-Schlüsselgenerierung hinzugefügt.

- Unterstützung für das Signieren und Verifizieren von eDDSA (Edwards-Curve Digital Signature Algorithm) in der CloudHSM-CLI hinzugefügt.

CloudHSM-CLI

- Unterstützung für Ed25519-Kurven zur EC-Schlüsselgenerierung für Typen vom Typ hsm2m.medium ohne FIPS hinzugefügt.
- Unterstützung für das Signieren und Verifizieren von eDDSA (Edwards-Curve Digital Signature Algorithm) in der CloudHSM-CLI hinzugefügt. Weitere Informationen erhalten Sie unter [Generieren Sie eine Signatur mit dem HashEd DSA-Mechanismus in der CloudHSM-CLI](#) und [Überprüfen Sie eine mit dem HashEd DSA-Mechanismus signierte Signatur in der CloudHSM-CLI](#).

OpenSSL-Anbieter

- Das neue OpenSSL Provider SDK bietet die Integration mit der OpenSSL 3.2+ Provider-Architektur für die Typen hsm2m.medium.

Fehlerbehebungen/Verbesserungen

- Verbesserte Wiederholungsmechanismen für kryptografische Operationen.

Version 5.16.2

Amazon Linux 2023

Laden Sie die Version 5.16.2 der Software für Amazon Linux 2023 auf der x86_64-Architektur herunter:

- [PKCS-#11-Bibliothek](#) (SHA256 checksum
e223649cbf689afaeca445b962cceaab3c143b901ee7974a7a5360b6424c409b)
- [OpenSSL Dynamic Engine](#) (SHA256 checksum
9b51e8b1429f22705399299c111385e9434873a1dd26dda2c70498646590148c)
- [JCE-Anbieter](#) (SHA256 checksum 45ec10c2e3d592fb1fe56cc9dee1d8653e3a6dfa15b8185618fd3cf66c567314)
- [Javadocs für AWS CloudHSM](#) (SHA256 checksum
17029e68946b83d47c8512f824599338e7315a99317e4fc33a2058cdc5512d0d)
- [CloudHSM-CLI](#) (SHA256 checksum 8134a0e316286310eda1606b6a14e1c9fd6f0e27934a43e55e24d3094ef870a6)

Laden Sie die Version 5.16.2 der Software für Amazon Linux 2023 auf ARM64 Architektur herunter:

- [PKCS-#11-Bibliothek](#) (SHA256 checksum
7556c69294d0c67b359cf9995ec7047cdc90e21542d2c9d5e110e35f13c329c5)
- [OpenSSL Dynamic Engine](#) (SHA256 checksum
6ab4f3b9731a6f26891658f5e3a0ff74255096390c42aa7f702ba07e0bd924b1)
- [JCE-Anbieter](#) (SHA256 checksum 98b80696762a73f3f9f9be6c3c7af0f8f9db23ba63d66d4d941690b6698019702)
 - [Javadocs für AWS CloudHSM](#) (SHA256 checksum
17029e68946b83d47c8512f824599338e7315a99317e4fc33a2058cdc5512d0d)
- [CloudHSM-CLI](#) (SHA256 checksum 649397606f527f330049cf7adb5ad667126dfd7f63f878f1f46b469e2f127814)

Amazon Linux 2

Laden Sie die Version 5.16.2 der Software für Amazon Linux 2 auf der x86_64-Architektur herunter:

- [PKCS-#11-Bibliothek](#) (SHA256 checksum 073ee736ad9c7de6e6451afdecc4d77e6ed1a63f314a7fc09373ff35986fa0fc)
- [OpenSSL Dynamic Engine](#) (SHA256 checksum
b174fb24a15f1e22dd520ea9ac4ac14229c2adce14ac6df8901f9d3c0e7e24f6)
- [JCE-Anbieter](#) (SHA256 checksum f18e42f2679ab0fb51c421965a5dd6c8fe1361442d92403d1d230bedb003b27a)
 - [Javadocs für AWS CloudHSM](#) (SHA256 checksum
17029e68946b83d47c8512f824599338e7315a99317e4fc33a2058cdc5512d0d)
- [CloudHSM-CLI](#) (SHA256 checksum 5a6c83833b60017dd494c31fd4abf3896f57177be7951674c85d0d4e909d2041)

Laden Sie die Version 5.16.2 der Software für Amazon Linux 2 auf ARM64 Architektur herunter:

- [PKCS-#11-Bibliothek](#) (SHA256 checksum
df2c97a27855916983f242eba05e0741335158432ac46ebbeae47c4a3b7b0bef)
- [OpenSSL Dynamic Engine](#) (SHA256 checksum
d25456e3da39c63d9c0d4a928815f995239b146f4f0b26a8c50c742905543207)
- [JCE-Anbieter](#) (SHA256 checksum 7e31734efbaae49203e57d7ad4d1e291faa5b7e16c469a3596da82a7380bceeb)
 - [Javadocs für AWS CloudHSM](#) (SHA256 checksum
17029e68946b83d47c8512f824599338e7315a99317e4fc33a2058cdc5512d0d)
- [CloudHSM-CLI](#) (SHA256 checksum b77eb1f9ff58ccc864854d7f261e331ce50836cd770efc8e8eaeabb33d78c23e2)

RHEL 10 (10.0+)

Laden Sie die Software der Version 5.16.2 für RHEL 10 auf der x86_64-Architektur herunter:

- [PKCS-#11-Bibliothek](#) (SHA256 checksum 8044cb40c9c65ae944900c740308e68eeb0156d3b93051d60463cd3f6095ef67)
- [JCE-Anbieter](#) (SHA256 checksum a1b7f110c2b15ba7ce9adee74baf1f108d85146269321991314f9153a074ca27)
 - [Javadocs für AWS CloudHSM](#) (SHA256 checksum 17029e68946b83d47c8512f824599338e7315a99317e4fc33a2058cdc5512d0d)
- [CloudHSM-CLI](#) (SHA256 checksum 3ded29fdb74537b284cf871af91dd64fce8483770141a3e4470e94fb69524621)

Laden Sie die Software der Version 5.16.2 für RHEL 10 auf Architektur herunter: ARM64

- [PKCS-#11-Bibliothek](#) (SHA256 checksum 643fa23ffea47dfb21fd326cacf99e0e45f44d767ca6198ab06023239eda5d2c)
- [JCE-Anbieter](#) (SHA256 checksum e6b5c2760c60f5c82a248d36f3b47cea9b1bbb743cf1c02a1c13554b70ab87f8)
 - [Javadocs für AWS CloudHSM](#) (SHA256 checksum 17029e68946b83d47c8512f824599338e7315a99317e4fc33a2058cdc5512d0d)
- [CloudHSM-CLI](#) (SHA256 checksum a6af281ff9f429f964ac9d2babc251b4423748d2b2e6da22fc51b9fd5c59714c)

RHEL 9 (9.2+)

Laden Sie die Softwareversion 5.16.2 für RHEL 9 auf der x86_64-Architektur herunter:

- [PKCS-#11-Bibliothek](#) (SHA256 checksum e0a44849c9cb562cf803ff182e54a35e7eee682efa2a99fdfac51a996822d126)
- [OpenSSL Dynamic Engine](#) (SHA256 checksum 97463a8a88239d4a81854f2a4da7e0371dbcb375a662c7eae7753c371ef3c39f)
- [JCE-Anbieter](#) (SHA256 checksum 10cd4b9826451b7e9a11390cabcc7df548f5b3197e8bc87f5efe04605e503dc1)
 - [Javadocs für AWS CloudHSM](#) (SHA256 checksum 17029e68946b83d47c8512f824599338e7315a99317e4fc33a2058cdc5512d0d)
- [CloudHSM-CLI](#) (SHA256 checksum 6c7fc59dcce4d7bc0c4460ae7c793f13fcbb306924942bc6e1457f729602f76f)

Laden Sie die Software der Version 5.16.2 für RHEL 9 auf Architektur herunter: ARM64

- [PKCS-#11-Bibliothek](#) (SHA256 checksum 7b8d3f5e2bd09f30ddc1599619c63aa4452b16e1037bf10af070c055ae48cbd7)
- [OpenSSL Dynamic Engine](#) (SHA256 checksum 4edb0315c050b3b0699cb244fe36b987c9fd2921999cf7519fb669354d2efe2d)

- [JCE-Anbieter](#) (SHA256 checksum eae2064d9523df9f819ef4c7950a0ad15ef8bb01d8dcec062ee2558f5bebf029)
- [Javadocs für AWS CloudHSM](#) (SHA256 checksum 17029e68946b83d47c8512f824599338e7315a99317e4fc33a2058cdc5512d0d)
- [CloudHSM-CLI](#) (SHA256 checksum f9f4073ebcde2b654b706efb1c0f99c08fe24903576e760029097b0e3c783f6a)

RHEL 8 (8.3+)

Laden Sie die Softwareversion 5.16.2 für RHEL 8 auf der x86_64-Architektur herunter:

- [PKCS-#11-Bibliothek](#) (SHA256 checksum 9a3e023fd5ac8c444ef07bb28b96e81b467e066c9dc2da19f648f38eb04d05c9)
- [OpenSSL Dynamic Engine](#) (SHA256 checksum 6fc6ca58f16fe2f486f4730cd0a6029e9eebae98b3aed41fd8c605fe5c36c7a0)
- [JCE-Anbieter](#) (SHA256 checksum cb6e9eccfeebca7377b1fb815c2ae306b39b271513ce73a6cd5f1dcc55eac4c)
- [Javadocs für AWS CloudHSM](#) (SHA256 checksum 17029e68946b83d47c8512f824599338e7315a99317e4fc33a2058cdc5512d0d)
- [CloudHSM-CLI](#) (SHA256 checksum e848f3f2221e7f6b984bd2001331f9e90f20e6d7351484e4e2f12431c00d87e5)

Ubuntu 24.04 LTS

Laden Sie die Software der Version 5.16.2 für Ubuntu 24.04 LTS auf der x86_64-Architektur herunter:

- [PKCS-#11-Bibliothek](#) (SHA256 checksum d993f8c84d15cf93432b73cb37693c4ab68b3e765c35742adef3f947744b9893)
- [OpenSSL Dynamic Engine](#) (SHA256 checksum 055f0e4cc74ec7ead77fff3970422b4f1729e26a2cf3cc2be530f0e6a06f887e)
- [JCE-Anbieter](#) (SHA256 checksum 5bd62f8f5cb3ff4ae9430020aa4f5c8a29cb635674d2400d989f512c370c755c)
 - [Javadocs für AWS CloudHSM](#) (SHA256 checksum 17029e68946b83d47c8512f824599338e7315a99317e4fc33a2058cdc5512d0d)
- [CloudHSM-CLI](#) (SHA256 checksum de263f6035cf7280a982434448aa079164050e4a09508816fd06bf000cf15f0c)

Laden Sie die Version 5.16.2 der Software für Ubuntu 24.04 LTS auf die Architektur herunter:
ARM64

- [PKCS-#11-Bibliothek](#) (SHA256 checksum a6496f28f82f3e92b0acc8e3ae1fbe230a6d3ed82586e8ca6fe6d2f82c275e7c)

- [OpenSSL Dynamic Engine](#) (SHA256 checksum 82275d6ac2628b4291162e3609d6788b4cf01d6488a9b6a84ed8787a5a96e873)
- [JCE-Anbieter](#) (SHA256 checksum 67800957c69efca0fe01623364cb3b152f2780d15624955130d507a3f591e8e5)
 - [Javadocs für AWS CloudHSM](#) (SHA256 checksum 17029e68946b83d47c8512f824599338e7315a99317e4fc33a2058cdc5512d0d)
- [CloudHSM-CLI](#) (SHA256 checksum 088f53d75250aa0ce3db7a7ba3f06ad7804b3dbeed6226683a3252890a867ac6)

Ubuntu 22.04 LTS

Laden Sie die Version 5.16.2 der Software für Ubuntu 22.04 LTS auf der x86_64-Architektur herunter:

- [PKCS-#11-Bibliothek](#) (SHA256 checksum b9799ba75fb8885782283abae7f49aa76a72d27f692654dd3a415157f0ea9043)
- [OpenSSL Dynamic Engine](#) (SHA256 checksum 3c7675e9b7fa4bb692865559f09fc1f9fd78dadb97d8bd6b5d1fc03367d75b86)
- [JCE-Anbieter](#) (SHA256 checksum 37c21821ad4d9fde9551112e8a6d7ec6b7bb62a540c0c08b6edd650d5c87b5cf)
 - [Javadocs für AWS CloudHSM](#) (SHA256 checksum 17029e68946b83d47c8512f824599338e7315a99317e4fc33a2058cdc5512d0d)
- [CloudHSM-CLI](#) (SHA256 checksum 7d3a724892e993aebfb06ad5c3d1ef8fc650d65dff0a7de3cb9c17c089badc58)

Laden Sie die Version 5.16.2 der Software für Ubuntu 22.04 LTS auf die Architektur herunter:
ARM64

- [PKCS-#11-Bibliothek](#) (SHA256 checksum 8a2b392e48b7f9021966f6c457444ddd0964f9bb85b7ecb988bc0d372bbef6c9)
- [OpenSSL Dynamic Engine](#) (SHA256 checksum cca0736d208d59d070ced8b82181e24919f21fbf0497e5a8a2f40436e06a8dd9)
- [JCE-Anbieter](#) (SHA256 checksum c12e1b9bfe1cc78ecf35892050dd06904c6a29736cc628ac1481dca4ec3e3bfe)
 - [Javadocs für AWS CloudHSM](#) (SHA256 checksum 17029e68946b83d47c8512f824599338e7315a99317e4fc33a2058cdc5512d0d)
- [CloudHSM-CLI](#) (SHA256 checksum 586ee7663d54e76652e9f073772fcbda218dc3c6432a221d664ffbabee3590cd)

Windows Server 2025

Laden Sie die Software der Version 5.16.2 für Windows Server 2025 auf der x86_64-Architektur herunter:

- [PKCS-#11-Bibliothek](#) (SHA256 checksum b5f55bc72fc994244f09e0252c7fc4a3ddf9607b20860c8718ff1d8a3db0a177)
- [JCE-Anbieter](#) (SHA256 checksum c28ea8164f277bf067f4630d433ab7090b370225116a2d594dd563dfb5f9f991)
 - [Javadocs für AWS CloudHSM](#) (SHA256 checksum 17029e68946b83d47c8512f824599338e7315a99317e4fc33a2058cdc5512d0d)
- [CloudHSM-CLI](#) (SHA256 checksum db708fc33bdc91b9ad554aa63de86a572d3ccff4ba3ab10feac9c5fd9ed79b94)
- [Wichtiger Speicheranbieter \(KSP\)](#) (SHA256 checksum c060be59673f325240bd0f1ccc9d9d97639eb7937fa323d71e223451bc0d2e19)

Windows Server 2022

Laden Sie die Softwareversion 5.16.2 für Windows Server 2022 auf der x86_64-Architektur herunter:

- [PKCS-#11-Bibliothek](#) (SHA256 checksum b5f55bc72fc994244f09e0252c7fc4a3ddf9607b20860c8718ff1d8a3db0a177)
- [JCE-Anbieter](#) (SHA256 checksum c28ea8164f277bf067f4630d433ab7090b370225116a2d594dd563dfb5f9f991)
 - [Javadocs für AWS CloudHSM](#) (SHA256 checksum 17029e68946b83d47c8512f824599338e7315a99317e4fc33a2058cdc5512d0d)
- [CloudHSM-CLI](#) (SHA256 checksum db708fc33bdc91b9ad554aa63de86a572d3ccff4ba3ab10feac9c5fd9ed79b94)
- [Wichtiger Speicheranbieter \(KSP\)](#) (SHA256 checksum c060be59673f325240bd0f1ccc9d9d97639eb7937fa323d71e223451bc0d2e19)

Windows Server 2019

Laden Sie die Softwareversion 5.16.2 für Windows Server 2019 auf der x86_64-Architektur herunter:

- [PKCS-#11-Bibliothek](#) (SHA256 checksum b5f55bc72fc994244f09e0252c7fc4a3ddf9607b20860c8718ff1d8a3db0a177)
- [JCE-Anbieter](#) (SHA256 checksum c28ea8164f277bf067f4630d433ab7090b370225116a2d594dd563dfb5f9f991)
 - [Javadocs für AWS CloudHSM](#) (SHA256 checksum 17029e68946b83d47c8512f824599338e7315a99317e4fc33a2058cdc5512d0d)
- [CloudHSM-CLI](#) (SHA256 checksum db708fc33bdc91b9ad554aa63de86a572d3ccff4ba3ab10feac9c5fd9ed79b94)

- [Wichtiger Speicheranbieter \(KSP\)](#) (SHA256 checksum
c060be59673f325240bd0f1ccc9d9d97639eb7937fa323d71e223451bc0d2e19)

Windows Server 2016

Laden Sie die Softwareversion 5.16.2 für Windows Server 2016 auf der x86_64-Architektur herunter:

- [PKCS-#11-Bibliothek](#) (SHA256 checksum b5f55bc72fc994244f09e0252c7fc4a3ddf9607b20860c8718ff1d8a3db0a177)
- [JCE-Anbieter](#) (SHA256 checksum c28ea8164f277bf067f4630d433ab7090b370225116a2d594dd563dfb5f9f991)
 - [Javadocs für AWS CloudHSM](#) (SHA256 checksum
17029e68946b83d47c8512f824599338e7315a99317e4fc33a2058cdc5512d0d)
- [CloudHSM-CLI](#) (SHA256 checksum db708fc33bdc91b9ad554aa63de86a572d3ccff4ba3ab10feac9c5fd9ed79b94)
- [Wichtiger Speicheranbieter \(KSP\)](#) (SHA256 checksum
c060be59673f325240bd0f1ccc9d9d97639eb7937fa323d71e223451bc0d2e19)

Das Client-SDK 5.16.2 fügt Unterstützung für die RHEL 10-Plattform hinzu und entfernt die Unterstützung für Ubuntu 20.04 LTS. Das Client-SDK 5.16.2 bietet außerdem Unterstützung für JCE Provider und KSP mit neuen Funktionen und bietet gleichzeitig Bugfixes und Verbesserungen in anderen Bereichen. SDKs

Plattformunterstützung

- RHEL 10-Unterstützung für x86- und ARM-Architekturen hinzugefügt.
- OpenSSL Engine wird in RHEL 10 im Rahmen der Umstellung von Red Hat auf die neuere Provider-basierte Architektur, die in OpenSSL 3.0 eingeführt wurde, nicht unterstützt.

CloudHSM-CLI

- Es wurde ein Fehler behoben, bei dem beim Signieren mit vorgehashten Daten Token im interaktiven Modus nicht gelöscht wurden.
- Es wurde ein Fehler behoben `quorum token-sign list`, der zuvor zu Fehlern führte, wenn drei oder mehr Tokens vorhanden waren.

JCE-Anbieter

- Unterstützung für die Suche nach EC-Schlüsseln (Elliptic Curve) mit ECParameters und ECPoint für die Typen hsm2m.medium wurde hinzugefügt.
- sharedInfoParameterunterstützung für ECDH mit X9.63 KDF hinzugefügt.

PKCS-#11-Bibliothek

- Unterstützung für die Suche nach Tasten mit elliptischen Kurven (EC) mit ECParameters und für die Typen hsm2m.medium wurde hinzugefügt. ECPoint

Fehlerbehebungen/Verbesserungen

- Verbesserte Funktion zum Auflisten von Tokens, sodass mehr Token in einem einzigen Vorgang angezeigt werden können.
- Das Problem beim Löschen von Schlüsselverwendungstoken im interaktiven CloudHSM-CLI-Modus beim Signieren mit vorgehashten Daten wurde behoben.
- Es wurde ein bekanntes Problem behoben, bei dem einige von HSM gedrosselte Operationen auf hsm2m.medium nicht automatisch wiederholt wurden. Weitere Einzelheiten finden Sie unter [Problem: Während der Backup-Erstellung können Operationen fehlschlagen](#).

Version 5.16.1

Amazon Linux 2023

Laden Sie die Version 5.16.1 der Software für Amazon Linux 2023 auf der x86_64-Architektur herunter:

- [PKCS-#11-Bibliothek](#) (SHA256 checksum
342e81846436708cfc1fb459a7ef1c7b065d8a68b50a5e63653c24918d0338a0)
- [OpenSSL Dynamic Engine](#) (SHA256 checksum
4e83bebcd20201c04629a03b70136df8d225d3056346789054c4e2ce8c9b3cba)
- [JCE-Anbieter](#) (SHA256 checksum 014e3c804a56087f8855ed30e480f9c193def0e1132a4769eae3ecd235d76c4f)
- [Javadocs für AWS CloudHSM](#) (SHA256 checksum
70320f9aee48a250f9384f626fc16db00ea35c32af4c29cd63e9c82065a0fe1b)
- [CloudHSM-CLI](#) (SHA256 checksum 854c5c21f9fbef4f53a710340f213c272365c2bb1233106266eca5a5aa547e11)

Laden Sie die Version 5.16.1 der Software für Amazon Linux 2023 auf ARM64 Architektur herunter:

- [PKCS-#11-Bibliothek](#) (SHA256 checksum
dbe9498d91e1e2e8f80fc870a4a6576b0dc801dc20cb7a82bfdded79b3a362ac0)
- [OpenSSL Dynamic Engine](#) (SHA256 checksum
76cff941f36275163db146e05bf5fe64440248643ceb5dfcb3b64103f4f016a7)
- [JCE-Anbieter](#) (SHA256 checksum 8c38082797172b5630c74a8fc0c9e2652f0898bd2232b72911fbbaaa0260b5a6)
- [Javadocs für AWS CloudHSM](#) (SHA256 checksum
70320f9aee48a250f9384f626fc16db00ea35c32af4c29cd63e9c82065a0fe1b)
- [CloudHSM-CLI](#) (SHA256 checksum 2a0640e32405a7db138afb2444a542eab11c0a7d9610d5406478c7ac4602a14b)

Amazon Linux 2

Laden Sie die Version 5.16.1 der Software für Amazon Linux 2 auf der x86_64-Architektur herunter:

- [PKCS-#11-Bibliothek](#) (SHA256 checksum
cfa59458ec239553c86ac55773bebc69a5dc7cdc08a3927917fd5918e10abe93)
- [OpenSSL Dynamic Engine](#) (SHA256 checksum
49e50bece7d73f3b7dc95eed1df856a6dee40e27f24f40f015c6a4a2e8dee839)
- [JCE-Anbieter](#) (SHA256 checksum 85c8853860aa36a6c54e75c94d607a334d64dd34683ba70430ff5be61eedcc56)
- [Javadocs für AWS CloudHSM](#) (SHA256 checksum
70320f9aee48a250f9384f626fc16db00ea35c32af4c29cd63e9c82065a0fe1b)
- [CloudHSM-CLI](#) (SHA256 checksum 0efc3783e8429331aa446e45776fef46c7f9726a4768763ba4881dc2b05e090e)

Laden Sie die Version 5.16.1 der Software für Amazon Linux 2 auf ARM64 Architektur herunter:

- [PKCS-#11-Bibliothek](#) (SHA256 checksum
dad60a0380ef0ea9c469cae4de10dc47124bbecedafb6956d009734cd49abceb)
- [OpenSSL Dynamic Engine](#) (SHA256 checksum
7ee303421d94544cbe9df03022c48af327833e631d8f1ec59a466673b6e9395d)
- [JCE-Anbieter](#) (SHA256 checksum 5ebe8157abb042ee92a7edfbfcb98bf79fc3a9907684565176fd28c387c08e88)
- [Javadocs für AWS CloudHSM](#) (SHA256 checksum
70320f9aee48a250f9384f626fc16db00ea35c32af4c29cd63e9c82065a0fe1b)

- [CloudHSM-CLI](#) (SHA256 checksum 5e76b4a9021a3c92e59fc608c8263af731013835738977f376fb8ad9189add56)

RHEL 9 (9.2+)

Laden Sie die Softwareversion 5.16.1 für RHEL 9 auf der x86_64-Architektur herunter:

- [PKCS-#11-Bibliothek](#) (SHA256 checksum 696e58f74d89bd8e39f85bddc547f8c94fa2fbca498318a7bb94f22e8be7a668)
- [OpenSSL Dynamic Engine](#) (SHA256 checksum 86197cd8bdd70db91331bb8380ea094352b4087c95a04768d2cefc3bba18dffa)
- [JCE-Anbieter](#) (SHA256 checksum 37ccc2df3e8aadda74dc060f7f5bbe63e4769311f2ccbf313a8ab5d8831f206)
 - [Javadocs für AWS CloudHSM](#) (SHA256 checksum 70320f9aee48a250f9384f626fc16db00ea35c32af4c29cd63e9c82065a0fe1b)
- [CloudHSM-CLI](#) (SHA256 checksum 45a6db22db9ae64c239b49ea9281d29d2b4364dd63ee2408f463de17c4da1877)

Laden Sie die Version 5.16.1 der Software für RHEL 9 auf Architektur herunter: ARM64

- [PKCS-#11-Bibliothek](#) (SHA256 checksum fd79fbccf504f53a6e70497135fcea4e20982fca75b628124c7510dcbaddf56c)
- [OpenSSL Dynamic Engine](#) (SHA256 checksum 23fe7407be9c5f8da0ae64e560a9741887bc31936f0b88c8d1490e3c6893a8a8)
- [JCE-Anbieter](#) (SHA256 checksum bd1c43e3e28c2f71e78a41342c064ff873fd8e052f89f90227fe00eaa28f089e)
 - [Javadocs für AWS CloudHSM](#) (SHA256 checksum 70320f9aee48a250f9384f626fc16db00ea35c32af4c29cd63e9c82065a0fe1b)
- [CloudHSM-CLI](#) (SHA256 checksum 3e877a232303052b8a6b2b869dcf228edbb0da913d3d41393f622831f4455a27)

RHEL 8 (8.3+)

Laden Sie die Softwareversion 5.16.1 für RHEL 8 auf der x86_64-Architektur herunter:

- [PKCS-#11-Bibliothek](#) (SHA256 checksum d396c53c229b1eaa7cb30d4fcd17addc9170c7942795d1c82b23a157ec379686)
- [OpenSSL Dynamic Engine](#) (SHA256 checksum c01f2cb66a6c5be839906b25c6a7f7990507b1b8ec3da34c128c1a90838e48df)
- [JCE-Anbieter](#) (SHA256 checksum 1745ab78de0712111659e91301f80e771e1b7ebc2ff8c4cc5e37bce591058520)
 - [Javadocs für AWS CloudHSM](#) (SHA256 checksum 70320f9aee48a250f9384f626fc16db00ea35c32af4c29cd63e9c82065a0fe1b)

- [CloudHSM-CLI](#) (SHA256 checksum b3ca40ba66062856ef63848c71f6e9dfa0a46a2b18d44c2d96a798fc4a4fb9cf)

Ubuntu 24.04 LTS

Laden Sie die Version 5.16.1 der Software für Ubuntu 24.04 LTS auf der x86_64-Architektur herunter:

- [PKCS-#11-Bibliothek](#) (SHA256 checksum ade113ee72547cb6a8fe91a4f3ac403f462d0acd0a306f01d16ac90699f9b59c)
- [OpenSSL Dynamic Engine](#) (SHA256 checksum 9e00cb32b16fbb286a853f65c5c5154c3e805ad67820d829a0c5343f12cc9e00)
- [JCE-Anbieter](#) (SHA256 checksum 4fab322a138e14372aba394de29f444af640479dd338539be84e0cf659c8993b)
 - [Javadocs für AWS CloudHSM](#) (SHA256 checksum 70320f9aee48a250f9384f626fc16db00ea35c32af4c29cd63e9c82065a0fe1b)
- [CloudHSM-CLI](#) (SHA256 checksum f8c1e369885eb77ba4517388dab47c067a617e860fb67a028a1ebde0f96acaef)

Laden Sie die Version 5.16.1 der Software für Ubuntu 24.04 LTS auf die Architektur herunter:
ARM64

- [PKCS-#11-Bibliothek](#) (SHA256 checksum 5e15a69f8fb1429ed5c10429c91323fe82a63c0fe7bcc2f968820733d1737449)
- [OpenSSL Dynamic Engine](#) (SHA256 checksum f5136ee61bd34b74d59a01d37964b70a0f80ec5981f732b265af1c5466309e6f)
- [JCE-Anbieter](#) (SHA256 checksum acf5bebe7009461cdc1e31b95caadcbf80da09ec78d8f69142072da9baf61b79)
 - [Javadocs für AWS CloudHSM](#) (SHA256 checksum 70320f9aee48a250f9384f626fc16db00ea35c32af4c29cd63e9c82065a0fe1b)
- [CloudHSM-CLI](#) (SHA256 checksum 511376d2faf9f991f2193d08f80fc1edd53bf06daaa6fd8bd7a3fba0d6563ad1)

Ubuntu 22.04 LTS

Laden Sie die Version 5.16.1 der Software für Ubuntu 22.04 LTS auf der x86_64-Architektur herunter:

- [PKCS-#11-Bibliothek](#) (SHA256 checksum e1d787f10ee51d94732732811e4d2110f1b8e448e67fd47df8b53a2f7e56e3c8)
- [OpenSSL Dynamic Engine](#) (SHA256 checksum 29a9c41379754ce098a025feadc026f2f75a8638981f5b95ed07007d5b3d8510)
- [JCE-Anbieter](#) (SHA256 checksum 390d21e499ba181dc2774c3140433f6e60b49d9de3a892a89038c27e1aff157b)

- [Javadocs für AWS CloudHSM](#) (SHA256 checksum
70320f9aee48a250f9384f626fc16db00ea35c32af4c29cd63e9c82065a0fe1b)
- [CloudHSM-CLI](#) (SHA256 checksum cadadafaa9464f944e5c2446df92fb1777a9e1e5116cab1b359a8aa101cf7ae8)

Laden Sie die Version 5.16.1 der Software für Ubuntu 22.04 LTS auf die Architektur herunter:
ARM64

- [PKCS-#11-Bibliothek](#) (SHA256 checksum 3b6a3d50b4a55c150f12308ee3914451897bc4cc4c8420accf1d2d706316fcce)
- [OpenSSL Dynamic Engine](#) (SHA256 checksum
c366a3c5faf4de32a6dedc4613234ba3d331b0abac3241bed7d25e0109a44f64)
- [JCE-Anbieter](#) (SHA256 checksum 705416b47bc62781b5a6dfc78dd02dadb4f18f842b569c3e210b9535813b9e70)
 - [Javadocs für AWS CloudHSM](#) (SHA256 checksum
70320f9aee48a250f9384f626fc16db00ea35c32af4c29cd63e9c82065a0fe1b)
- [CloudHSM-CLI](#) (SHA256 checksum 811b501615cf34665b70f103905094ca84ed7c126f72d91b040aacda99dbf22b)

Ubuntu 20.04 LTS

Laden Sie die Version 5.16.1 der Software für Ubuntu 20.04 LTS auf der x86_64-Architektur herunter:

- [PKCS-#11-Bibliothek](#) (SHA256 checksum 6a2c33c78ada33fb435bbf8939b7cde3efe968e9f03250083dc6024b7ebd45b8)
- [OpenSSL Dynamic Engine](#) (SHA256 checksum
6ee178454e78d88e0ab92cd7c5b056d0c04cd6de192aea731d7ebcbd4c5ed761)
- [JCE-Anbieter](#) (SHA256 checksum 1e61ccd8d2b37c64467051d26e8d8bd592465ecef04282d6f4a9491707ba059d)
 - [Javadocs für AWS CloudHSM](#) (SHA256 checksum
70320f9aee48a250f9384f626fc16db00ea35c32af4c29cd63e9c82065a0fe1b)
- [CloudHSM-CLI](#) (SHA256 checksum 645693f2ede4fa0d7879eedaa41ae23120d902f24c9a637ba088f277e703cb96)

Windows Server 2025

Laden Sie die Software der Version 5.16.1 für Windows Server 2025 auf der x86_64-Architektur herunter:

- [PKCS-#11-Bibliothek](#) (SHA256 checksum
25beadaf28323412c60ed6695d89b6261d34da9cccf08869923b979854aa0329)

- [JCE-Anbieter](#) (SHA256 checksum 65f487f22c0786b80d81b387f5f4c8a6c9395c8d31a020c91fc0105829b8ec2c)
- [Javadocs für AWS CloudHSM](#) (SHA256 checksum 70320f9aee48a250f9384f626fc16db00ea35c32af4c29cd63e9c82065a0fe1b)
- [CloudHSM-CLI](#) (SHA256 checksum d4dc9f5efd2b5e2eb07a59fb237f7f8dfd5476c859cdb9f91841354ce88783c4)
- [Wichtiger Speicheranbieter \(KSP\)](#) (SHA256 checksum 80f70eb3ba22d34e49b5b5da3fa183c86c751bff46229ae16fb83a5fc69a4d0c)

Windows Server 2022

Laden Sie die Version 5.16.1 der Software für Windows Server 2022 auf der x86_64-Architektur herunter:

- [PKCS-#11-Bibliothek](#) (SHA256 checksum 25beadaf28323412c60ed6695d89b6261d34da9cccf08869923b979854aa0329)
- [JCE-Anbieter](#) (SHA256 checksum 65f487f22c0786b80d81b387f5f4c8a6c9395c8d31a020c91fc0105829b8ec2c)
- [Javadocs für AWS CloudHSM](#) (SHA256 checksum 70320f9aee48a250f9384f626fc16db00ea35c32af4c29cd63e9c82065a0fe1b)
- [CloudHSM-CLI](#) (SHA256 checksum d4dc9f5efd2b5e2eb07a59fb237f7f8dfd5476c859cdb9f91841354ce88783c4)
- [Wichtiger Speicheranbieter \(KSP\)](#) (SHA256 checksum 80f70eb3ba22d34e49b5b5da3fa183c86c751bff46229ae16fb83a5fc69a4d0c)

Windows Server 2019

Laden Sie die Version 5.16.1 der Software für Windows Server 2019 auf der x86_64-Architektur herunter:

- [PKCS-#11-Bibliothek](#) (SHA256 checksum 25beadaf28323412c60ed6695d89b6261d34da9cccf08869923b979854aa0329)
- [JCE-Anbieter](#) (SHA256 checksum 65f487f22c0786b80d81b387f5f4c8a6c9395c8d31a020c91fc0105829b8ec2c)
- [Javadocs für AWS CloudHSM](#) (SHA256 checksum 70320f9aee48a250f9384f626fc16db00ea35c32af4c29cd63e9c82065a0fe1b)
- [CloudHSM-CLI](#) (SHA256 checksum d4dc9f5efd2b5e2eb07a59fb237f7f8dfd5476c859cdb9f91841354ce88783c4)
- [Wichtiger Speicheranbieter \(KSP\)](#) (SHA256 checksum 80f70eb3ba22d34e49b5b5da3fa183c86c751bff46229ae16fb83a5fc69a4d0c)

Windows Server 2016

Laden Sie die Version 5.16.1 der Software für Windows Server 2016 auf der x86_64-Architektur herunter:

- [PKCS-#11-Bibliothek](#) (SHA256 checksum
25beadaf28323412c60ed6695d89b6261d34da9cccf08869923b979854aa0329)
- [JCE-Anbieter](#) (SHA256 checksum 65f487f22c0786b80d81b387f5f4c8a6c9395c8d31a020c91fc0105829b8ec2c)
 - [Javadocs für AWS CloudHSM](#) (SHA256 checksum
70320f9aee48a250f9384f626fc16db00ea35c32af4c29cd63e9c82065a0fe1b)
- [CloudHSM-CLI](#) (SHA256 checksum d4dc9f5efd2b5e2eb07a59fb237f7f8dfd5476c859cdb9f91841354ce88783c4)
- [Wichtiger Speicheranbieter \(KSP\)](#) (SHA256 checksum
80f70eb3ba22d34e49b5b5da3fa183c86c751bff46229ae16fb83a5fc69a4d0c)

Das Client-SDK 5.16.1 bietet Unterstützung für das Signieren und Überprüfen von vorab gehashten Daten in der CloudHSM-CLI.

Plattformunterstützung

- SDK 5.16.1 ist die letzte Version, die Unterstützung für die Ubuntu 20.04 LTS-Plattform bietet. [Weitere Informationen finden Sie auf der Ubuntu-Website.](#)

CloudHSM-CLI

- Unterstützung für das Signieren und Überprüfen von vorgehashten Daten in der CloudHSM-CLI hinzugefügt. Weitere Informationen erhalten Sie unter [Die Kategorie Kryptozeichen in CloudHSM CLI](#) und [Die Kategorie Crypto Verify in CloudHSM CLI](#).

JCE-Anbieter

- Der AES/CBC/Pkcs5Padding Verschlüsselungsmodus wurde aktualisiert, sodass automatisch ein zufälliger Initialisierungsvektor (IV) generiert wird, wenn Null IV angegeben wird. Bisher führte ein Nullwert IV zu einem Betriebsfehler. Explizite IVs Daten sind für Entschlüsselungsvorgänge weiterhin erforderlich.

Fehlerbehebungen/Verbesserungen

- Geringere Latenz bei wiederholten Digest-Aktualisierungsvorgängen in JCE.
- Der Befehl `Generate Key-Reference` wurde aktualisiert, sodass die KSP-Schlüsselreferenzdateien auf der Grundlage ihrer Attribut-ID-Werte korrekt benannt werden, wenn sie auf Windows Server ausgeführt werden. Weitere Informationen finden Sie unter [Generieren von KSP-Schlüsselreferenzen \(Windows\)](#).

Version 5.16.0

Amazon Linux 2023

Laden Sie die Version 5.16.0 der Software für Amazon Linux 2023 auf der x86_64-Architektur herunter:

- [PKCS-#11-Bibliothek](#) (SHA256 checksum d63271304f32f49838390a58d94a2140ae8a744ac64efcf0e6a65983c858d862)
- [OpenSSL Dynamic Engine](#) (SHA256 checksum 25d78c1df82355601ed6887bb47d64a06380d001da15a070cfa89dce65417fb6)
- [JCE-Anbieter](#) (SHA256 checksum fcd876813e5d114e2b042765ff98b256410e988a4e373ea1aae6f260aebdd554)
 - [Javadocs für AWS CloudHSM](#) (SHA256 checksum bbbbe99e427b7b4d51d7018a97c4d44372db020e44fdbb76fd3954fb104010e3)
- [CloudHSM-CLI](#) (SHA256 checksum 7f1441eb08daa7ceab91161900d89974ee86e0a3c54d878a29e43db775bd04f5)

Laden Sie die Softwareversion 5.16.0 für Amazon Linux 2023 auf ARM64 Architektur herunter:

- [PKCS-#11-Bibliothek](#) (SHA256 checksum 0b42e4e89cddda10c5be5f295a52d7a541bba057a3a07d3d9b192cf9eb49c776)
- [OpenSSL Dynamic Engine](#) (SHA256 checksum e8ee949d44b9f9ba64ed36eb7944acacbd1d76d7cf09d5492daee5928a9953e9)
- [JCE-Anbieter](#) (SHA256 checksum 67a6fd8f7b5dfc60ec6d6d380c31035675373b542205ec7e5a687175191cd275)
 - [Javadocs für AWS CloudHSM](#) (SHA256 checksum bbbbe99e427b7b4d51d7018a97c4d44372db020e44fdbb76fd3954fb104010e3)
- [CloudHSM-CLI](#) (SHA256 checksum 1e6c3053b19aded54045a3baa9fe2545f81a35332366cf2d77dca92ee888e654)

Amazon Linux 2

Laden Sie die Version 5.16.0 der Software für Amazon Linux 2 auf der x86_64-Architektur herunter:

- [PKCS-#11-Bibliothek](#) (SHA256 checksum 5a89ff9801f89f51e27e70869f2713cfe9bbd87d1198246cffaaf74aff9e809)
- [OpenSSL Dynamic Engine](#) (SHA256 checksum 991b6288289d07972915aa0be6dfd6b3c33c1d6312bc304225715e30832e688f)
- [JCE-Anbieter](#) (SHA256 checksum c5e673f79c2efd83195a288a5c8860c4bf74dba5bc8e422741dd5cc1be230fb3)
 - [Javadocs für AWS CloudHSM](#) (SHA256 checksum bbbbe99e427b7b4d51d7018a97c4d44372db020e44fdbb76fd3954fb104010e3)
- [CloudHSM-CLI](#) (SHA256 checksum b9021b0b348c08f433e7cce247ab1710e2a79f33a82ac36e1d6a4672171028db)

Laden Sie die Version 5.16.0 der Software für Amazon Linux 2 auf ARM64 Architektur herunter:

- [PKCS-#11-Bibliothek](#) (SHA256 checksum 97677a7f9759637b548205d31c9a08a3e7b27dfa5a38e9a4e8e2398ae9869a5a)
- [OpenSSL Dynamic Engine](#) (SHA256 checksum dde369dc4efd9f59e5c9a5459a532ad5d273f87839fba6dd5dcde2c3a2b61517)
- [JCE-Anbieter](#) (SHA256 checksum b7d96e7453e26ff07ccd9437b102f4c21857c79daa6315054728fc7ac66150bc)
 - [Javadocs für AWS CloudHSM](#) (SHA256 checksum bbbbe99e427b7b4d51d7018a97c4d44372db020e44fdbb76fd3954fb104010e3)
- [CloudHSM-CLI](#) (SHA256 checksum c7386ac016b14ea8cb96a6ed655a67f46826cdb925bd87902277b5ce9b593d51)

RHEL 9 (9.2+)

Laden Sie die Softwareversion 5.16.0 für RHEL 9 auf der x86_64-Architektur herunter:

- [PKCS-#11-Bibliothek](#) (SHA256 checksum 178d3e1ff0ec8cdc7f8c8be9aab772b4195602bb1c48dea692b628689be5ea3d)
- [OpenSSL Dynamic Engine](#) (SHA256 checksum 0488ad66cd825fafee70d1a1ccb1c8045b089af7b7d6ccadbec75f535376048c)
- [JCE-Anbieter](#) (SHA256 checksum cc9a60fa981a46c674de1e715d92ec1b8cd5c801394bfb3f0d101880589d4202)
 - [Javadocs für AWS CloudHSM](#) (SHA256 checksum bbbbe99e427b7b4d51d7018a97c4d44372db020e44fdbb76fd3954fb104010e3)

- [CloudHSM-CLI](#) (SHA256 checksum cb7e3ce6704f015d5b9462f12e6ad5489366fb3e80f554a2bdf7f04e688be1a9)

Laden Sie die Softwareversion 5.16.0 für RHEL 9 auf Architektur herunter: ARM64

- [PKCS-#11-Bibliothek](#) (SHA256 checksum 9567fb016b641821869919d4d6e9754ae08fd549381446640ff135a1e225b809)
- [OpenSSL Dynamic Engine](#) (SHA256 checksum 8c46a9dd908e77a0d8412e7050621d0c570e091845480815c230e26d1917227b)
- [JCE-Anbieter](#) (SHA256 checksum 78a7c714aedd068e5e2f12f5044a6ce1618e6dfacbcd91b4a9b505842dcc6c9)
 - [Javadocs für AWS CloudHSM](#) (SHA256 checksum bbbbe99e427b7b4d51d7018a97c4d44372db020e44fdbb76fd3954fb104010e3)
- [CloudHSM-CLI](#) (SHA256 checksum d2340ef4092ca6c1fa1ac8b484550508a5d9d8670e663bef6e595a07f8151be3)

RHEL 8 (8.3+)

Laden Sie die Softwareversion 5.16.0 für RHEL 8 auf der x86_64-Architektur herunter:

- [PKCS-#11-Bibliothek](#) (SHA256 checksum 47bf628a4a2663b8e8c7207b43e5af1d05ed2247e85e1371773596c132c9a4ec)
- [OpenSSL Dynamic Engine](#) (SHA256 checksum 49dd44243863121f7781e0f4ac556828c746d85db723864ccf16ea5fb042846a)
- [JCE-Anbieter](#) (SHA256 checksum c5e8a6664e6a8e5338a98b3845bed029f5936b517e6043aeb65ad58c3364eed8)
 - [Javadocs für AWS CloudHSM](#) (SHA256 checksum bbbbe99e427b7b4d51d7018a97c4d44372db020e44fdbb76fd3954fb104010e3)
- [CloudHSM-CLI](#) (SHA256 checksum 71f2cd6bd8892ccaea845bfe575cdf20016dd7176e5216b76290e499c822a2f2)

Ubuntu 24.04 LTS

Laden Sie die Version 5.16.0 der Software für Ubuntu 24.04 LTS auf der x86_64-Architektur herunter:

- [PKCS-#11-Bibliothek](#) (SHA256 checksum fd7e78050aba017b7dc859f0b1e6f354f56b22091cb262b328aad204064a2960)
- [OpenSSL Dynamic Engine](#) (SHA256 checksum 9d2a9e8049caa12c8027ad2963fa30f22e1032d4025e39a4e2d5985073c283f5)

- [JCE-Anbieter](#) (SHA256 checksum ae19c13f55e01538e989e8693aaf3a5d998c60f1ed71e721a34404a228947c20)
- [Javadocs für AWS CloudHSM](#) (SHA256 checksum
bbbbe99e427b7b4d51d7018a97c4d44372db020e44fdbb76fd3954fb104010e3)
- [CloudHSM-CLI](#) (SHA256 checksum 7ebedb3e8c4b8e01cc1e5dfe32ae18c9542d9c2940c77bd71ee102093aaf1d1c)

Laden Sie die Version 5.16.0 der Software für Ubuntu 24.04 LTS auf die Architektur herunter:
ARM64

- [PKCS-#11-Bibliothek](#) (SHA256 checksum
cfd9526cf495e81a2477b464ca3c124a89d07b1a366a14b39eeb27d6b5e2404b)
- [OpenSSL Dynamic Engine](#) (SHA256 checksum
02a5cc2cd9442b7bf982bb3ae44c9b134758b46e68e46fe4729b4adf1320d75c)
- [JCE-Anbieter](#) (SHA256 checksum 124d04f667a7222d624f528404a289405669f4e4a1983b8a7926758bf67e35ec)
- [Javadocs für AWS CloudHSM](#) (SHA256 checksum
bbbbe99e427b7b4d51d7018a97c4d44372db020e44fdbb76fd3954fb104010e3)
- [CloudHSM-CLI](#) (SHA256 checksum b874556b34ff5f0d082ca19f62822cf97d6e7074ce3861d7577090f99cba0cbd)

Ubuntu 22.04 LTS

Laden Sie die Version 5.16.0 der Software für Ubuntu 22.04 LTS auf der x86_64-Architektur herunter:

- [PKCS-#11-Bibliothek](#) (SHA256 checksum
e36d9bf900e195e417db261034d820cb935bec5135a0aa332d46019c70257cac)
- [OpenSSL Dynamic Engine](#) (SHA256 checksum
2c39129a5acce96b8693ce8c4b4aa174d9c13bc8ea3c2652fa505b4e0a933842)
- [JCE-Anbieter](#) (SHA256 checksum 03b70d834985a2500b73eecd831bff4f30389eb67460bf1d799466e59a9f9cfd)
- [Javadocs für AWS CloudHSM](#) (SHA256 checksum
bbbbe99e427b7b4d51d7018a97c4d44372db020e44fdbb76fd3954fb104010e3)
- [CloudHSM-CLI](#) (SHA256 checksum 3fc4fc662188ebbd430d9b5e01a57774567b926f3478d0a455fee4968c47460d)

Laden Sie die Version 5.16.0 der Software für Ubuntu 22.04 LTS auf die Architektur herunter:
ARM64

- [PKCS-#11-Bibliothek](#) (SHA256 checksum c835d9bb604b5c47a702fd45267ffb9dfa37d68dd0734391f060835ea680a169)

- [OpenSSL Dynamic Engine](#) (SHA256 checksum 943213a0f14b50fbdf3a90286e7f7ab219ab77750698f5062013873419484a04)
- [JCE-Anbieter](#) (SHA256 checksum 0ecc3e65a3b94c2dd3913f44aba58764e5558a8bedd3b742a42e4769533f3734)
 - [Javadocs für AWS CloudHSM](#) (SHA256 checksum bbbbe99e427b7b4d51d7018a97c4d44372db020e44fdbb76fd3954fb104010e3)
- [CloudHSM-CLI](#) (SHA256 checksum 8558328b224b850cf9e1b3c4f8773565a90e38ab43e2c9e90bc5b841f98898d2)

Ubuntu 20.04 LTS

Laden Sie die Version 5.16.0 der Software für Ubuntu 20.04 LTS auf der x86_64-Architektur herunter:

- [PKCS-#11-Bibliothek](#) (SHA256 checksum 64a01e84d898aca3cd03f22022cf8dbbc806a1735a84df5f820ba5bfa3339b0e)
- [OpenSSL Dynamic Engine](#) (SHA256 checksum 53a73cfc60c3fddd3ad9173bf6e602faf289f20935b3c469a137847148442954)
- [JCE-Anbieter](#) (SHA256 checksum 09544cb8129af4d06a241e1fbc0b6e6c207e1a518e5cb202f7b6c9802507d1b8)
 - [Javadocs für AWS CloudHSM](#) (SHA256 checksum bbbbe99e427b7b4d51d7018a97c4d44372db020e44fdbb76fd3954fb104010e3)
- [CloudHSM-CLI](#) (SHA256 checksum e1fdf1e014541f57d3ba4688bad88e7557e32ba974e020b4da76f85f1fc6aa29)

Windows Server 2025

Laden Sie die Software Version 5.16.0 für Windows Server 2025 auf der x86_64-Architektur herunter:

- [PKCS-#11-Bibliothek](#) (SHA256 checksum 6d3142d178127372de9ab40e7af5dc81fa7c627a18286dd521ad786d0947f17d)
- [JCE-Anbieter](#) (SHA256 checksum 9f5c28583b5127b90e753dd7ce6d081bbe4810c0c30c424192704efa6ad1be34)
 - [Javadocs für AWS CloudHSM](#) (SHA256 checksum bbbbe99e427b7b4d51d7018a97c4d44372db020e44fdbb76fd3954fb104010e3)
- [CloudHSM-CLI](#) (SHA256 checksum 22cfed533f528e0975f452af51d334baa1b5c143e722182dca107622e703b1ca)
- [Wichtiger Speicheranbieter \(KSP\)](#) (SHA256 checksum 9045c4f3e81093c4b49b93f3ea3f5caf2a4e6980628e0db2d971c437dc203bfc)

Windows Server 2022

Laden Sie die Softwareversion 5.16.0 für Windows Server 2022 auf der x86_64-Architektur herunter:

- [PKCS-#11-Bibliothek](#) (SHA256 checksum
6d3142d178127372de9ab40e7af5dc81fa7c627a18286dd521ad786d0947f17d)
- [JCE-Anbieter](#) (SHA256 checksum 9f5c28583b5127b90e753dd7ce6d081bbe4810c0c30c424192704efa6ad1be34)
 - [Javadocs für AWS CloudHSM](#) (SHA256 checksum
bbbbe99e427b7b4d51d7018a97c4d44372db020e44fdbb76fd3954fb104010e3)
- [CloudHSM-CLI](#) (SHA256 checksum 22cfed533f528e0975f452af51d334baa1b5c143e722182dca107622e703b1ca)
- [Wichtiger Speicheranbieter \(KSP\)](#) (SHA256 checksum
9045c4f3e81093c4b49b93f3ea3f5caf2a4e6980628e0db2d971c437dc203bfc)

Windows Server 2019

Laden Sie die Softwareversion 5.16.0 für Windows Server 2019 auf der x86_64-Architektur herunter:

- [PKCS-#11-Bibliothek](#) (SHA256 checksum
6d3142d178127372de9ab40e7af5dc81fa7c627a18286dd521ad786d0947f17d)
- [JCE-Anbieter](#) (SHA256 checksum 9f5c28583b5127b90e753dd7ce6d081bbe4810c0c30c424192704efa6ad1be34)
 - [Javadocs für AWS CloudHSM](#) (SHA256 checksum
bbbbe99e427b7b4d51d7018a97c4d44372db020e44fdbb76fd3954fb104010e3)
- [CloudHSM-CLI](#) (SHA256 checksum 22cfed533f528e0975f452af51d334baa1b5c143e722182dca107622e703b1ca)
- [Wichtiger Speicheranbieter \(KSP\)](#) (SHA256 checksum
9045c4f3e81093c4b49b93f3ea3f5caf2a4e6980628e0db2d971c437dc203bfc)

Windows Server 2016

Laden Sie die Softwareversion 5.16.0 für Windows Server 2016 auf der x86_64-Architektur herunter:

- [PKCS-#11-Bibliothek](#) (SHA256 checksum
6d3142d178127372de9ab40e7af5dc81fa7c627a18286dd521ad786d0947f17d)
- [JCE-Anbieter](#) (SHA256 checksum 9f5c28583b5127b90e753dd7ce6d081bbe4810c0c30c424192704efa6ad1be34)

- [Javadocs für AWS CloudHSM](#) (SHA256 checksum
bbbbe99e427b7b4d51d7018a97c4d44372db020e44fdbb76fd3954fb104010e3)
- [CloudHSM-CLI](#) (SHA256 checksum 22cfed533f528e0975f452af51d334baa1b5c143e722182dca107622e703b1ca)
- [Wichtiger Speicheranbieter \(KSP\)](#) (SHA256 checksum
9045c4f3e81093c4b49b93f3ea3f5caf2a4e6980628e0db2d971c437dc203bfc)

Das Client SDK 5.16 fügt ECDH mit X963 KDF-Unterstützung auf dem Clustertyp hsm2m.medium für den JCE-Anbieter und die PKCS #11 -Bibliothek hinzu. Das Client SDK 5.16 bietet außerdem Unterstützung für die Generierung von KSP-Schlüsselreferenzdateien auf Windows Server mit CloudHSM CLI.

CloudHSM-CLI

- Unterstützung für die Generierung von KSP-Schlüsselreferenzdateien auf Windows Server hinzugefügt. Weitere Informationen finden Sie unter. [Generieren von KSP-Schlüsselreferenzen \(Windows\)](#)

JCE-Anbieter

- ECDH mit X963 KDF-Unterstützung für den Clustertyp hsm2m.medium hinzugefügt, siehe. [Unterstützte Mechanismen für den JCE-Anbieter für AWS CloudHSM Client SDK 5](#)

PKCS-#11-Bibliothek

- ECDH mit X963 KDF-Unterstützung für den Clustertyp hsm2m.medium hinzugefügt, siehe. [Unterstützte Mechanismen für die PKCS #11 -Bibliothek für AWS CloudHSM das Client-SDK 5](#)

Fehlerbehebungen/Verbesserungen

- Es wurde ein Fehler behoben, bei dem Sitzungsschlüssel beim Trennen der Verbindung nicht richtig gereinigt wurden.
- Es wurde ein Problem behoben, bei dem erfolgreiche mTLS-Abmeldevorgänge auf hsm2m.medium fälschlicherweise zu einer Fehlerantwort führten.
- Ein Problem mit einem Verbindungsfehler beim Windows-Start wurde behoben.
- Es wurde ein Problem behoben, bei dem SDK-Info-Tools auf Ubuntu-Systemen eine falsche Ausgabe anzeigten.

- Es wurde ein Problem behoben, das zuvor ECDSA mit schwächeren Hashes im FIPS-Modus zuließ.

Version 5.15.0

Amazon Linux 2023

Laden Sie die Version 5.15.0 der Software für Amazon Linux 2023 auf der x86_64-Architektur herunter:

- [PKCS-#11-Bibliothek](#) (SHA256 checksum 41ef3178811df1dbb03b2cbac83fe0f4768bdc9b17005c409f1c0229f93ef11c)
- [OpenSSL Dynamic Engine](#) (SHA256 checksum
afa1f9f8bd99f54866dea1b8928c00b951a6e492f5f36d0d6c7c38fff341d609)
- [JCE-Anbieter](#) (SHA256 checksum 6ea775e05570ef3497a4df5c35a6ec1c682aea73c48e7fec3e541af995759e)
 - [Javadocs für AWS CloudHSM](#) (SHA256 checksum
ce5d92731f2a9c7f46c92bba92f3d0adff4ceb06b9d653597994100dfc352fbe)
- [CloudHSM-CLI](#) (SHA256 checksum 78c10bb213dd14fcfc5836e358de6aedac61db05125fd61137b0082214fdecbe)

Laden Sie die Version 5.15.0 der Software für Amazon Linux 2023 auf ARM64 Architektur herunter:

- [PKCS-#11-Bibliothek](#) (SHA256 checksum cd0617d29b6d64c00c8e14fd9a92f604e14acda4746f4f0b86b9de42367192fb)
- [OpenSSL Dynamic Engine](#) (SHA256 checksum
bc9acfd04eb1246eb3d5b0a8f3736ec017c0d1699d5395f85868d4a1722cd83)
- [JCE-Anbieter](#) (SHA256 checksum 96d05301486206577b7be05aac561649167b57272c9c06ff839fd8ff2b5d96d5)
 - [Javadocs für AWS CloudHSM](#) (SHA256 checksum
ce5d92731f2a9c7f46c92bba92f3d0adff4ceb06b9d653597994100dfc352fbe)
- [CloudHSM-CLI](#) (SHA256 checksum b870eadb27736a2cde98022d57e9704c67ae15878cf0b910738859cdabaa35a2)

Amazon Linux 2

Laden Sie die Version 5.15.0 der Software für Amazon Linux 2 auf der x86_64-Architektur herunter:

- [PKCS-#11-Bibliothek](#) (SHA256 checksum
c70ae4f0181a8187c9380481c51c1d03e12236dd86863ec818ed3f210b294c8e)

- [OpenSSL Dynamic Engine](#) (SHA256 checksum 08e9fd1dd80efa637f9a1727bb0de205ba124a3776b2e8bc21008ee458063a42)
- [JCE-Anbieter](#) (SHA256 checksum 7932ed060e72c53b2556f30694b0ffe5342b244b6628c7a9dc03966aa49c8fe6)
 - [Javadocs für AWS CloudHSM](#) (SHA256 checksum ce5d92731f2a9c7f46c92bba92f3d0adff4ceb06b9d653597994100dfc352fbe)
- [CloudHSM-CLI](#) (SHA256 checksum 0a939e4d5d0a2ff308c7a1d9e73ebc865e426214d556bde1bd29dbe807fbb583)

Laden Sie die Version 5.15.0 der Software für Amazon Linux 2 auf ARM64 Architektur herunter:

- [PKCS-#11-Bibliothek](#) (SHA256 checksum bcf907a8fb4722b09c54b8e5785fe2b8cffcedd6ee3fdda2d21879a012138077)
- [OpenSSL Dynamic Engine](#) (SHA256 checksum 8f70edc3a6a4a1bf0264c6567b1dda1ac69055f206753f88eeadb8b8bf9f38)
- [JCE-Anbieter](#) (SHA256 checksum 75dd67736bb08fe7e46e113af10803a255ba8edee3016ca963c1ee94fe59d43b)
 - [Javadocs für AWS CloudHSM](#) (SHA256 checksum ce5d92731f2a9c7f46c92bba92f3d0adff4ceb06b9d653597994100dfc352fbe)
- [CloudHSM-CLI](#) (SHA256 checksum 1c2ecf90c955281d99bcd8d1956d63debb15bbc8419744c83f88821ef8b78aee)

RHEL 9 (9.2+)

Laden Sie die Softwareversion 5.15.0 für RHEL 9 auf der x86_64-Architektur herunter:

- [PKCS-#11-Bibliothek](#) (SHA256 checksum 65bd0b815eebc806674a7bf7c54e9f884595881547f5fffd08ff6a38aabdccebe)
- [OpenSSL Dynamic Engine](#) (SHA256 checksum f2af9f5882ab2e5a11defecc660f8af5c4d9d6e2e2b89873e6833fc2976f44ac)
- [JCE-Anbieter](#) (SHA256 checksum 5124bace5d1544775c891f13a0e309b30dd73699116c46ecc9a77bba5f9cf633)
 - [Javadocs für AWS CloudHSM](#) (SHA256 checksum ce5d92731f2a9c7f46c92bba92f3d0adff4ceb06b9d653597994100dfc352fbe)
- [CloudHSM-CLI](#) (SHA256 checksum 1eec31fb2c0ab3c2839d9bdb37874cf8dbc74a383279068fce9e8613966d06e0)

Laden Sie die Softwareversion 5.15.0 für RHEL 9 auf Architektur herunter: ARM64

- [PKCS-#11-Bibliothek](#) (SHA256 checksum f498697519afa04b6b0362a4388a6a38e2cb6813b781d6ce3d97a7de89c5cbfe)
- [OpenSSL Dynamic Engine](#) (SHA256 checksum 5410da63108a1b209e567e9bccf8bd7e4035af88b0d58b9d78b10917be1b40c1)

- [JCE-Anbieter](#) (SHA256 checksum b46f233e6994d2c0ed505dc5c717ee3009daaeb2063d260aa78e06273770bffa)
- [Javadocs für AWS CloudHSM](#) (SHA256 checksum ce5d92731f2a9c7f46c92bba92f3d0adff4ceb06b9d653597994100dfc352fbe)
- [CloudHSM-CLI](#) (SHA256 checksum e9f93eaa58db2f7ea1174164ef96ab219700933d353243c3c6ab1aebac5ccffe)

RHEL 8 (8.3+)

Laden Sie die Softwareversion 5.15.0 für RHEL 8 auf der x86_64-Architektur herunter:

- [PKCS-#11-Bibliothek](#) (SHA256 checksum 87131e179d0e60ade302ec07b22803cfb39294bf060b786c41f154d95791ac94)
- [OpenSSL Dynamic Engine](#) (SHA256 checksum f412a2f5cd761db5940288bb252ce060d44735c6b436bb6d4fa7d3687a44a026)
- [JCE-Anbieter](#) (SHA256 checksum d0844f55c08f9ff9c393138a9041efe1b59dc3dce20c0b2c23406efe6acc43db)
- [Javadocs für AWS CloudHSM](#) (SHA256 checksum ce5d92731f2a9c7f46c92bba92f3d0adff4ceb06b9d653597994100dfc352fbe)
- [CloudHSM-CLI](#) (SHA256 checksum 9fd8033e478ce6d7d640c063c4f007359cb04c19d519826a745ad0885f96a0f8)

Ubuntu 24.04 LTS

Laden Sie die Version 5.15.0 der Software für Ubuntu 24.04 LTS auf der x86_64-Architektur herunter:

- [PKCS-#11-Bibliothek](#) (SHA256 checksum ca5f2f80ae921cfdbdc5c8bc35c39d2b19cfabfd5981932409eaf2e7c00a9097)
- [OpenSSL Dynamic Engine](#) (SHA256 checksum e44cd7b678a421957c84e4fc0f70280360fd4e1e66f4cabdd1b20b955ee5fcca)
- [JCE-Anbieter](#) (SHA256 checksum bc6382f12769c3d87c34522d0d616b7a8c108574c003814e1300938c386655ac)
- [Javadocs für AWS CloudHSM](#) (SHA256 checksum ce5d92731f2a9c7f46c92bba92f3d0adff4ceb06b9d653597994100dfc352fbe)
- [CloudHSM-CLI](#) (SHA256 checksum f786be86c204680e670f7817376ff376733ebd96b51c31e9c98213078596637f)

Laden Sie die Version 5.15.0 der Software für Ubuntu 24.04 LTS auf die Architektur herunter:
ARM64

- [PKCS-#11-Bibliothek](#) (SHA256 checksum
1ecf057d67137c0e9e1bb0dda57d581690c647b0360c3a617a8dc668919d08de)
- [OpenSSL Dynamic Engine](#) (SHA256 checksum
4f3d23e6b798f88be587108d8e6a225d796a3d080aef61ea384eb74a1270612a)
- [JCE-Anbieter](#) (SHA256 checksum b9fd16bdcc1fcf59fde0d3e0debee500b0b7edfdff69209e84d14393097fe9d2)
 - [Javadocs für AWS CloudHSM](#) (SHA256 checksum
ce5d92731f2a9c7f46c92bba92f3d0adff4ceb06b9d653597994100dfc352fbe)
- [CloudHSM-CLI](#) (SHA256 checksum a95922d9b44e64a64723db0c21ac89566515a5a6c87de990af4a9e1f40c7424f)

Ubuntu 22.04 LTS

Laden Sie die Version 5.15.0 der Software für Ubuntu 22.04 LTS auf der x86_64-Architektur herunter:

- [PKCS-#11-Bibliothek](#) (SHA256 checksum
966be12eb32de813ca07e766abf7b5616c0d2e105e9296d920aadaca10e5afdf)
- [OpenSSL Dynamic Engine](#) (SHA256 checksum
f2840151d87b7f9cbff68993c25397afd48a16f054abf0f2fd4624662d3087d6)
- [JCE-Anbieter](#) (SHA256 checksum 01aebdda05640e50a82cae04c5cb6d33ab909dadd917bd834957d4f75ad8c577)
 - [Javadocs für AWS CloudHSM](#) (SHA256 checksum
ce5d92731f2a9c7f46c92bba92f3d0adff4ceb06b9d653597994100dfc352fbe)
- [CloudHSM-CLI](#) (SHA256 checksum 2e5eba9409abc429828505779c512cd2424719766c0e488f50aee288966cf61d)

Laden Sie die Version 5.15.0 der Software für Ubuntu 22.04 LTS auf die Architektur herunter:
ARM64

- [PKCS-#11-Bibliothek](#) (SHA256 checksum 614f74f101c9c64fff515128c5a59fa23de5047494fe248e8aabdb441092b3e)
- [OpenSSL Dynamic Engine](#) (SHA256 checksum
c15d6db77b76bce690749b73b947567eb5f2d76669887843116d0ce56c1f8ea7)
- [JCE-Anbieter](#) (SHA256 checksum bca8511d5c0a173b0fef326016ce5091b6e6829fa2b3cb45ed5621290ae3e42a)
 - [Javadocs für AWS CloudHSM](#) (SHA256 checksum
ce5d92731f2a9c7f46c92bba92f3d0adff4ceb06b9d653597994100dfc352fbe)
- [CloudHSM-CLI](#) (SHA256 checksum 2f5b673148b682d7e34619c51b5e8799abe7dc7fd4f046158a0d05320ba24dc1)

Ubuntu 20.04 LTS

Laden Sie die Version 5.15.0 der Software für Ubuntu 20.04 LTS auf der x86_64-Architektur herunter:

- [PKCS-#11-Bibliothek](#) (SHA256 checksum 86c8394b5ddff91a71194fb87c327efde36baa2380e559c04f9d543a6e74d61b)
- [OpenSSL Dynamic Engine](#) (SHA256 checksum 0a4227389fea61e6e7ac7cfa715eb341f7a4eeae9ed10e4c96da2c0dd4a18f9e)
- [JCE-Anbieter](#) (SHA256 checksum 65b2d926ff9dfbe6c7864bc3a41b3da2383bc731dd199bcef8805a0543fbe612)
 - [Javadocs für AWS CloudHSM](#) (SHA256 checksum ce5d92731f2a9c7f46c92bba92f3d0adff4ceb06b9d653597994100dfc352fbe)
- [CloudHSM-CLI](#) (SHA256 checksum 09c2e55bcf72f9e530717950d8c5fd48574ae6ccb09a049526ca5b2a3b8aa9)

Windows Server 2025

Laden Sie die Software Version 5.15.0 für Windows Server 2025 auf der x86_64-Architektur herunter:

- [PKCS-#11-Bibliothek](#) (SHA256 checksum a903b63fe286f15bf669c0555b1fa4d86b33592ed05af0809acac28c0d3ace16)
- [JCE-Anbieter](#) (SHA256 checksum fdef6251f06d77d51fddbc2184d3eec87ddec4fe35b3ac620343eb66c95ddf64)
 - [Javadocs für AWS CloudHSM](#) (SHA256 checksum ce5d92731f2a9c7f46c92bba92f3d0adff4ceb06b9d653597994100dfc352fbe)
- [CloudHSM-CLI](#) (SHA256 checksum 6cc9205fe1fc514ddd9774f824e512940d6b1bc4cc0e251265bc46ab99746c28)
- [Wichtiger Speicheranbieter \(KSP\)](#) (SHA256 checksum 52ed9b08cd0ce100b8dcd3d8e8f411b6201f9f1b27872b19d1136c0bf36a29b8)

Windows Server 2022

Laden Sie die Softwareversion 5.15.0 für Windows Server 2022 auf der x86_64-Architektur herunter:

- [PKCS-#11-Bibliothek](#) (SHA256 checksum a903b63fe286f15bf669c0555b1fa4d86b33592ed05af0809acac28c0d3ace16)
- [JCE-Anbieter](#) (SHA256 checksum fdef6251f06d77d51fddbc2184d3eec87ddec4fe35b3ac620343eb66c95ddf64)
 - [Javadocs für AWS CloudHSM](#) (SHA256 checksum ce5d92731f2a9c7f46c92bba92f3d0adff4ceb06b9d653597994100dfc352fbe)
- [CloudHSM-CLI](#) (SHA256 checksum 6cc9205fe1fc514ddd9774f824e512940d6b1bc4cc0e251265bc46ab99746c28)

- [Wichtiger Speicheranbieter \(KSP\)](#) (SHA256 checksum
52ed9b08cd0ce100b8dcd3d8e8f411b6201f9f1b27872b19d1136c0bf36a29b8)

Windows Server 2019

Laden Sie die Softwareversion 5.15.0 für Windows Server 2019 auf der x86_64-Architektur herunter:

- [PKCS-#11-Bibliothek](#) (SHA256 checksum a903b63fe286f15bf669c0555b1fa4d86b33592ed05af0809acac28c0d3ace16)
- [JCE-Anbieter](#) (SHA256 checksum fdef6251f06d77d51fddbc2184d3eec87ddec4fe35b3ac620343eb66c95ddf64)
- [Javadocs für AWS CloudHSM](#) (SHA256 checksum
ce5d92731f2a9c7f46c92bba92f3d0adff4ceb06b9d653597994100dfc352fbe)
- [CloudHSM-CLI](#) (SHA256 checksum 6cc9205fe1fc514ddd9774f824e512940d6b1bc4cc0e251265bc46ab99746c28)
- [Wichtiger Speicheranbieter \(KSP\)](#) (SHA256 checksum
52ed9b08cd0ce100b8dcd3d8e8f411b6201f9f1b27872b19d1136c0bf36a29b8)

Windows Server 2016

Laden Sie die Softwareversion 5.15.0 für Windows Server 2016 auf der x86_64-Architektur herunter:

- [PKCS-#11-Bibliothek](#) (SHA256 checksum a903b63fe286f15bf669c0555b1fa4d86b33592ed05af0809acac28c0d3ace16)
- [JCE-Anbieter](#) (SHA256 checksum fdef6251f06d77d51fddbc2184d3eec87ddec4fe35b3ac620343eb66c95ddf64)
- [Javadocs für AWS CloudHSM](#) (SHA256 checksum
ce5d92731f2a9c7f46c92bba92f3d0adff4ceb06b9d653597994100dfc352fbe)
- [CloudHSM-CLI](#) (SHA256 checksum 6cc9205fe1fc514ddd9774f824e512940d6b1bc4cc0e251265bc46ab99746c28)
- [Wichtiger Speicheranbieter \(KSP\)](#) (SHA256 checksum
52ed9b08cd0ce100b8dcd3d8e8f411b6201f9f1b27872b19d1136c0bf36a29b8)

Das Client SDK 5.15 bietet Unterstützung für die Replikation eines Benutzers über geklonte Cluster mit CloudHSM CLI. Das Client-SDK 5.15 fügt außerdem Installationspakete für die PKCS #11 - Bibliothek, den JCE-Anbieter, die CloudHSM-CLI und den Key Storage Provider (KSP) für Windows Server 2025 hinzu.

Plattformunterstützung

- Unterstützung für Windows Server 2025 für die PKCS #11 -Bibliothek, den JCE-Anbieter, die CloudHSM-CLI und den Key Storage Provider (KSP) hinzugefügt.

CloudHSM-CLI

- Der folgende Befehl wurde hinzugefügt:
 - Replizieren Sie einen Benutzer mit CloudHSM CLI

Version 5.14.0

Amazon Linux 2023

Laden Sie die Version 5.14.0 der Software für Amazon Linux 2023 auf der x86_64-Architektur herunter:

- [PKCS-#11-Bibliothek](#) (SHA256 checksum
05e7a3882166c694a7a09bc735f08f91c8145a4215176665eacacdf3e509abe8)
- [OpenSSL Dynamic Engine](#) (SHA256 checksum
f4dd9966988418e100c276dc0d521f91afdfc0e6c008dbf8eda446ebaca83c14)
- [JCE-Anbieter](#) (SHA256 checksum c4dee5c1173f6a1c7683aedb58e61d329c36933435c416f288d94bc9a68a6b31)
 - [Javadocs für AWS CloudHSM](#) (SHA256 checksum
173087a8f286dc4f88dc915ecc00aa515bec0ae2faf219654df9f3422d8e83bb)
- [CloudHSM-CLI](#) (SHA256 checksum 1fad069fde305450254287e43750d597db9af0cb8fd168300cd5eaed9e2af33a)

Laden Sie die Softwareversion 5.14.0 für Amazon Linux 2023 auf ARM64 Architektur herunter:

- [PKCS-#11-Bibliothek](#) (SHA256 checksum 9f34163d02bce26c1280e589310cda891d27995c50cec0a7fe083100ecff2b69)
- [OpenSSL Dynamic Engine](#) (SHA256 checksum
7444a4daad6e4715c82d6c39a7b03a07ee0201a13fe0f98da96acbf9d24abf6c)
- [JCE-Anbieter](#) (SHA256 checksum 3393fe3a0f5c3a9c92106d74b7de074c818e095f97d2cfd600dbd47779b90b37)
 - [Javadocs für AWS CloudHSM](#) (SHA256 checksum
173087a8f286dc4f88dc915ecc00aa515bec0ae2faf219654df9f3422d8e83bb)
- [CloudHSM-CLI](#) (SHA256 checksum d9a2edce48c5f6646d5a351cb431712f2d2fc62d21f8318e7bb1ce579819d7f4)

Amazon Linux 2

Laden Sie die Version 5.14.0 der Software für Amazon Linux 2 auf der x86_64-Architektur herunter:

- [PKCS-#11-Bibliothek](#) (SHA256 checksum 9c47b90bfa0ad51627cdb0dd8f148a56090fbdeb2490f1ab4009170c7b9c1120)
- [OpenSSL Dynamic Engine](#) (SHA256 checksum 215f9768331565085a317585b3dbe0514b251fdc428c96ed32491c4abb9fea56)
- [JCE-Anbieter](#) (SHA256 checksum a802f941e95fcbf0ef37775fc096c0d6ae4c916fa08330a9d56defc5f99ff2b7)
- [Javadocs für AWS CloudHSM](#) (SHA256 checksum 173087a8f286dc4f88dc915ecc00aa515bec0ae2faf219654df9f3422d8e83bb)
- [CloudHSM-CLI](#) (SHA256 checksum ac10f23dd81264b1d10a0760f62b5006d2b2b66bf1c6378248ca9326afb65a83)

Laden Sie die Version 5.14.0 der Software für Amazon Linux 2 auf ARM64 Architektur herunter:

- [PKCS-#11-Bibliothek](#) (SHA256 checksum f2d1550f043c655ef5d51e2ca14a0416886d99902e0702bd7f30f93b2c563d4d)
- [OpenSSL Dynamic Engine](#) (SHA256 checksum 2c8e2c81af53ba3646d1f947894d84b9b54780bbca79b58f354125e4ac9427c0)
- [JCE-Anbieter](#) (SHA256 checksum 3f9c881056e6905d46358585db72143b59971958a708fa4ac75cb53994487213)
- [Javadocs für AWS CloudHSM](#) (SHA256 checksum 173087a8f286dc4f88dc915ecc00aa515bec0ae2faf219654df9f3422d8e83bb)
- [CloudHSM-CLI](#) (SHA256 checksum 6d5985e44c9852409dd3d342239fbbf7d0f9ad43d445e0980153c9b9eafe2b6f)

RHEL 9 (9.2+)

Laden Sie die Softwareversion 5.14.0 für RHEL 9 auf der x86_64-Architektur herunter:

- [PKCS-#11-Bibliothek](#) (SHA256 checksum c4812210421ff2fa5dac8477a2e6b10552aabd88b1f03d717e044e7293823158)
- [OpenSSL Dynamic Engine](#) (SHA256 checksum 3f6aeaa6ae1faae7d8bba1596f358cad1eec9e562cf08aaab9ded92cabe94719)
- [JCE-Anbieter](#) (SHA256 checksum bd8f120b08f738ad4d1534b0e32aad903758d5e75d3aba7cb9ff6a77dec533db)
- [Javadocs für AWS CloudHSM](#) (SHA256 checksum 173087a8f286dc4f88dc915ecc00aa515bec0ae2faf219654df9f3422d8e83bb)
- [CloudHSM-CLI](#) (SHA256 checksum 84f13a9babd767edf90dabb0b14034ec5b2208898123a13966dd6b8519961c27)

Laden Sie die Softwareversion 5.14.0 für RHEL 9 auf Architektur herunter: ARM64

- [PKCS-#11-Bibliothek](#) (SHA256 checksum
3faa1de563aa9773939cab7b39907dcb7b981ed7225019f9070a9dd52db7ae70)
- [OpenSSL Dynamic Engine](#) (SHA256 checksum
13b41ac47ef7ee7bf78f585b8347ca4da9ebc296e4fc1e6a0c2ff5b333354ca6)
- [JCE-Anbieter](#) (SHA256 checksum 06803655ebe54d59c180bb17ac6fe56337bdd58ad0f6fe87c50ff8df32f70258)
 - [Javadocs für AWS CloudHSM](#) (SHA256 checksum
173087a8f286dc4f88dc915ecc00aa515bec0ae2faf219654df9f3422d8e83bb)
- [CloudHSM-CLI](#) (SHA256 checksum e6b4e9688d0db9d72bbee3450fe19736d640c9931adbd4f4ef73cb7ac2a08cf4)

RHEL 8 (8.3+)

Laden Sie die Softwareversion 5.14.0 für RHEL 8 auf der x86_64-Architektur herunter:

- [PKCS-#11-Bibliothek](#) (SHA256 checksum e400aeea6dbf7721f97e71c643a0db4f5f1094fb197fc46dc0ab293de9d16f2d)
- [OpenSSL Dynamic Engine](#) (SHA256 checksum
bfcc27d251e62f9eba0fd508e7d08dc62126642d4cdd0b5566183957768b8c54)
- [JCE-Anbieter](#) (SHA256 checksum b5500031b572c918a8df4a0347e01c8ea00a7366b865b310bd92427fa1ed53e3)
 - [Javadocs für AWS CloudHSM](#) (SHA256 checksum
173087a8f286dc4f88dc915ecc00aa515bec0ae2faf219654df9f3422d8e83bb)
- [CloudHSM-CLI](#) (SHA256 checksum 000712d0a691efc64a6c5d54bfef1ab48b315ebb5dd6926b0502e326bf700291)

Ubuntu 24.04 LTS

Laden Sie die Version 5.14.0 der Software für Ubuntu 24.04 LTS auf der x86_64-Architektur herunter:

- [PKCS-#11-Bibliothek](#) (SHA256 checksum
bbec70a198a4b173620b4018accc297ce6a6a80d372706e0101997d6bca35bca)
- [OpenSSL Dynamic Engine](#) (SHA256 checksum
fcb77f75bd465b22401a09a20c410985833340295101263b7171cdcc4ac9f980)
- [JCE-Anbieter](#) (SHA256 checksum e1fa16aae2f6095c89a8bf392ef2cb9ca4db8853c858904ff90abf4bb491d74b)
 - [Javadocs für AWS CloudHSM](#) (SHA256 checksum
173087a8f286dc4f88dc915ecc00aa515bec0ae2faf219654df9f3422d8e83bb)

- [CloudHSM-CLI](#) (SHA256 checksum af32e0ac1c5f8c7f8cb40d255abe7c63ce9d981293187a3fb452562fa05756f4)

Laden Sie die Version 5.14.0 der Software für Ubuntu 24.04 LTS auf die Architektur herunter:
ARM64

- [PKCS-#11-Bibliothek](#) (SHA256 checksum 1232318b084347889f92136536537ca519373683ce39bcec20f6ac3fa2f42d7c)
- [OpenSSL Dynamic Engine](#) (SHA256 checksum 1745ee3a33d8e6ea72644e903c55c6a206204cf0a8bea200bc4a7b15736ed801)
- [JCE-Anbieter](#) (SHA256 checksum 7a44acabb9c90c996594ed53661937f9242850823347f7c386f02fc041a97471a)
 - [Javadocs für AWS CloudHSM](#) (SHA256 checksum 173087a8f286dc4f88dc915ecc00aa515bec0ae2faf219654df9f3422d8e83bb)
- [CloudHSM-CLI](#) (SHA256 checksum e6778bd12c55fd152b50033833531fe569472f4f2bd9927a345eb126e8305739)

Ubuntu 22.04 LTS

Laden Sie die Version 5.14.0 der Software für Ubuntu 22.04 LTS auf der x86_64-Architektur herunter:

- [PKCS-#11-Bibliothek](#) (SHA256 checksum 6b4b1620e9a85267950633b171dd188b7ac7094e371e188fab1bef7a911a16f)
- [OpenSSL Dynamic Engine](#) (SHA256 checksum fee5f0a65fab0f46ad58689af5dc510721581f31364d3be5cbbf79f5d9a60db8)
- [JCE-Anbieter](#) (SHA256 checksum 5883a3d15e160d65f8e26f185e1ee30f68becad5f6fcd16abc1c4586689800b5)
 - [Javadocs für AWS CloudHSM](#) (SHA256 checksum 173087a8f286dc4f88dc915ecc00aa515bec0ae2faf219654df9f3422d8e83bb)
- [CloudHSM-CLI](#) (SHA256 checksum 7b1a8cd2962f4be7325154c080dabfb820beac5629e272c3a3ebc0e6cab11d27)

Laden Sie die Version 5.14.0 der Software für Ubuntu 22.04 LTS auf die Architektur herunter:
ARM64

- [PKCS-#11-Bibliothek](#) (SHA256 checksum 8680f63fd74272ea0e4d4c32dfb96b3eb9b60e03483c202d4e9b65ee101a178f)
- [OpenSSL Dynamic Engine](#) (SHA256 checksum c757304e8fc5f38be3bab7ac6d37a35dcb56d31e62f719da62b3a176593d1d8)
- [JCE-Anbieter](#) (SHA256 checksum a78ee59341da56af315d126ec7ed9f0dfe6e99649f659db2436be3863cb035b)

- [Javadocs für AWS CloudHSM](#) (SHA256 checksum 173087a8f286dc4f88dc915ecc00aa515bec0ae2faf219654df9f3422d8e83bb)
- [CloudHSM-CLI](#) (SHA256 checksum 18931bff869a0bd54846b3296d870fa19beba5652f979352be8fba6307e6d1aa)

Ubuntu 20.04 LTS

Laden Sie die Version 5.14.0 der Software für Ubuntu 20.04 LTS auf der x86_64-Architektur herunter:

- [PKCS-#11-Bibliothek](#) (SHA256 checksum f1461c16b135ebcc17deec46aab88bd113ea122b8942fc188d4f05cd03e919a8)
- [OpenSSL Dynamic Engine](#) (SHA256 checksum 89211a7a7ed50eda2dc385c31ea76f1fbabd389ca691204873531d983c3eb0f7)
- [JCE-Anbieter](#) (SHA256 checksum f098c32d61a53b073459a75d88b68e377a9b16335874fae060cb10df0da00df0)
 - [Javadocs für AWS CloudHSM](#) (SHA256 checksum 173087a8f286dc4f88dc915ecc00aa515bec0ae2faf219654df9f3422d8e83bb)
- [CloudHSM-CLI](#) (SHA256 checksum ff91fb930717c917344af2ba344dc6e02bd5abc004dcb6147e9412b67e2aa7ab)

Windows Server 2022

Laden Sie die Version 5.14.0 der Software für Windows Server 2022 auf der x86_64-Architektur herunter:

- [PKCS-#11-Bibliothek](#) (SHA256 checksum ee5a8e8e85fff7acd0bbafa23740e3981f7dc52e708972b600c2b26603786838)
- [JCE-Anbieter](#) (SHA256 checksum 2ae0274f09f66981c03fd1e3c264e896ba7cd211168ea31369335db1b3ea2e77)
 - [Javadocs für AWS CloudHSM](#) (SHA256 checksum 173087a8f286dc4f88dc915ecc00aa515bec0ae2faf219654df9f3422d8e83bb)
- [CloudHSM-CLI](#) (SHA256 checksum 3938654f88ce010042e48909e23991e178e411e2bd9e3c9ec25fcb8157c0cd55)
- [Wichtiger Speicheranbieter \(KSP\)](#) (SHA256 checksum b026e4d8c11e9ff6f22a7b9e10b8bb29e7572665f0d7978a3cef7d2354b7693f)

Windows Server 2019

Laden Sie die Softwareversion 5.14.0 für Windows Server 2019 auf der x86_64-Architektur herunter:

- [PKCS-#11-Bibliothek](#) (SHA256 checksum ee5a8e8e85fff7acd0bbafa23740e3981f7dc52e708972b600c2b26603786838)
- [JCE-Anbieter](#) (SHA256 checksum 2ae0274f09f66981c03fd1e3c264e896ba7cd211168ea31369335db1b3ea2e77)
 - [Javadocs für AWS CloudHSM](#) (SHA256 checksum 173087a8f286dc4f88dc915ecc00aa515bec0ae2faf219654df9f3422d8e83bb)
- [CloudHSM-CLI](#) (SHA256 checksum 3938654f88ce010042e48909e23991e178e411e2bd9e3c9ec25fcb8157c0cd55)
- [Wichtiger Speicheranbieter \(KSP\)](#) (SHA256 checksum b026e4d8c11e9ff6f22a7b9e10b8bb29e7572665f0d7978a3cef7d2354b7693f)

Windows Server 2016

Laden Sie die Softwareversion 5.14.0 für Windows Server 2016 auf der x86_64-Architektur herunter:

- [PKCS-#11-Bibliothek](#) (SHA256 checksum ee5a8e8e85fff7acd0bbafa23740e3981f7dc52e708972b600c2b26603786838)
- [JCE-Anbieter](#) (SHA256 checksum 2ae0274f09f66981c03fd1e3c264e896ba7cd211168ea31369335db1b3ea2e77)
 - [Javadocs für AWS CloudHSM](#) (SHA256 checksum 173087a8f286dc4f88dc915ecc00aa515bec0ae2faf219654df9f3422d8e83bb)
- [CloudHSM-CLI](#) (SHA256 checksum 3938654f88ce010042e48909e23991e178e411e2bd9e3c9ec25fcb8157c0cd55)
- [Wichtiger Speicheranbieter \(KSP\)](#) (SHA256 checksum b026e4d8c11e9ff6f22a7b9e10b8bb29e7572665f0d7978a3cef7d2354b7693f)

Das Client-SDK 5.14 bietet Unterstützung für quorumgesteuerte Schlüsselerwendungs- und Schlüsselerwaltungsvorgänge mithilfe der CloudHSM-CLI. Das Client-SDK 5.14 bietet auch Unterstützung für Windows-Plattformen. [Schlüsselspeicheranbieter \(KSP\) für AWS CloudHSM Client SDK 5](#) Darüber hinaus fügt das Client SDK 5.14 Installationspakete für die PKCS #11 -Bibliothek, den JCE-Anbieter, die CloudHSM-CLI und den Key Storage Provider (KSP) für Windows Server 2022 hinzu.

Plattformunterstützung

- Unterstützung für Windows Server 2022 für die PKCS #11 -Bibliothek, den JCE-Anbieter, die CloudHSM-CLI und den Key Storage Provider (KSP) hinzugefügt.

CloudHSM-CLI

- Unterstützung für quorumgesteuerte Schlüsselnverwendungs- und Schlüsselverwaltungsvorgänge wurde hinzugefügt.

Key Storage Provider (KSP)

- Unterstützung für Key Storage Provider (KSP) hinzugefügt, eine kryptografische API speziell für das Microsoft Windows-Betriebssystem. Weitere Informationen finden Sie unter [Schlüsselspeicheranbieter \(KSP\) für AWS CloudHSM Client SDK 5](#).

Version 5.13.0

Das Client-SDK 5.13 bietet Unterstützung für die Konfiguration von Mutual TLS auf hsm2m.medium-Clustertypen. Informationen zur Verwendung von Mutual TLS mit CloudHSM finden Sie unter [Richten Sie gegenseitiges TLS zwischen Client und AWS CloudHSM \(empfohlen\) ein](#). Das Client SDK 5.13 fügt auch Installationspakete für Ubuntu 24.04 LTS hinzu.

Plattformunterstützung

- Unterstützung für Ubuntu 24.04 LTS auf x86_64 und Architekturen für alle hinzugefügt. ARM64 SDKs

CloudHSM-CLI

- Es wurde Unterstützung für Admin-Benutzer hinzugefügt, um den Befehl auszuführen. [Einen Schlüssel mit CloudHSM CLI replizieren](#) Mit dem Client SDK 5.12 wurde der Befehl key replicate eingeführt, der von Krypto-Benutzern verwendet werden kann.
- Der folgende Befehl wurde hinzugefügt:
 - [Die Cluster-MTLS-Kategorie in der CloudHSM-CLI](#)

Fehlerbehebungen/Verbesserungen

- Es wurde ein Problem behoben, das die Zeit verringert, die der Client benötigt, um fehlerhafte HSM-Verbindungen zu erkennen, wodurch Verbindungsabbrüche bei Lambda-Warmstarts vermieden werden.

Version 5.12.0

Das Client SDK 5.12.0 bietet ARM-Unterstützung für mehrere Plattformen und Leistungsverbesserungen für alle. SDKs Der CloudHSM CLI und der JCE-Anbieter wurden um neue Funktionen erweitert.

Plattformunterstützung

- Unterstützung für Amazon Linux 2023 auf ARM64 Architecture for all hinzugefügt SDKs.
- Unterstützung für Red Hat Enterprise Linux 9 (9.2+) für ARM64 Architektur für alle SDKs hinzugefügt.
- Unterstützung für Ubuntu 22.04 LTS auf ARM64 Architektur für alle hinzugefügt. SDKs

CloudHSM-CLI

- Der folgende Befehl wurde hinzugefügt:
 - [Einen Schlüssel mit CloudHSM CLI replizieren](#)
- Unterstützung für die Verbindung zu mehreren Clustern hinzugefügt. Weitere Informationen finden Sie unter [Verbindung zu mehreren Clustern mit CloudHSM CLI herstellen](#).

JCE-Anbieter

- Hinzugefügt KeyReferenceSpec für das Abrufen von Schlüsseln mit KeyStoreWithAttributes.
- Hinzugefügt getKeys zum gleichzeitigen Abrufen mehrerer Schlüssel mit. KeyStoreWithAttributes

Leistungsverbesserungen

- Leistungsverbesserungen für den NoPadding AES-CBC-Betrieb für alle. SDKs

Version 5.11.0

Das Client SDK 5.11.0 fügt neue Funktionen hinzu, verbessert die Stabilität und beinhaltet Bugfixes für alle. SDKs

Plattformunterstützung

- Unterstützung für Amazon Linux 2023 und RHEL 9 (9.2+) für alle hinzugefügt. SDKs
- Die Unterstützung für Ubuntu 18.04 LTS wurde aufgrund des kürzlichen Ablaufs entfernt.
- Die Unterstützung für Amazon Linux wurde aufgrund des kürzlichen Ablaufs entfernt.

CloudHSM-CLI

- Die folgenden Befehle wurden hinzugefügt:
 - [Die Kategorie Kryptozeichen in CloudHSM CLI](#)
 - [Die Kategorie Crypto Verify in CloudHSM CLI](#)
 - [Importieren Sie einen Schlüssel im PEM-Format mit der CloudHSM-CLI](#)
 - [Der Befehl zum Entpacken von Schlüsseln in der CloudHSM-CLI](#)
 - [Der Key Wrap-Befehl in der CloudHSM-CLI](#)
- [Exportieren Sie einen asymmetrischen Schlüssel mit CloudHSM CLI](#) unterstützt jetzt den Export von öffentlichen Schlüsseln.

OpenSSL Dynamic Engine

- Die AWS CloudHSM OpenSSL Dynamic Engine wird jetzt auf Plattformen unterstützt, auf denen eine OpenSSL-Bibliotheksversion 3.x installiert ist. Dazu gehören Amazon Linux 2023, RHEL 9 (9.2+) und Ubuntu 22.04.

JCE

- Unterstützung für JDK 17 und JDK 21 hinzugefügt.
- Unterstützung für AES-Schlüssel zur Verwendung für HMAC-Operationen hinzugefügt.
- Das neue Schlüsselattribut ID wurde hinzugefügt.
- Eine neue DataExceptionCause Variante für die Erschöpfung von Schlüsseln wurde eingeführt: `DataExceptionCause.KEY_EXHAUSTED`.

Fehlerbehebungen/Verbesserungen

- Die maximale Länge für das `label` Attribut wurde von 126 auf 127 Zeichen erhöht.

- Es wurde ein Fehler behoben, der das Auspacken von EC-Schlüsseln mit dem RsaOaep Mechanismus verhinderte.
- Ein bekanntes Problem für den GetKey-Vorgang im JCE-Anbieter wurde behoben. Weitere Einzelheiten finden Sie unter [Problem: Speicherverlust im Client-SDK 5 bei GetKey-Vorgängen](#).
- Die Protokollierung aller Triple-DES-Schlüssel, die ihr maximales Blocklimit SDKs für die Verschlüsselung erreicht haben, wurde gemäß FIPS 140-2 verbessert.
- Bekannte Probleme für die OpenSSL Dynamic Engine wurden hinzugefügt. Details dazu finden Sie unter [Bekannte Probleme mit der OpenSSL Dynamic Engine für AWS CloudHSM](#).

Version 5.10.0

Das Client SDK 5.10.0 verbessert die Stabilität und enthält Bugfixes für alle. SDKs

CloudHSM-CLI

- Es wurden neue Befehle hinzugefügt, mit denen Kunden Schlüssel mithilfe der CloudHSM-CLI verwalten können, darunter:
 - Symmetrische Schlüssel und asymmetrische Schlüsselpaare erstellen
 - Freigabe und Aufheben der Freigabe von Schlüsseln
 - Schlüssel mithilfe von Schlüsselattributen auflisten und filtern
 - Festlegen von Schlüsselattributen
 - Wichtige Referenzdateien generieren
 - Löschen von Schlüsseln
- Verbesserte Fehlerprotokollierung.
- Unterstützung für mehrzeilige Unicode-Befehle im interaktiven Modus hinzugefügt.

Fehlerbehebungen/Verbesserungen

- Verbesserte Leistung beim Importieren, Entpacken, Ableiten und Erstellen von Sitzungsschlüsseln für alle. SDKs
- Es wurde ein Fehler im JCE-Anbieter behoben, der verhinderte, dass temporäre Dateien beim Beenden entfernt wurden.
- Es wurde ein Fehler behoben, der unter bestimmten Bedingungen nach dem Austausch HSMs im Cluster zu einem Verbindungsfehler führte.

- Das JCEgetVersion-Ausgabeformat wurde geändert, um große kleinere Versionsnummern zu verarbeiten und die Patch-Nummer einzubeziehen.

Plattformunterstützung

- Unterstützung für Ubuntu 22.04 mit JCE, PKCS #11 und CloudHSM-CLI hinzugefügt (Unterstützung für OpenSSL Dynamic Engine ist noch nicht verfügbar).

Version 5.9.0

Das Client SDK 5.9.0 verbessert die Stabilität und beinhaltet Bugfixes für alle SDKs. Für alle SDKs wurde eine Optimierung vorgenommen, sodass Anwendungen sofort über einen Betriebsausfall informiert werden, wenn festgestellt wird, dass ein HSM nicht verfügbar ist. Diese Version enthält Leistungsverbesserungen für JCE.

JCE-Anbieter

- Verbesserte Leistung
- Ein [bekanntes Problem](#) mit der Erschöpfung des Sitzungspools wurde behoben

AWS CloudHSM veraltete Client-SDK-Versionen

Versionen 5.8.0 und früher sind veraltet. Wir raten davon ab, veraltete Versionen in Produktionsumgebungen zu verwenden. Wir stellen keine abwärtskompatiblen Updates für veraltete Versionen bereit und stellen auch keine veralteten Versionen zum Herunterladen bereit. Wenn Sie bei der Verwendung veralteter Versionen Auswirkungen auf die Produktion feststellen, müssen Sie ein Upgrade durchführen, um Softwarekorrekturen zu erhalten.

Veraltete Versionen des Client SDK 5

In diesem Abschnitt werden veraltete Versionen des Client SDK 5 aufgeführt.

Version 5.8.0

Version 5.8.0 bietet Quorum-Authentifizierung für CloudHSM CLI, SSL/TLS offload with JSSE, multi-slot support for PKCS #11, multi-cluster/multi Benutzerunterstützung für JCE, Schlüsselextraktion mit JCE, unterstütztes KeyFactory für JCE, neue Wiederholungskonfigurationen für Rückgabecodes ohne Terminal und enthält verbesserte Stabilität und Bugfixes für alle. SDKs

PKCS-#11-Bibliothek

- Unterstützung für die Konfiguration mit mehreren Steckplätzen wurde hinzugefügt.

JCE-Anbieter

- Konfigurationsbasierte Schlüsselextraktion hinzugefügt.
- Unterstützung für Konfigurationen mit mehreren Clustern und mehreren Benutzern hinzugefügt.
- Unterstützung für SSL- und TLS-Offload mit JSSE hinzugefügt.
- Unwrap-Unterstützung für hinzugefügt. AES/CBC/NoPadding
- Neue Typen von Schlüsselfabriken hinzugefügt: SecretKeyFactory und KeyFactory.

CloudHSM-CLI

- Unterstützung für Quorum-Authentifizierung hinzugefügt

Version 5.7.0

Version 5.7.0 führt die CloudHSM-CLI ein und enthält einen neuen verschlüsselten CMAC-Algorithmus (Message Authentication Code). Diese Version fügt die ARM-Architektur auf Amazon Linux 2 hinzu. Javadocs des JCE-Anbieters sind jetzt verfügbar für AWS CloudHSM.

PKCS-#11-Bibliothek

- Verbesserte Stabilität und Fehlerbehebungen.
- Wird jetzt auf der ARM-Architektur mit Amazon Linux 2 unterstützt.
- Algorithmen
 - CKM_AES_CMAC (signieren und verifizieren)

OpenSSL Dynamic Engine

- Verbesserte Stabilität und Fehlerbehebungen.
- Wird jetzt auf der ARM-Architektur mit Amazon Linux 2 unterstützt.

JCE-Anbieter

- Verbesserte Stabilität und Fehlerbehebungen.
- Algorithmen
 - AESCMAC

Version 5.6.0

Version 5.6.0 beinhaltet die Unterstützung neuer Mechanismen für die PKCS-#11-Bibliothek und den JCE-Anbieter. Darüber hinaus unterstützt Version 5.6 Ubuntu 20.04.

PKCS-#11-Bibliothek

- Verbesserte Stabilität und Fehlerbehebungen.
- Mechanismen
 - CKM_RSA_X_509, für die Modi Verschlüsseln, Entschlüsseln, Signieren und Verifizieren

OpenSSL Dynamic Engine

- Verbesserte Stabilität und Fehlerbehebungen.

JCE-Anbieter

- Verbesserte Stabilität und Fehlerbehebungen.
- Verschlüsselungen
 - RSA/ECB/NoPadding, für Verschlüsselungs- und Entschlüsselungsmodi

Unterstützte Schlüssel

- EC mit den Kurven secp224r1 und secp521r1

Plattformunterstützung

- Unterstützung für Ubuntu 20.04 hinzugefügt.

Version 5.5.0

Version 5.5.0 bietet Unterstützung für OpenJDK 11, Keytool- und Jarsigner-Integration sowie zusätzliche Mechanismen für den JCE-Anbieter. Behebt ein [bekanntes Problem](#), bei dem eine KeyGenerator Klasse den Schlüsselgrößenparameter fälschlicherweise als Anzahl von Bytes statt als Bits interpretiert.

PKCS-#11-Bibliothek

- Verbesserte Stabilität und Fehlerbehebungen.

OpenSSL Dynamic Engine

- Verbesserte Stabilität und Fehlerbehebungen.

JCE-Anbieter

- Unterstützung für die Dienstprogramme Keytool und Jarsigner
- Unterstützung für OpenJDK 11 auf allen Plattformen
- Verschlüsselungen
 - AES/CBC/NoPaddingModus „Verschlüsseln und Entschlüsseln“
 - AES/ECB/PKCS5Padding Modus „Verschlüsseln und Entschlüsseln“
 - AES/CTR/NoPaddingModus „Verschlüsseln und Entschlüsseln“
 - AES/GCM/NoPaddingModus „Wrap“ und „Unwrap“
 - DESede/ECB/PKCS5Padding Modus „Verschlüsseln und Entschlüsseln“
 - DESede/CBC/NoPaddingModus „Verschlüsseln und Entschlüsseln“
 - AESWrap/ECB/NoPaddingModus „Wrap“ und „Unwrap“
 - AESWrap/ECB/PKCS5Padding Wrap- und Unwrap-Modus
 - AESWrap/ECB/ZeroPaddingWrap- und Unwrap-Modus
 - RSA/ECB/PKCS1Padding Wrap- und Unwrap-Modus
 - RSA/ECB/OAEPWrap- und Unwrap-Modus
 - RSA/ECB/OAEPWithSHA-1 ANDMGF1 Padding Wrap- und Unwrap-Modus
 - RSA/ECB/OAEPWithSHA-224 ANDMGF1 Padding Wrap- und Unwrap-Modus
 - ~~RSA/ECB/OAEPWithSHA-256 ANDMGF1 Padding Wrap- und Unwrap-Modus~~

- RSA/ECB/OAEPWithSHA-384 ANDMGF1 Padding Wrap- und Unwrap-Modus
- RSA/ECB/OAEPWithSHA-512 ANDMGF1 Padding Wrap- und Unwrap-Modus
- RSAAESWrap/ECB/OAEPWithSHA-1 ANDMGF1 Padding Wrap- und Unwrap-Modus
- RSAAESWrap/ECB/OAEPWithSHA-224 ANDMGF1 Padding Wrap- und Unwrap-Modus
- RSAAESWrap/ECB/OAEPWithSHA-256 ANDMGF1 Padding Wrap- und Unwrap-Modus
- RSAAESWrap/ECB/OAEPWithSHA-384 ANDMGF1 Padding Wrap- und Unwrap-Modus
- RSAAESWrap/ECB/OAEPWithSHA-512 ANDMGF1 Padding Wrap- und Unwrap-Modus
- KeyFactory und SecretKeyFactory
 - RSA – 2048-Bit- bis 4096-Bit-RSA-Schlüssel, in Schritten von je 256 Bit.
 - AES – 128-, 192- und 256-Bit-AES-Schlüssel
 - EC-Schlüsselpaare für die NIST secp256r1- (P-256), secp384r1- (P-384) und secp256k1-Kurven.
 - DESede (3DES)
 - GenericSecret
 - HMAC — mit SHA1,, SHA224 SHA256 SHA384, SHA512 Hash-Unterstützung
- Signieren/Überprüfen
 - RSASSA-PSS
 - SHA1withRSA/PSS
 - SHA224withRSA/PSS
 - SHA256withRSA/PSS
 - SHA384withRSA/PSS
 - SHA512withRSA/PSS
 - SHA1withRSAandMGF1
 - SHA224withRSAandMGF1
 - SHA256withRSAandMGF1
 - SHA384withRSAandMGF1
 - SHA512withRSAandMGF1

Version 5.4.2

Version 5.4.2 beinhaltet verbesserte Stabilität und Bugfixes für alle SDKs. Dies ist auch die letzte Version für die CentOS-8-Plattform. Weitere Informationen finden Sie auf der [CentOS-Website](#).

PKCS-#11-Bibliothek

- Verbesserte Stabilität und Fehlerbehebungen.

OpenSSL Dynamic Engine

- Verbesserte Stabilität und Fehlerbehebungen.

JCE-Anbieter

- Verbesserte Stabilität und Fehlerbehebungen.

Version 5.4.1

Version 5.4.1 behebt ein [bekanntes Problem](#) mit der PKCS-#11-Bibliothek. Dies ist auch die letzte Version für die CentOS-8-Plattform. Weitere Informationen finden Sie auf der [CentOS-Website](#).

PKCS-#11-Bibliothek

- Verbesserte Stabilität und Fehlerbehebungen.

OpenSSL Dynamic Engine

- Verbesserte Stabilität und Fehlerbehebungen.

JCE-Anbieter

- Verbesserte Stabilität und Fehlerbehebungen.

Version 5.4.0

Version 5.4.0 bietet anfängliche Unterstützung für den JCE-Anbieter für alle Plattformen. Der JCE-Anbieter ist mit OpenJDK 8 kompatibel.

PKCS-#11-Bibliothek

- Verbesserte Stabilität und Fehlerbehebungen.

OpenSSL Dynamic Engine

- Verbesserte Stabilität und Fehlerbehebungen.

JCE-Anbieter

- Schlüsseltypen
 - RSA – 2048-Bit- bis 4096-Bit-RSA-Schlüssel, in Schritten von je 256 Bit.
 - AES – 128-, 192- und 256-Bit AES-Schlüssel.
 - ECC-Schlüsselpaare für die NIST secp256r1- (P-256), secp384r1- (P-384) und secp256k1-Kurven.
 - DESede (3DES)
 - HMAC — mit SHA1,, SHA224 SHA256 SHA384, SHA512 Hash-Unterstützung.
- Chiffren (nur verschlüsseln und entschlüsseln)
 - AES/GCM/NoPadding
 - AES/ECB/NoPadding
 - AES/CBC/PKCS5. Polsterung
 - DESede/ECB/NoPadding
 - DESede/CBC/PKCS5 Polsterung
 - AES/CTR/NoPadding
 - RSA/ECB/PKCS1 Polsterung
 - RSA/ECB/OAEPPadding
 - RSA/ECB/OAEPWithSHA-1 Polsterung ANDMGF1
 - RSA/ECB/OAEPWithSHA-224 Polsterung ANDMGF1
 - RSA/ECB/OAEPWithSHA-256 Polsterung ANDMGF1
 - RSA/ECB/OAEPWithSHA-384 Polsterung ANDMGF1
 - RSA/ECB/OAEPWithSHA-512 Polsterung ANDMGF1

- Digests
 - SHA-1
 - SHA-224
 - SHA-1
 - SHA-384
 - SHA-512
- Signieren/Überprüfen
 - NONEwithRSA
 - SHA1withRSA
 - SHA224withRSA
 - SHA256withRSA
 - SHA384withRSA
 - SHA512withRSA
 - NONEwithECDSA
 - SHA1withECDSA
 - SHA224withECDSA
 - SHA256withECDSA
 - SHA384withECDSA
 - SHA512withECDSA
- Integration mit Java KeyStore

Version 5.3.0

PKCS-#11-Bibliothek

- Verbesserte Stabilität und Fehlerbehebungen.

OpenSSL Dynamic Engine

- Unterstützung für ECDSA sign/verify mit den Kurven P-256, P-384 und secp256k1 hinzugefügt.
- Unterstützung für folgende Plattformen hinzufügen: Amazon Linux, Amazon Linux 2, CentOS 7.8+,

- Unterstützung für OpenSSL Version 1.0.2 hinzugefügt.
- Verbesserte Stabilität und Fehlerbehebungen.

JCE-Anbieter

- Schlüsseltypen
 - RSA – 2048-Bit- bis 4096-Bit-RSA-Schlüssel, in Schritten von je 256 Bit.
 - AES – 128-, 192- und 256-Bit AES-Schlüssel.
 - EC-Schlüsselpaare für die NIST secp256r1- (P-256), secp384r1- (P-384) und secp256k1-Kurven.
 - DESede (3DES)
 - HMAC — mit SHA1,, SHA224 SHA256 SHA384, SHA512 Hash-Unterstützung.
- Chiffren (nur verschlüsseln und entschlüsseln)
 - AES/GCM/NoPadding
 - AES/ECB/NoPadding
 - AES/CBC/PKCS5. Polsterung
 - DESede/ECB/NoPadding
 - DESede/CBC/PKCS5 Polsterung
 - AES/CTR/NoPadding
 - RSA/ECB/PKCS1 Polsterung
 - RSA/ECB/OAEPPadding
 - RSA/ECB/OAEPWithSHA-1 Polsterung ANDMGF1
 - RSA/ECB/OAEPWithSHA-224 Polsterung ANDMGF1
 - RSA/ECB/OAEPWithSHA-256 Polsterung ANDMGF1
 - RSA/ECB/OAEPWithSHA-384 Polsterung ANDMGF1
 - RSA/ECB/OAEPWithSHA-512 Polsterung ANDMGF1
- Digests
 - SHA-1
 - SHA-224
 - **SHA-1**
 - SHA-384

- SHA-512
- Signieren/Überprüfen
 - NONEwithRSA
 - SHA1withRSA
 - SHA224withRSA
 - SHA256withRSA
 - SHA384withRSA
 - SHA512withRSA
 - NONEwithECDSA
 - SHA1withECDSA
 - SHA224withECDSA
 - SHA256withECDSA
 - SHA384withECDSA
 - SHA512withECDSA
- Integration mit Java KeyStore

Version 5.2.1

PKCS-#11-Bibliothek

- Verbesserte Stabilität und Fehlerbehebungen.

OpenSSL Dynamic Engine

- Verbesserte Stabilität und Fehlerbehebungen.

Version 5.2.0

Version 5.2.0 erweitert die PKCS-#11-Bibliothek um Unterstützung für zusätzliche Schlüsseltypen und Mechanismen.

PKCS-#11-Bibliothek

Schlüsseltypen

- ECDSA – Kurven P-224, P-256, P-384, P-521 und secp256k1
- Triple DES (3DES)

Mechanismen

- CKM_EC_KEY_PAIR_GEN
- CKM__KEY_GEN DES3
- DES3CKM_CBC
- DES3CKM__CBC_PAD
- CKM_DES3_EZB
- CKM_ECDSA
- CKM_ECDSA_SHA1
- CKM_ECDSA_SHA224
- CKM_ECDSA_SHA256
- CKM_ECDSA_SHA384
- CKM_ECDSA_SHA512
- CKM_RSA_PKCS für Verschlüsseln/Entschlüsseln

OpenSSL Dynamic Engine

- Verbesserte Stabilität und Fehlerbehebungen.

Version 5.1.0

Mit Version 5.1.0 wird die PKCS-#11-Bibliothek um zusätzliche Mechanismen erweitert.

PKCS-#11-Bibliothek

Mechanismen

- CKM_RSA_PKCS für einpacken/entpacken
- CKM_RSA_PKCS_PSS
- SHA1CKM_RSA_PKCS_PSS
- SHA224CKM_RSA_PKCS_PSS
- SHA256CKM_RSA_PKCS_PSS

- SHA384CKM_RSA_PKCS_PSS
- SHA512CKM_RSA_PKCS_PSS
- CKM_AES_ECB
- CKM_AES_CTR
- CKM_AES_CBC
- CKM_AES_CBC_PAD
- SP800CKM__108_COUNTER_KDF
- CKM_GENERIC_SECRET_KEY_GEN
- CKM_SHA_1_HMAC
- SHA224CKM_HMAC
- SHA256CKM_HMAC
- SHA384CKM_HMAC
- SHA512CKM_HMAC
- Nur Wrap/Unwrap CKM_RSA_PKCS_OAEP
- CKM_RSA_AES_KEY_WRAP
- CKM_CLOUDHSM_AES_KEY_WRAP_NO_PAD
- CKM_CLOUDHSM_AES_KEY_WRAP__PAD PKCS5
- CKM_CLOUDHSM_AES_KEY_WRAP_ZERO_PAD

API-Operationen

- C_ CreateObject
- C_ DeriveKey
- C_ WrapKey
- C_ UnWrapKey

OpenSSL Dynamic Engine

- Verbesserte Stabilität und Fehlerbehebungen.

Version 5.0.1

Version 5.0.1 fügt anfängliche Unterstützung für OpenSSL Dynamic Engine hinzu.

PKCS-#11-Bibliothek

- Verbesserte Stabilität und Fehlerbehebungen.

OpenSSL Dynamic Engine

- Erste Version von OpenSSL Dynamic Engine.
- Diese Version bietet einführende Unterstützung für Schlüsseltypen und OpenSSL APIs:
 - RSA-Schlüsselgenerierung für 2048-, 3072- und 4096-Bit-Schlüssel
 - OpenSSL APIs:
 - [RSA Sign mit RSA](#) PKCS mit SHA1 /224/256/384/512 und RSA PSS
 - [Generierung von RSA-Schlüsseln](#)

Weitere Informationen finden Sie unter [OpenSSL Dynamic Engine](#).

- Unterstützte Plattformen: CentOS 8.3+, Red Hat Enterprise Linux (RHEL) 8.3+ und Ubuntu 18.04 LTS
 - Benötigt: OpenSSL 1.1.1

Weitere Informationen finden Sie unter [Unterstützte Plattformen](#).

- Support für SSL/TLS Offload auf CentOS 8.3+, Red Hat Enterprise Linux (RHEL) 8.3 und Ubuntu 18.04 LTS, einschließlich NGINX 1.19 (für ausgewählte Cipher Suites).

[Weitere Informationen finden Sie unter SSL/TLS-Offload unter Linux mit Tomcat oder SSL/TLS-Offload unter Linux mit NGINX oder Apache.](#)

Version 5.0.0

Version 5.0.0 ist die erste Version.

PKCS-#11-Bibliothek

- Dies ist die Erstversion.

Unterstützung für Einsteiger der PKCS-#11-Bibliothek in Client SDK Version 5.0.0

Dieser Abschnitt beschreibt die Unterstützung für Schlüsseltypen, Mechanismen, API-Operationen und Attribute Client-SDK Version 5.0.0.

Schlüsseltypen:

- AES – 128-, 192- und 256-Bit-AES-Schlüssel
- RSA – 2048-Bit- bis 4096-Bit-RSA-Schlüssel, in Schritten von je 256 Bit.

Mechanismen:

- CKM_AES_GCM
- CKM_AES_KEY_GEN
- CKM_CLOUDHSM_AES_GCM
- CKM_RSA_PKCS
- CKM_RSA_X9_31_KEY_PAIR_GEN
- CKM_SHA1
- CKM_RSA_PKCS_SHA1
- CKM_SHA224
- CKM_RSA_PKCS_SHA224
- CKM_SHA256
- CKM_RSA_PKCS_SHA256
- CKM_SHA384
- CKM_RSA_PKCS_SHA384
- CKM_SHA512
- CKM_RSA_PKCS_SHA512

API-Operationen:

- 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

Attribute:

- GenerateKeyPair
 - Alle RSA-Schlüsselattribute
- GenerateKey
 - Alle AES-Schlüsselattribute
- GetAttributeValue
 - Alle RSA-Schlüsselattribute
 - Alle AES-Schlüsselattribute

Beispiele:

- [Schlüssel generieren \(AES, RSA, EC\)](#)
- [Schlüsselattribute auflisten](#)
- [Verschlüsseln und Entschlüsseln von Daten mit AES GCM](#)
- [Signieren und Verifizieren von Daten mit RSA](#)

Veraltete Versionen des Client SDK 3

In diesem Abschnitt werden veraltete Client SDK 3-Versionen aufgeführt.

Version 3.4.4

Version 3.4.4 enthält Aktualisierungen für den JCE-Anbieter.

AWS CloudHSM Kundensoftware

- Versionsnummer wurde aus Konsistenzgründen aktualisiert.

PKCS-#11-Bibliothek

- Versionsnummer wurde aus Konsistenzgründen aktualisiert.

OpenSSL Dynamic Engine

- Versionsnummer wurde aus Konsistenzgründen aktualisiert.

JCE-Anbieter

- Aktualisieren Sie log4j auf Version 2.17.1.

Windows (CNG- und KSP-Anbieter)

- Versionsnummer wurde aus Konsistenzgründen aktualisiert.

Version 3.4.3

Version 3.4.3 enthält Aktualisierungen für den JCE-Anbieter.

AWS CloudHSM Software für Kunden

- Versionsnummer wurde aus Konsistenzgründen aktualisiert.

PKCS-#11-Bibliothek

- Versionsnummer wurde aus Konsistenzgründen aktualisiert.

OpenSSL Dynamic Engine

- Versionsnummer wurde aus Konsistenzgründen aktualisiert.

JCE-Anbieter

- Aktualisieren Sie log4j auf Version 2.17.0.

Windows (CNG- und KSP-Anbieter)

- Versionsnummer wurde aus Konsistenzgründen aktualisiert.

Version 3.4.2

Version 3.4.2 enthält Aktualisierungen für den JCE-Anbieter.

AWS CloudHSM Software für Kunden

- Versionsnummer wurde aus Konsistenzgründen aktualisiert.

PKCS-#11-Bibliothek

- Versionsnummer wurde aus Konsistenzgründen aktualisiert.

OpenSSL Dynamic Engine

- Versionsnummer wurde aus Konsistenzgründen aktualisiert.

JCE-Anbieter

- Aktualisieren Sie log4j auf Version 2.16.0.

Windows (CNG- und KSP-Anbieter)

- Versionsnummer wurde aus Konsistenzgründen aktualisiert.

Version 3.4.1

Version 3.4.1 enthält Aktualisierungen für den JCE-Anbieter.

AWS CloudHSM Software für Kunden

- Versionsnummer wurde aus Konsistenzgründen aktualisiert.

PKCS-#11-Bibliothek

- Versionsnummer wurde aus Konsistenzgründen aktualisiert.

OpenSSL Dynamic Engine

- Versionsnummer wurde aus Konsistenzgründen aktualisiert.

JCE-Anbieter

- Aktualisieren Sie log4j auf Version 2.15.0.

Windows (CNG- und KSP-Anbieter)

- Versionsnummer wurde aus Konsistenzgründen aktualisiert.

Version 3.4.0

Version 3.4.0 enthält Aktualisierungen für alle Komponenten.

AWS CloudHSM Software für Kunden

- Verbesserte Stabilität und Fehlerbehebungen.

PKCS-#11-Bibliothek

- Verbesserte Stabilität und Fehlerbehebungen.

OpenSSL Dynamic Engine

- Verbesserte Stabilität und Fehlerbehebungen.

JCE-Anbieter

- Verbesserte Stabilität und Fehlerbehebungen.

Windows (CNG- und KSP-Anbieter)

- Verbesserte Stabilität und Fehlerbehebungen.

Version 3.3.2

Version 3.3.2 behebt ein [Problem](#) mit dem client_info-Skript.

AWS CloudHSM Client-Software

- Versionsnummer wurde aus Konsistenzgründen aktualisiert.

PKCS-#11-Bibliothek

- Versionsnummer wurde aus Konsistenzgründen aktualisiert.

OpenSSL Dynamic Engine

- Versionsnummer wurde aus Konsistenzgründen aktualisiert.

JCE-Anbieter

- Versionsnummer wurde aus Konsistenzgründen aktualisiert.

Windows (CNG- und KSP-Anbieter)

- Versionsnummer wurde aus Konsistenzgründen aktualisiert.

Version 3.3.1

Version 3.3.1 enthält Aktualisierungen für alle Komponenten.

AWS CloudHSM Software für Kunden

- Verbesserte Stabilität und Fehlerbehebungen.

PKCS-#11-Bibliothek

- Verbesserte Stabilität und Fehlerbehebungen.

OpenSSL Dynamic Engine

- Verbesserte Stabilität und Fehlerbehebungen.

JCE-Anbieter

- Verbesserte Stabilität und Fehlerbehebungen.

Windows (CNG- und KSP-Anbieter)

- Verbesserte Stabilität und Fehlerbehebungen.

Version 3.3.0

Version 3.3.0 fügt die Zwei-Faktor-Authentifizierung (2FA) und weitere Verbesserungen hinzu.

AWS CloudHSM Software für Kunden

- 2FA-Authentifizierung für Crypto Officer (CO) hinzugefügt. Weitere Informationen finden Sie unter [Verwalten der Zwei-Faktor-Authentifizierung für Crypto Officer](#).
- Die Plattformunterstützung für RedHat Enterprise Linux 6 und CentOS 6 wurde entfernt. Weitere Informationen finden Sie unter [Linux-Support](#).
- Es wurde eine eigenständige Version von CMU zur Verwendung mit Client-SDK 5 oder Client-SDK 3 hinzugefügt. Dies ist dieselbe CMU-Version, die im Client-Daemon der Version 3.3.0 enthalten ist. Jetzt können Sie CMU herunterladen, ohne den Client-Daemon herunterzuladen.

PKCS-#11-Bibliothek

- Verbesserte Stabilität und Fehlerbehebungen.
- Die Plattformunterstützung für RedHat Enterprise Linux 6 und CentOS 6 wurde entfernt. Weitere Informationen finden Sie unter [Linux-Support](#).

OpenSSL Dynamic Engine

- Versionsnummer wurde aus Konsistenzgründen aktualisiert.
- Die Plattformunterstützung für RedHat Enterprise Linux 6 und CentOS 6 wurde entfernt. Weitere Informationen finden Sie unter [Linux-Support](#).

JCE-Anbieter

- Verbesserte Stabilität und Fehlerbehebungen.

- Die Plattformunterstützung für RedHat Enterprise Linux 6 und CentOS 6 wurde entfernt. Weitere Informationen finden Sie unter [Linux-Support](#).

Windows (CNG- und KSP-Anbieter)

- Versionsnummer wurde aus Konsistenzgründen aktualisiert.

Version 3.2.1

Version 3.2.1 fügt eine Konformitätsanalyse zwischen der AWS CloudHSM Implementierung der PKCS #11 -Bibliothek und des PKCS #11 -Standards, neuen Plattformen und anderen Verbesserungen hinzu.

AWS CloudHSM Software für den Kunden

- Plattformunterstützung für CentOS 8, RHEL 8 und Ubuntu 18.04 LTS hinzugefügt. Weitere Informationen finden Sie unter [???](#).

PKCS-#11-Bibliothek

- [Bericht zur Einhaltung der PKCS-#11-Bibliothek für das Client-SDK 3.2.1](#)
- Plattformunterstützung für CentOS 8, RHEL 8 und Ubuntu 18.04 LTS hinzugefügt. Weitere Informationen finden Sie unter [???](#).

OpenSSL Dynamic Engine

- Keine Unterstützung für CentOS 8, RHEL 8 und Ubuntu 18.04 LTS. Weitere Informationen finden Sie unter [???](#).

JCE-Anbieter

- Plattformunterstützung für CentOS 8, RHEL 8 und Ubuntu 18.04 LTS hinzugefügt. Weitere Informationen finden Sie unter [???](#).

Windows (CNG- und KSP-Anbieter)

- Verbesserte Stabilität und Fehlerbehebungen.

Version 3.2.0

Version 3.2.0 bietet Unterstützung für das Maskieren von Passwörtern und andere Verbesserungen.

AWS CloudHSM Software für Kunden

- Fügt Unterstützung für das Verbergen Ihres Passworts hinzu, wenn Sie Befehlszeilentools verwenden. Weitere Informationen finden Sie unter [loginHSM und logoutHSM](#) (cloudhsm_mgmt_util) und [loginHSM und logoutHSM](#) (key_mgmt_util).

PKCS-#11-Bibliothek

- Integriert die Unterstützung für das Hashing großer Datenmengen in Software für einige PKCS-#11-Mechanismen, die zuvor nicht unterstützt wurden. Weitere Informationen finden Sie unter [unterstützte Mechanismen](#).

OpenSSL Dynamic Engine

- Verbesserte Stabilität und Fehlerbehebungen.

JCE-Anbieter

- Versionsnummer wurde aus Konsistenzgründen aktualisiert.

Windows (CNG- und KSP-Anbieter)

- Verbesserte Stabilität und Fehlerbehebungen.

Version 3.1.2

Version 3.1.2 enthält Aktualisierungen für den JCE-Anbieter.

AWS CloudHSM Software für Kunden

- Versionsnummer wurde aus Konsistenzgründen aktualisiert.

PKCS-#11-Bibliothek

- Versionsnummer wurde aus Konsistenzgründen aktualisiert.

OpenSSL Dynamic Engine

- Versionsnummer wurde aus Konsistenzgründen aktualisiert.

JCE-Anbieter

- Aktualisieren Sie log4j auf Version 2.13.3.

Windows (CNG- und KSP-Anbieter)

- Versionsnummer wurde aus Konsistenzgründen aktualisiert.

Version 3.1.1

AWS CloudHSM Software für Kunden

- Versionsnummer wurde aus Konsistenzgründen aktualisiert.

PKCS-#11-Bibliothek

- Versionsnummer wurde aus Konsistenzgründen aktualisiert.

OpenSSL Dynamic Engine

- Versionsnummer wurde aus Konsistenzgründen aktualisiert.

JCE-Anbieter

- Fehlerbehebungen und Leistungsverbesserungen.

Windows (CNG, KSP)

- Versionsnummer wurde aus Konsistenzgründen aktualisiert.

Version 3.1.0

Version 3.1.0 fügt das [standardkonforme AES Key Wrapping](#) hinzu.

AWS CloudHSM Software für Kunden

- Eine neue Upgrade-Anforderung: Die Version Ihres Clients muss mit der Version aller von Ihnen verwendeten Softwarebibliotheken übereinstimmen. Für ein Upgrade müssen Sie einen Stapelbefehl verwenden, der gleichzeitig den Client und alle Bibliotheken aktualisiert. Weitere Informationen finden Sie unter [Client-SDK-3-Upgrade](#).
- Key_mgmt_util (KMU) enthält folgende Aktualisierungen:
 - Es wurden zwei neue AES-Verschlüsselungsmethoden hinzugefügt: standardkonforme AES-Verschlüsselung mit Zero Padding und AES-Verschlüsselung ohne Padding. Weitere Informationen finden Sie unter [wrapKey](#) oder [unwrapKey](#).
 - Die Möglichkeit, beim Umschließen eines Schlüssels mit PKCS5 AES_KEY_WRAP_PAD_ eine benutzerdefinierte IV anzugeben, wurde deaktiviert. Weitere Informationen finden Sie unter [AES Key Wrapping](#).

PKCS-#11-Bibliothek

- Es wurden zwei neue AES-Verschlüsselungsmethoden hinzugefügt: standardkonforme AES-Verschlüsselung mit Zero Padding und AES-Verschlüsselung ohne Padding. Weitere Informationen finden Sie unter [AES Key Wrapping](#).
- Sie können die Salt-Länge für RSA-PSS-Signaturen konfigurieren. [Informationen zur Verwendung dieser Funktion finden Sie unter Konfigurierbare Salzlänge für RSA-PSS-Signaturen](#). GitHub

OpenSSL Dynamic Engine

- **BAHNBRECHENDE ÄNDERUNG:** Die Verschlüsselungssammlungen TLS 1.0 und 1.2 mit SHA1 sind in OpenSSL Engine 3.1.0 nicht verfügbar. Dieses Problem wird in Kürze behoben.
- Wenn Sie beabsichtigen, die OpenSSL Dynamic Engine-Bibliothek auf RHEL 6 oder CentOS 6 zu installieren, informieren Sie sich über ein [bekanntes Problem](#) bezüglich der OpenSSL-Standardversion, die auf diesen Betriebssystemen installiert ist.
- Stabilitätsverbesserungen und Fehlerbehebungen

JCE-Anbieter

- **BAHNBRECHENDE ÄNDERUNG:** Um ein Problem mit der Kompatibilität mit der Java Cryptography Extension (JCE) zu beheben, verwendet AES Wrap and Unwrap jetzt korrekt den

Algorithmus anstelle des AESWrap AES-Algorithmus. Das bedeutet, dass `Cipher.WRAP_MODE` `Cipher.UNWRAP_MODE` AES/CBC-Mechanismen nicht mehr erfolgreich sind. AES/ECB

Um ein Upgrade auf die Client-Version 3.1.0 durchzuführen, müssen Sie Ihren Code aktualisieren. Wenn Sie bereits verpackte Schlüssel haben, müssen Sie besonders auf den Mechanismus zum Entpacken achten und darauf, wie sich die IV-Standardwerte geändert haben. Wenn Sie Schlüssel mit der Client-Version 3.0.0 oder früher verpackt haben, müssen Sie in 3.1.1 Ihre vorhandenen Schlüssel AESWrap/ECB/PKCS5Padding zum Entpacken verwenden. Weitere Informationen finden Sie unter [AES Key Wrapping](#).

- Sie können mehrere Schlüssel mit demselben Label aus der JCE-Anbieter auflisten. Informationen dazu, wie Sie durch alle verfügbaren Schlüssel iterieren können, [finden Sie](#) unter Alle Schlüssel suchen auf. GitHub
- Sie können bei der Schlüsselerstellung restriktivere Werte für Attribute festlegen, einschließlich der Angabe unterschiedlicher Labels für öffentliche und private Schlüssel. Weitere Informationen finden Sie unter [Unterstützte Java-Attribute](#).

Windows (CNG, KSP)

- Verbesserte Stabilität und Fehlerbehebungen.

AWS CloudHSM end-of-life SDK-Versionen für Kunden

Die folgenden AWS CloudHSM Client-Versionen haben das Ende des Supports erreicht. Diese AWS CloudHSM Client-Versionen sind nicht mehr mit dem Service kompatibel und werden nicht aktualisiert. Um die Sicherheit Ihrer Anwendung zu gewährleisten, werden AWS CloudHSM möglicherweise Verbindungen von Versionen abgelehnt, deren Support abgelaufen ist.

- Die SDK-Versionen 3.4.4 und früher haben das Ende des Supports erreicht.
- Die SDK-Versionen 5.8.0 und früher haben das Ende der Unterstützung erreicht.

Dokumentverlauf

In diesem Thema werden wichtige Aktualisierungen im AWS CloudHSM -Benutzerhandbuch beschrieben.

Themen

- [Neueste Aktualisierungen](#)
- [Frühere Aktualisierungen](#)

Neueste Aktualisierungen

Die folgende Tabelle enthält signifikante Änderungen an dieser Dokumentation seit April 2018. Neben den hier aufgelisteten größeren Änderungen aktualisieren wir die Dokumentation regelmäßig überarbeitet, um Beschreibungen und Beispiele zu verbessern und Ihre Rückmeldungen zu berücksichtigen. Wenn Sie über signifikante Änderungen benachrichtigt werden möchten, verwenden Sie den Link oben rechts, um den RSS-Feed zu abonnieren.

Einzelheiten zu neuen Versionen finden Sie unter [Downloads für das AWS CloudHSM Client-SDK](#).

Änderung	Beschreibung	Datum
Neue Version hinzugefügt	AWS CloudHSM Client-Version 5.17.1 veröffentlicht.	19. März 2026
Neue Version hinzugefügt	AWS CloudHSM Client-Version 5.17.0 veröffentlicht.	6. Januar 2026
Download-Links für OpenSSL Provider hinzugefügt	OpenSSL Provider-Download-Links zur AWS CloudHSM Client-Version 5.17 hinzugefügt.	6. Januar 2026
Neue Version hinzugefügt	AWS CloudHSM Client-Version 5.16.2 veröffentlicht.	16. September 2025
Neue Version hinzugefügt	AWS CloudHSM Client-Version 5.16.1 veröffentlicht.	25. Juni 2025

Neue Version hinzugefügt	AWS CloudHSM Client-Version 5.16.0 veröffentlicht.	01. Mai 2025
Neue Version hinzugefügt	AWS CloudHSM Client-Version 5.15.0 veröffentlicht.	3. Februar 2025
Neue Version hinzugefügt	AWS CloudHSM Client-Version 5.14.0 veröffentlicht.	26. November 2024
Neuer HSM-Typ und Clustermodus	Unterstützung für die Erstellung von Clustern (hsm2m.medium) im FIPS-Modus hinzugefügt.	20. August 2024
Neue Version hinzugefügt	Client-Version 5.13.0 veröffentlicht. AWS CloudHSM	13. August 2024
Neuer HSM-Typ und Clustermodus	Ein neuer HSM-Typ (hsm2m.medium) und ein neuer Clustermodus (Non-FIPS) wurden eingeführt.	10. Juni 2024
Neue Version hinzugefügt	Client-Version 5.12.0 veröffentlicht AWS CloudHSM .	20. März 2024
Neue Version hinzugefügt	AWS CloudHSM Client-Version 5.11.0 veröffentlicht.	17. Januar 2024
Neue Version hinzugefügt	AWS CloudHSM Client-Version 5.10.0 veröffentlicht.	28. Juli 2023
Neue Version hinzugefügt	AWS CloudHSM Client-Version 5.9.0 veröffentlicht.	23. Mai 2023
Neue Version hinzugefügt	AWS CloudHSM Client-Version 5.8.0 veröffentlicht.	16. März 2023
Neue Version hinzugefügt	AWS CloudHSM Client-Version 5.7.0 veröffentlicht.	16. November 2022

Neue Version hinzugefügt	AWS CloudHSM Client-Version 5.6.0 veröffentlicht.	01. September 2022
Neue Version hinzugefügt	AWS CloudHSM Client-Version 5.5.0 veröffentlicht.	13. Mai 2022
Neue Version hinzugefügt	AWS CloudHSM Client-Version 5.4.2 veröffentlicht.	18. März 2022
Neue Version hinzugefügt	AWS CloudHSM Client-Version 5.4.1 veröffentlicht.	10. Februar 2022
Neue Version hinzugefügt	Die AWS CloudHSM JCE-Provider-Version 5.4.0 für Windows-Plattformen wurde veröffentlicht.	1. Februar 2022
Neue Version hinzugefügt	Die AWS CloudHSM Client-Version 5.4.0 wurde veröffentlicht, die anfängliche Unterstützung für den JCE-Anbieter für alle Linux-Plattformen bietet.	28. Januar 2022
Neue Version hinzugefügt	AWS CloudHSM Client-Version 5.3.0 veröffentlicht.	3. Januar 2022
Neue Version hinzugefügt	AWS CloudHSM Client-Version 3.4.4 veröffentlicht.	3. Januar 2022
Neue Version hinzugefügt	AWS CloudHSM Client-Version 3.4.3 veröffentlicht.	20. Dezember 2021
Neue Version hinzugefügt	AWS CloudHSM Client-Version 3.4.2 veröffentlicht.	15. Dezember 2021
Neue Version hinzugefügt	AWS CloudHSM Client-Version 3.4.1 veröffentlicht.	10. Dezember 2021

Neue Version hinzugefügt	AWS CloudHSM Client-Version 5.2.1 veröffentlicht.	4. Oktober 2021
Neue Version hinzugefügt	AWS CloudHSM Client-Version 3.4.0 veröffentlicht.	25. August 2021
Neue Version hinzugefügt	AWS CloudHSM Client-Version 5.2.0 veröffentlicht.	3. August 2021
Neue Version hinzugefügt	AWS CloudHSM Client-Version 3.3.2 veröffentlicht.	2. Juli 2021
Neue Version hinzugefügt	AWS CloudHSM Client-Version 5.1.0 veröffentlicht.	1. Juni 2021
Neue Version hinzugefügt	AWS CloudHSM Client-Version 3.3.1 veröffentlicht.	26. April 2021
Neue Version hinzugefügt	AWS CloudHSM Client-Version 5.0.1 veröffentlicht.	8. April 2021
Neue Version hinzugefügt	AWS CloudHSM Client-Version 5.0.0 veröffentlicht.	12. März 2021
Zusätzliche neue Inhalte	Schnittstellen-VPC-Endpunkt hinzugefügt, eine AWS-Funktion, mit der Sie eine private Verbindung zwischen Ihrer VPC herstellen können, AWS CloudHSM ohne dass ein Zugriff über das Internet oder über ein NAT-Gerät, eine VPN-Verbindung oder eine Direct Connect Verbindung erforderlich ist.	10. Februar 2021
Neue Version hinzugefügt	AWS CloudHSM Client-Version 3.3.0 veröffentlicht.	3. Februar 2021

Zusätzliche neue Inhalte	Es wurde die verwaltete Aufbewahrung von Sicherungen hinzugefügt, eine Funktion, die alte Sicherungen automatisch löscht.	18. November 2020
Zusätzliche neue Inhalte	Es wurde ein Konformitätsbericht hinzugefügt, der die Implementierung der PKCS #11 -Bibliothek im AWS CloudHSM Client-SDK 3.2.1 mit dem PKCS #11 -Standard analysiert.	29. Oktober 2020
Neue Version hinzugefügt	AWS CloudHSM Client-Version 3.2.1 veröffentlicht.	8. Oktober 2020
Zusätzliche neue Inhalte	Es wurde eine Dokumentation hinzugefügt, die die wichtigsten Synchronisationseinstellungen in AWS CloudHSM beschreibt.	1. September 2020
Neue Version hinzugefügt	AWS CloudHSM Client-Version 3.2.0 veröffentlicht.	31. August 2020
Neue Version hinzugefügt	AWS CloudHSM Client-Version 3.1.2 veröffentlicht.	30. Juli 2020
Neue Version hinzugefügt	AWS CloudHSM Client-Version 3.1.1 veröffentlicht.	3. Juni 2020
Neue Version hinzugefügt	AWS CloudHSM Client-Version 3.1.0 veröffentlicht.	21. Mai 2020
Neue Version hinzugefügt	AWS CloudHSM Client-Version 3.0.1 veröffentlicht.	20. April 2020

Neue Version hinzugefügt	AWS CloudHSM Client-Version 3.0.0 für die Windows Server-Plattform veröffentlicht.	30. Oktober 2019
Neue Version hinzugefügt	Veröffentlichte AWS CloudHSM Client-Version 3.0.0 für alle Plattformen außer Windows.	22. Oktober 2019
Neue Version hinzugefügt	AWS CloudHSM Client-Version 2.0.4 veröffentlicht.	26. August 2019
Neue Version hinzugefügt	AWS CloudHSM Client-Version 2.0.3 veröffentlicht.	13. Mai 2019
Neue Version hinzugefügt	AWS CloudHSM Client-Version 2.0.1 veröffentlicht.	21. März 2019
Neue Version hinzugefügt	AWS CloudHSM Client-Version 2.0.0 veröffentlicht.	6. Februar 2019
Zusätzlicher Support für Regionen	AWS CloudHSM Unterstützung für die Regionen EU (Stockholm) und AWS GovCloud (USA-Ost) hinzugefügt.	19. Dezember 2018
Neue Version hinzugefügt	AWS CloudHSM Client-Version 1.1.2 für Windows veröffentlicht.	20. November 2018
Bekannte Probleme aktualisiert	Neue Inhalte wurden dem Handbuch zur Fehlerbehebung hinzugefügt.	8. November 2018
Neue Version hinzugefügt	AWS CloudHSM Client-Version 1.1.2 für Linux-Plattformen veröffentlicht.	8. November 2018

Zusätzlicher Support für Regionen	AWS CloudHSM Unterstützung für die Regionen EU (Paris) und Asien-Pazifik (Seoul) hinzugefügt.	24. Oktober 2018
Zusätzliche neue Inhalte	Es wurde die Möglichkeit hinzugefügt, AWS CloudHSM Backups zu löschen und wiederherzustellen.	10. September 2018
Zusätzliche neue Inhalte	Automatische Übermittlung von Auditprotokollen zu Amazon CloudWatch Logs hinzugefügt.	13. August 2018
Zusätzliche neue Inhalte	Es wurde die Möglichkeit hinzugefügt, ein AWS CloudHSM Cluster-Backup regionsübergreifend zu kopieren.	30. Juli 2018
Zusätzlicher Support für Regionen	AWS CloudHSM Unterstützung für die Region EU (London) hinzugefügt.	13. Juni 2018
Zusätzliche neue Inhalte	AWS CloudHSM Client- und Bibliotheksunterstützung für Amazon Linux 2, Red Hat Enterprise Linux (RHEL) 6, Red Hat Enterprise Linux (RHEL) 7, CentOS 6, CentOS 7 und Ubuntu 16.04 LTS hinzugefügt.	10. Mai 2018
Neue Version hinzugefügt	AWS CloudHSM Ein Windows-Client wurde hinzugefügt.	30. April 2018

Frühere Aktualisierungen

In der folgenden Tabelle werden die wichtigen Änderungen gegenüber der AWS CloudHSM Version vor 2018 beschrieben.

Änderungen	Beschreibung	Date
Neuer Inhalt	Quorum-Authentifizierung (M-of-N-Zugriffskontrolle) für Kryptobeamte hinzugefügt (COs). Weitere Informationen finden Sie unter Verwendung von CloudHSM Management Utility (CMU) zur Verwaltung der Quorum-Authentifizierung (M-von-N-Zugriffskontrolle) .	9. November 2017
Aktualisierung	Dokumentation zur Verwendung des Befehlszeilen-Tools <code>key_mgmt_util</code> hinzugefügt. Weitere Informationen finden Sie unter Referenz für Befehle AWS CloudHSM der Key Management Utility .	9. November 2017
Neuer Inhalt	Oracle Transparent Data Encryption hinzugefügt. Weitere Informationen finden Sie unter Oracle Database-Verschlüsselung .	25. Oktober 2017
Neuer Inhalt	SSL-Auslagerung hinzugefügt. Weitere Informationen finden Sie unter SSL/TLS-Auslagerung .	12. Oktober 2017

Änderungen	Beschreibung	Date
Neues Handbuch	In dieser Version wird Folgendes vorgestellt AWS CloudHSM	14. August 2017

Die vorliegende Übersetzung wurde maschinell erstellt. Im Falle eines Konflikts oder eines Widerspruchs zwischen dieser übersetzten Fassung und der englischen Fassung (einschließlich infolge von Verzögerungen bei der Übersetzung) ist die englische Fassung maßgeblich.