API Reference

AWS Key Management Service



API Version 2014-11-01

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

AWS Key Management Service: API Reference

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

Amazon's trademarks and trade dress may not be used in connection with any product or service that is not Amazon's, in any manner that is likely to cause confusion among customers, or in any manner that disparages or discredits Amazon. All other trademarks not owned by Amazon are the property of their respective owners, who may or may not be affiliated with, connected to, or sponsored by Amazon.

Table of Contents

Welcome **1**
Actions **4**
CancelKeyDeletion 6
Request Syntax 6
Request Parameters 6
Response Syntax 7
Response Elements 7
Errors 7
Examples 8
See Also 9
ConnectCustomKeyStore 11
Request Syntax 12
Request Parameters 13
Response Elements 13
Errors 13
See Also 15
CreateAlias 17
Request Syntax 18
Request Parameters 18
Response Elements 19
Errors 19
Examples 21
See Also 22
CreateCustomKeyStore 23
Request Syntax 24
Request Parameters 25
Response Syntax 30
Response Elements 31
Errors 31
See Also 35
CreateGrant 36
Request Syntax 37
Request Parameters 37
Response Syntax 43

Response Elements ... 43
Errors ... 44
Examples ... 45
See Also ... 46
CreateKey ... 48
Request Syntax ... 52
Request Parameters ... 52
Response Syntax ... 61
Response Elements ... 62
Errors ... 62
Examples ... 65
See Also ... 67
Decapsulate ... 68
Request Syntax ... 69
Request Parameters ... 69
Response Syntax ... 71
Response Elements ... 71
Errors ... 72
Examples ... 74
See Also ... 75
Decrypt ... 77
Request Syntax ... 78
Request Parameters ... 79
Response Syntax ... 83
Response Elements ... 83
Errors ... 84
Examples ... 87
See Also ... 88
DeleteAlias ... 89
Request Syntax ... 89
Request Parameters ... 90
Response Elements ... 90
Errors ... 90
Examples ... 91
See Also ... 92
DeleteCustomKeyStore ... 93

Request Syntax ... 94
Request Parameters ... 94
Response Elements ... 94
Errors ... 95
See Also ... 96
DeleteImportedKeyMaterial ... 97
Request Syntax ... 97
Request Parameters ... 98
Response Syntax ... 99
Response Elements ... 99
Errors ... 100
Examples ... 101
See Also ... 102
DeriveSharedSecret ... 103
Request Syntax ... 105
Request Parameters ... 105
Response Syntax ... 108
Response Elements ... 108
Errors ... 110
Examples ... 112
See Also ... 113
DescribeCustomKeyStores ... 114
Request Syntax ... 115
Request Parameters ... 115
Response Syntax ... 116
Response Elements ... 117
Errors ... 118
See Also ... 118
DescribeKey ... 120
Request Syntax ... 121
Request Parameters ... 121
Response Syntax ... 122
Response Elements ... 123
Errors ... 124
Examples ... 124
See Also ... 126

DisableKey ... 127
Request Syntax ... 127
Request Parameters ... 127
Response Elements ... 128
Errors ... 128
Examples ... 129
See Also ... 130
DisableKeyRotation ... 131
Request Syntax ... 132
Request Parameters ... 132
Response Elements ... 132
Errors ... 133
Examples ... 134
See Also ... 135
DisconnectCustomKeyStore ... 136
Request Syntax ... 137
Request Parameters ... 137
Response Elements ... 137
Errors ... 137
See Also ... 139
EnableKey ... 140
Request Syntax ... 140
Request Parameters ... 140
Response Elements ... 141
Errors ... 141
Examples ... 142
See Also ... 143
EnableKeyRotation ... 144
Request Syntax ... 145
Request Parameters ... 145
Response Elements ... 147
Errors ... 147
Examples ... 148
See Also ... 149
Encapsulate ... 150
Request Syntax ... 150

Request Parameters ... 151
Response Syntax ... 152
Response Elements ... 153
Errors ... 154
Examples ... 156
See Also ... 157
Encrypt ... 158
Request Syntax ... 159
Request Parameters ... 160
Response Syntax ... 162
Response Elements ... 163
Errors ... 163
Examples ... 165
See Also ... 166
GenerateDataKey ... 168
Request Syntax ... 170
Request Parameters ... 170
Response Syntax ... 173
Response Elements ... 174
Errors ... 175
Examples ... 177
See Also ... 178
GenerateDataKeyPair ... 179
Request Syntax ... 181
Request Parameters ... 181
Response Syntax ... 184
Response Elements ... 185
Errors ... 186
See Also ... 188
GenerateDataKeyPairWithoutPlaintext ... 190
Request Syntax ... 191
Request Parameters ... 191
Response Syntax ... 194
Response Elements ... 194
Errors ... 195
See Also ... 197

GenerateDataKeyWithoutPlaintext ... 199
Request Syntax ... 200
Request Parameters ... 200
Response Syntax ... 203
Response Elements ... 203
Errors ... 204
Examples ... 206
See Also ... 207
GenerateMac ... 208
Request Syntax ... 208
Request Parameters ... 209
Response Syntax ... 210
Response Elements ... 211
Errors ... 211
See Also ... 213
GenerateRandom ... 215
Request Syntax ... 215
Request Parameters ... 216
Response Syntax ... 217
Response Elements ... 217
Errors ... 218
Examples ... 219
See Also ... 220
GetKeyLastUsage ... 221
Request Syntax ... 222
Request Parameters ... 222
Response Syntax ... 223
Response Elements ... 223
Errors ... 224
See Also ... 225
GetKeyPolicy ... 226
Request Syntax ... 226
Request Parameters ... 226
Response Syntax ... 228
Response Elements ... 228
Errors ... 228

Examples ... 229
See Also ... 232
GetKeyRotationStatus ... 233
Request Syntax ... 234
Request Parameters ... 234
Response Syntax ... 235
Response Elements ... 235
Errors ... 236
Examples ... 237
See Also ... 238
GetParametersForImport ... 240
Request Syntax ... 241
Request Parameters ... 241
Response Syntax ... 243
Response Elements ... 244
Errors ... 244
Examples ... 246
See Also ... 247
GetPublicKey ... 249
Request Syntax ... 250
Request Parameters ... 250
Response Syntax ... 251
Response Elements ... 251
Errors ... 254
See Also ... 256
ImportKeyMaterial ... 257
Request Syntax ... 259
Request Parameters ... 260
Response Syntax ... 263
Response Elements ... 263
Errors ... 264
Examples ... 266
See Also ... 268
ListAliases ... 269
Request Syntax ... 269
Request Parameters ... 270

Response Syntax ... 271
Response Elements ... 271
Errors ... 272
Examples ... 273
See Also ... 275
ListGrants ... 276
Request Syntax ... 277
Request Parameters ... 277
Response Syntax ... 279
Response Elements ... 280
Errors ... 280
Examples ... 282
See Also ... 284
ListKeyPolicies ... 285
Request Syntax ... 285
Request Parameters ... 285
Response Syntax ... 287
Response Elements ... 287
Errors ... 288
Examples ... 289
See Also ... 291
ListKeyRotations ... 292
Request Syntax ... 292
Request Parameters ... 293
Response Syntax ... 294
Response Elements ... 295
Errors ... 295
Examples ... 297
See Also ... 298
ListKeys ... 299
Request Syntax ... 300
Request Parameters ... 300
Response Syntax ... 302
Response Elements ... 302
Errors ... 303
Examples ... 303

See Also ... 306
ListResourceTags ... 307
Request Syntax ... 307
Request Parameters ... 307
Response Syntax ... 309
Response Elements ... 309
Errors ... 310
Examples ... 311
See Also ... 312
ListRetirableGrants ... 313
Request Syntax ... 314
Request Parameters ... 314
Response Syntax ... 315
Response Elements ... 316
Errors ... 317
Examples ... 318
See Also ... 319
PutKeyPolicy ... 320
Request Syntax ... 320
Request Parameters ... 321
Response Elements ... 324
Errors ... 324
Examples ... 325
See Also ... 328
ReEncrypt ... 329
Request Syntax ... 331
Request Parameters ... 331
Response Syntax ... 336
Response Elements ... 337
Errors ... 338
Examples ... 341
See Also ... 342
ReplicateKey ... 343
Request Syntax ... 344
Request Parameters ... 345
Response Syntax ... 349

Response Elements ... 350
Errors ... 351
See Also ... 353
RetireGrant ... 354
Request Syntax ... 354
Request Parameters ... 355
Response Elements ... 356
Errors ... 356
Examples ... 357
See Also ... 358
RevokeGrant ... 360
Request Syntax ... 360
Request Parameters ... 361
Response Elements ... 362
Errors ... 362
Examples ... 363
See Also ... 364
RotateKeyOnDemand ... 365
Request Syntax ... 366
Request Parameters ... 366
Response Syntax ... 367
Response Elements ... 367
Errors ... 367
Examples ... 369
See Also ... 370
ScheduleKeyDeletion ... 371
Request Syntax ... 372
Request Parameters ... 372
Response Syntax ... 373
Response Elements ... 374
Errors ... 375
Examples ... 376
See Also ... 377
Sign ... 378
Request Syntax ... 379
Request Parameters ... 379

Response Syntax ... 383
Response Elements ... 383
Errors ... 384
See Also ... 386
TagResource ... 388
Request Syntax ... 389
Request Parameters ... 389
Response Elements ... 390
Errors ... 390
Examples ... 391
See Also ... 392
UntagResource ... 394
Request Syntax ... 394
Request Parameters ... 395
Response Elements ... 396
Errors ... 396
Examples ... 397
See Also ... 398
UpdateAlias ... 399
Request Syntax ... 400
Request Parameters ... 400
Response Elements ... 401
Errors ... 401
Examples ... 403
See Also ... 403
UpdateCustomKeyStore ... 405
Request Syntax ... 407
Request Parameters ... 407
Response Elements ... 412
Errors ... 412
See Also ... 416
UpdateKeyDescription ... 418
Request Syntax ... 418
Request Parameters ... 418
Response Elements ... 419
Errors ... 419

Examples ... 420
See Also ... 421
UpdatePrimaryRegion ... 423
Request Syntax ... 424
Request Parameters ... 424
Response Elements ... 425
Errors ... 425
See Also ... 427
Verify ... 428
Request Syntax ... 429
Request Parameters ... 429
Response Syntax ... 433
Response Elements ... 433
Errors ... 434
See Also ... 436
VerifyMac ... 437
Request Syntax ... 437
Request Parameters ... 438
Response Syntax ... 440
Response Elements ... 440
Errors ... 441
See Also ... 443
Data Types ... 444
AliasListEntry ... 445
Contents ... 445
See Also ... 446
CustomKeyStoresListEntry ... 447
Contents ... 447
See Also ... 452
GrantConstraints ... 453
Contents ... 454
See Also ... 455
GrantListEntry ... 456
Contents ... 456
See Also ... 459
KeyLastUsageData ... 460

Contents 460
See Also 461
KeyListEntry 462
Contents 462
See Also 462
KeyMetadata 463
Contents 463
See Also 470
MultiRegionConfiguration 472
Contents 472
See Also 473
MultiRegionKey 474
Contents 474
See Also 474
RecipientInfo 476
Contents 476
See Also 477
RotationsListEntry 478
Contents 478
See Also 480
Tag 482
Contents 482
See Also 483
XksKeyConfigurationType 484
Contents 484
See Also 484
XksProxyAuthenticationCredentialType 486
Contents 486
See Also 486
XksProxyConfigurationType 488
Contents 488
See Also 490
Common Parameters 491
Common Error Types 494

Welcome

AWS Key Management Service (AWS KMS) is an encryption and key management web service. This guide describes the AWS KMS operations that you can call programmatically. For general information about AWS KMS, see the *AWS Key Management Service Developer Guide*.

Note

AWS KMS has replaced the term *customer master key (CMK)* with *AWS Key Management Service key* and *KMS key*. The concept has not changed. To prevent breaking changes, AWS KMS is keeping some variations of this term.
AWS provides SDKs that consist of libraries and sample code for various programming languages and platforms (Java, Rust, Python, Ruby, .Net, macOS, Android, etc.). The SDKs provide a convenient way to create programmatic access to AWS KMS and other AWS services. For example, the SDKs take care of tasks such as signing requests (see below), managing errors, and retrying requests automatically. For more information about the AWS SDKs, including how to download and install them, see Tools for Amazon Web Services.

We recommend that you use the AWS SDKs to make programmatic API calls to AWS KMS.

If you need to use FIPS 140-2 validated cryptographic modules when communicating with AWS, use one of the FIPS endpoints in your preferred AWS Region. If you need communicate over IPv6, use the dual-stack endpoint in your preferred AWS Region. For more information see Service endpoints in the AWS Key Management Service topic of the *Amazon Web Services General Reference* and Dual-stack endpoint support in the KMS Developer Guide.

All AWS KMS API calls must be signed and be transmitted using Transport Layer Security (TLS). AWS KMS recommends you always use the latest supported TLS version. Clients must also support cipher suites with Perfect Forward Secrecy (PFS) such as Ephemeral Diffie-Hellman (DHE) or Elliptic Curve Ephemeral Diffie-Hellman (ECDHE). Most modern systems such as Java 7 and later support these modes.

Note

With current technology, brute force attacks on AWS KMS TLS cipher suite key exchange mechanisms are not feasible. However, cipher suites that use Rivest–Shamir–Adleman (RSA)

or Diffie-Hellman (DH) key exchange will be susceptible if large-scale quantum computing becomes practical. NIST IR 8547 on the NIST website establishes a timeline to phase out quantum-vulnerable algorithms by 2030 and disallow them entirely by 2035. If your applications rely on the long-term confidentiality of AWS KMS API traffic, we recommend that you use post-quantum TLS with AWS KMS. For more information about post-quantum cybersecurity, see Post-Quantum Cybersecurity Resources on the National Security Agency (NSA) website.

Signing Requests

Requests must be signed using an access key ID and a secret access key. We strongly recommend that you do not use your AWS account root access key ID and secret access key for everyday work. You can use the access key ID and secret access key for an IAM user or you can use the AWS Security Token Service (AWS STS) to generate temporary security credentials and use those to sign requests.

All AWS KMS requests must be signed with Signature Version 4.

Logging API Requests

AWS KMS supports AWS CloudTrail, a service that logs AWS API calls and related events for your AWS account and delivers them to an Amazon S3 bucket that you specify. By using the information collected by CloudTrail, you can determine what requests were made to AWS KMS, who made the request, when it was made, and so on. To learn more about CloudTrail, including how to turn it on and find your log files, see the AWS CloudTrail User Guide.

Additional Resources

For more information about credentials and request signing, see the following:

- AWS Security Credentials - This topic provides general information about the types of credentials used to access AWS.
- Temporary Security Credentials - This section of the *IAM User Guide* describes how to create and use temporary security credentials.
- Signature Version 4 Signing Process - This set of topics walks you through the process of signing a request using an access key ID and a secret access key.

Commonly Used API Operations

Of the API operations discussed in this guide, the following will prove the most useful for most applications. You will likely perform operations other than these, such as creating keys and assigning policies, by using the console.

- Encrypt
- Decrypt
- GenerateDataKey
- GenerateDataKeyWithoutPlaintext

This document was last published on November 23, 2026.

Actions

The following actions are supported:

- CancelKeyDeletion
- ConnectCustomKeyStore
- CreateAlias
- CreateCustomKeyStore
- CreateGrant
- CreateKey
- Decapsulate
- Decrypt
- DeleteAlias
- DeleteCustomKeyStore
- DeleteImportedKeyMaterial
- DeriveSharedSecret
- DescribeCustomKeyStores
- DescribeKey
- DisableKey
- DisableKeyRotation
- DisconnectCustomKeyStore
- EnableKey
- EnableKeyRotation
- Encapsulate
- Encrypt
- GenerateDataKey
- GenerateDataKeyPair
- GenerateDataKeyPairWithoutPlaintext
- GenerateDataKeyWithoutPlaintext
- GenerateMac
- GenerateRandom

- GetKeyLastUsage
- GetKeyPolicy
- GetKeyRotationStatus
- GetParametersForImport
- GetPublicKey
- ImportKeyMaterial
- ListAliases
- ListGrants
- ListKeyPolicies
- ListKeyRotations
- ListKeys
- ListResourceTags
- ListRetirableGrants
- PutKeyPolicy
- ReEncrypt
- ReplicateKey
- RetireGrant
- RevokeGrant
- RotateKeyOnDemand
- ScheduleKeyDeletion
- Sign
- TagResource
- UntagResource
- UpdateAlias
- UpdateCustomKeyStore
- UpdateKeyDescription
- UpdatePrimaryRegion
- Verify
- VerifyMac

CancelKeyDeletion

Cancels the deletion of a KMS key. When this operation succeeds, the key state of the KMS key is `Disabled`. To enable the KMS key, use EnableKey.

For more information about scheduling and canceling deletion of a KMS key, see Deleting KMS keys in the *AWS Key Management Service Developer Guide*.

The KMS key that you use for this operation must be in a compatible key state. For details, see Key states of AWS KMS keys in the *AWS Key Management Service Developer Guide*.

Cross-account use: No. You cannot perform this operation on a KMS key in a different AWS account.

Required permissions: kms:CancelKeyDeletion (key policy)

Related operations: ScheduleKeyDeletion

Eventual consistency: The AWS KMS API follows an eventual consistency model. For more information, see AWS KMS eventual consistency.

Request Syntax

```
{
   "KeyId": "string"
}
```

Request Parameters

For information about the parameters that are common to all actions, see Common Parameters.

The request accepts the following data in JSON format.

> **Note**
>
> In the following list, the required parameters are described first.

KeyId

Identifies the KMS key whose deletion is being canceled.

Specify the key ID or key ARN of the KMS key.

For example:

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

To get the key ID and key ARN for a KMS key, use ListKeys or DescribeKey.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 2048.

Required: Yes

Response Syntax

```
{
   "KeyId": "string"
}
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

KeyId

The Amazon Resource Name (key ARN) of the KMS key whose deletion is canceled.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 2048.

Errors

For information about the errors that are common to all actions, see Common Error Types.

DependencyTimeoutException

The system timed out while trying to fulfill the request. You can retry the request.

HTTP Status Code: 500

InvalidArnException

The request was rejected because a specified ARN, or an ARN in a key policy, is not valid.

HTTP Status Code: 400

KMSInternalException

The request was rejected because an internal exception occurred. The request can be retried.

HTTP Status Code: 500

KMSInvalidStateException

The request was rejected because the state of the specified resource is not valid for this request.

This exceptions means one of the following:

- The key state of the KMS key is not compatible with the operation.

  To find the key state, use the DescribeKey operation. For more information about which key states are compatible with each AWS KMS operation, see Key states of AWS KMS keys in the AWS Key Management Service Developer Guide .
- For cryptographic operations on KMS keys in custom key stores, this exception represents a general failure with many possible causes. To identify the cause, see the error message that accompanies the exception.

HTTP Status Code: 400

NotFoundException

The request was rejected because the specified entity or resource could not be found.

HTTP Status Code: 400

Examples

Example Request

The following example is formatted for legibility.

```
POST / HTTP/1.1
Host: kms.us-east-2.amazonaws.com
Content-Length: 48
X-Amz-Target: TrentService.CancelKeyDeletion
X-Amz-Date: 20161025T182658Z
Content-Type: application/x-amz-json-1.1
Authorization: AWS4-HMAC-SHA256\
 Credential=AKIAI44QH8DHBEXAMPLE/20161025/us-east-2/kms/aws4_request,\
 SignedHeaders=content-type;host;x-amz-date;x-amz-target,\
 Signature=1a600d3edf52b2c14bd6fb6fa44c6ca591bdcb02931fd9cac2e8aa66bd52e3bf

{"KeyId":"1234abcd-12ab-34cd-56ef-1234567890ab"}
```

Example Response

This example illustrates one usage of CancelKeyDeletion.

```
HTTP/1.1 200 OK
Server: Server
Date: Tue, 25 Oct 2016 18:27:01 GMT
Content-Type: application/x-amz-json-1.1
Content-Length: 87
Connection: keep-alive
x-amzn-RequestId: 9f3b3cb8-9ae0-11e6-ac6b-03478315fc57

{"KeyId":"arn:aws:kms:us-east-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"}
```

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface V2
- AWS SDK for .NET V4
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin

- AWS SDK for PHP V3
- AWS SDK for Python (Boto3)
- AWS SDK for Ruby V3

ConnectCustomKeyStore

Connects or reconnects a custom key store to its backing key store. For an AWS CloudHSM key store, `ConnectCustomKeyStore` connects the key store to its associated AWS CloudHSM cluster. For an external key store, `ConnectCustomKeyStore` connects the key store to the external key store proxy that communicates with your external key manager.

The custom key store must be connected before you can create KMS keys in the key store or use the KMS keys it contains. You can disconnect and reconnect a custom key store at any time.

The connection process for a custom key store can take an extended amount of time to complete. This operation starts the connection process, but it does not wait for it to complete. When it succeeds, this operation quickly returns an HTTP 200 response and a JSON object with no properties. However, this response does not indicate that the custom key store is connected. To get the connection state of the custom key store, use the DescribeCustomKeyStores operation.

This operation is part of the custom key stores feature in AWS KMS, which combines the convenience and extensive integration of AWS KMS with the isolation and control of a key store that you own and manage.

The `ConnectCustomKeyStore` operation might fail for various reasons. To find the reason, use the DescribeCustomKeyStores operation and see the `ConnectionErrorCode` in the response. For help interpreting the `ConnectionErrorCode`, see CustomKeyStoresListEntry.

To fix the failure, use the DisconnectCustomKeyStore operation to disconnect the custom key store, correct the error, use the UpdateCustomKeyStore operation if necessary, and then use `ConnectCustomKeyStore` again.

AWS CloudHSM key store

During the connection process for an AWS CloudHSM key store, AWS KMS finds the AWS CloudHSM cluster that is associated with the custom key store, creates the connection infrastructure, connects to the cluster, logs into the AWS CloudHSM client as the `kmsuser` CU, and rotates its password.

To connect an AWS CloudHSM key store, its associated AWS CloudHSM cluster must have at least one active HSM. To get the number of active HSMs in a cluster, use the DescribeClusters operation. To add HSMs to the cluster, use the CreateHsm operation. Also, the `kmsuser` crypto user (CU) must not be logged into the cluster. This prevents AWS KMS from using this account to log in.

If you are having trouble connecting or disconnecting a AWS CloudHSM key store, see Troubleshooting an AWS CloudHSM key store in the *AWS Key Management Service Developer Guide*.

External key store

When you connect an external key store that uses public endpoint connectivity, AWS KMS tests its ability to communicate with your external key manager by sending a request via the external key store proxy.

When you connect to an external key store that uses VPC endpoint service connectivity, AWS KMS establishes the networking elements that it needs to communicate with your external key manager via the external key store proxy. This includes creating an interface endpoint to the VPC endpoint service and a private hosted zone for traffic between AWS KMS and the VPC endpoint service.

To connect an external key store, AWS KMS must be able to connect to the external key store proxy, the external key store proxy must be able to communicate with your external key manager, and the external key manager must be available for cryptographic operations.

If you are having trouble connecting or disconnecting an external key store, see Troubleshooting an external key store in the *AWS Key Management Service Developer Guide*.

Cross-account use: No. You cannot perform this operation on a custom key store in a different AWS account.

Required permissions: kms:ConnectCustomKeyStore (IAM policy)

Related operations

- CreateCustomKeyStore
- DeleteCustomKeyStore
- DescribeCustomKeyStores
- DisconnectCustomKeyStore
- UpdateCustomKeyStore

Eventual consistency: The AWS KMS API follows an eventual consistency model. For more information, see AWS KMS eventual consistency.

Request Syntax

```
{
   "CustomKeyStoreId": "string"
```

```
}
```

Request Parameters

For information about the parameters that are common to all actions, see Common Parameters.

The request accepts the following data in JSON format.

> **Note**
>
> In the following list, the required parameters are described first.

CustomKeyStoreId

Enter the key store ID of the custom key store that you want to connect. To find the ID of a custom key store, use the DescribeCustomKeyStores operation.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 64.

Required: Yes

Response Elements

If the action is successful, the service sends back an HTTP 200 response with an empty HTTP body.

Errors

For information about the errors that are common to all actions, see Common Error Types.

CloudHsmClusterInvalidConfigurationException

The request was rejected because the associated AWS CloudHSM cluster did not meet the configuration requirements for an AWS CloudHSM key store.

- The AWS CloudHSM cluster must be configured with private subnets in at least two different Availability Zones in the Region.
- The security group for the cluster (cloudhsm-cluster-*<cluster-id>*-sg) must include inbound rules and outbound rules that allow TCP traffic on ports 2223-2225. The **Source** in the

inbound rules and the **Destination** in the outbound rules must match the security group ID. These rules are set by default when you create the AWS CloudHSM cluster. Do not delete or change them. To get information about a particular security group, use the DescribeSecurityGroups operation.

- The AWS CloudHSM cluster must contain at least as many HSMs as the operation requires. To add HSMs, use the AWS CloudHSM CreateHsm operation.

  For the CreateCustomKeyStore, UpdateCustomKeyStore, and CreateKey operations, the AWS CloudHSM cluster must have at least two active HSMs, each in a different Availability Zone. For the ConnectCustomKeyStore operation, the AWS CloudHSM must contain at least one active HSM.

For information about the requirements for an AWS CloudHSM cluster that is associated with an AWS CloudHSM key store, see Assemble the Prerequisites in the *AWS Key Management Service Developer Guide*. For information about creating a private subnet for an AWS CloudHSM cluster, see Create a Private Subnet in the *AWS CloudHSM User Guide*. For information about cluster security groups, see Configure a Default Security Group in the AWS CloudHSM User Guide .

HTTP Status Code: 400

CloudHsmClusterNotActiveException

The request was rejected because the AWS CloudHSM cluster associated with the AWS CloudHSM key store is not active. Initialize and activate the cluster and try the command again. For detailed instructions, see Getting Started in the *AWS CloudHSM User Guide*.

HTTP Status Code: 400

CustomKeyStoreInvalidStateException

The request was rejected because of the `ConnectionState` of the custom key store. To get the `ConnectionState` of a custom key store, use the DescribeCustomKeyStores operation.

This exception is thrown under the following conditions:

- You requested the ConnectCustomKeyStore operation on a custom key store with a `ConnectionState` of `DISCONNECTING` or `FAILED`. This operation is valid for all other `ConnectionState` values. To reconnect a custom key store in a `FAILED` state, disconnect it (DisconnectCustomKeyStore), then connect it (`ConnectCustomKeyStore`).
- You requested the CreateKey operation in a custom key store that is not connected. This operations is valid only when the custom key store `ConnectionState` is `CONNECTED`.

- You requested the DisconnectCustomKeyStore operation on a custom key store with a `ConnectionState` of `DISCONNECTING` or `DISCONNECTED`. This operation is valid for all other `ConnectionState` values.
- You requested the UpdateCustomKeyStore or DeleteCustomKeyStore operation on a custom key store that is not disconnected. `UpdateCustomKeyStore` can be called on a custom key store in the `CONNECTED` state only to update `NewCustomKeyStoreName`. For all other properties, the custom key store `ConnectionState` must be `DISCONNECTED`.
- You requested the GenerateRandom operation in an AWS CloudHSM key store that is not connected. This operation is valid only when the AWS CloudHSM key store `ConnectionState` is `CONNECTED`.

HTTP Status Code: 400

CustomKeyStoreNotFoundException

The request was rejected because AWS KMS cannot find a custom key store with the specified key store name or ID.

HTTP Status Code: 400

KMSInternalException

The request was rejected because an internal exception occurred. The request can be retried.

HTTP Status Code: 500

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface V2
- AWS SDK for .NET V4
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin

- AWS SDK for PHP V3
- AWS SDK for Python (Boto3)
- AWS SDK for Ruby V3

CreateAlias

Creates a friendly name for a KMS key.

> **Note**
>
> Adding, deleting, or updating an alias can allow or deny permission to the KMS key. For details, see ABAC for AWS KMS in the *AWS Key Management Service Developer Guide*.

You can use an alias to identify a KMS key in the AWS KMS console, in the DescribeKey operation and in cryptographic operations, such as Encrypt and GenerateDataKey. You can also change the KMS key that's associated with the alias (UpdateAlias) or delete the alias (DeleteAlias) at any time. These operations don't affect the underlying KMS key.

You can associate the alias with any customer managed key in the same AWS Region. Each alias is associated with only one KMS key at a time, but a KMS key can have multiple aliases. A valid KMS key is required. You can't create an alias without a KMS key.

The alias must be unique in the account and Region, but you can have aliases with the same name in different Regions. For detailed information about aliases, see Aliases in AWS KMS in the *AWS Key Management Service Developer Guide*.

This operation does not return a response. To get the alias that you created, use the ListAliases operation.

The KMS key that you use for this operation must be in a compatible key state. For details, see Key states of AWS KMS keys in the *AWS Key Management Service Developer Guide*.

Cross-account use: No. You cannot perform this operation on an alias in a different AWS account.

Required permissions

- kms:CreateAlias on the alias (IAM policy).
- kms:CreateAlias on the KMS key (key policy).

For details, see Controlling access to aliases in the *AWS Key Management Service Developer Guide*.

Related operations:

- DeleteAlias
- ListAliases
- UpdateAlias

Eventual consistency: The AWS KMS API follows an eventual consistency model. For more information, see AWS KMS eventual consistency.

Request Syntax

```
{
   "AliasName": "string",
   "TargetKeyId": "string"
}
```

Request Parameters

For information about the parameters that are common to all actions, see Common Parameters.

The request accepts the following data in JSON format.

> **Note**
>
> In the following list, the required parameters are described first.

AliasName

Specifies the alias name. This value must begin with `alias/` followed by a name, such as `alias/ExampleAlias`.

> **Important**
>
> Do not include confidential or sensitive information in this field. This field may be displayed in plaintext in CloudTrail logs and other output.

The `AliasName` value must be string of 1-256 characters. It can contain only alphanumeric characters, forward slashes (/), underscores (_), and dashes (-). The alias name cannot begin with `alias/aws/`. The `alias/aws/` prefix is reserved for AWS managed keys.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 256.

Pattern: `^alias/[a-zA-Z0-9/_-]+$`

Required: Yes

TargetKeyId

Associates the alias with the specified customer managed key. The KMS key must be in the same AWS Region.

A valid key ID is required. If you supply a null or empty string value, this operation returns an error.

For help finding the key ID and ARN, see Find the key ID and key ARN in the AWS Key Management Service Developer Guide .

Specify the key ID or key ARN of the KMS key.

For example:

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

To get the key ID and key ARN for a KMS key, use ListKeys or DescribeKey.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 2048.

Required: Yes

Response Elements

If the action is successful, the service sends back an HTTP 200 response with an empty HTTP body.

Errors

For information about the errors that are common to all actions, see Common Error Types.

AlreadyExistsException

The request was rejected because it attempted to create a resource that already exists.

HTTP Status Code: 400

DependencyTimeoutException

The system timed out while trying to fulfill the request. You can retry the request.

HTTP Status Code: 500

InvalidAliasNameException

The request was rejected because the specified alias name is not valid.

HTTP Status Code: 400

KMSInternalException

The request was rejected because an internal exception occurred. The request can be retried.

HTTP Status Code: 500

KMSInvalidStateException

The request was rejected because the state of the specified resource is not valid for this request.

This exceptions means one of the following:

- The key state of the KMS key is not compatible with the operation.

  To find the key state, use the DescribeKey operation. For more information about which key states are compatible with each AWS KMS operation, see Key states of AWS KMS keys in the AWS Key Management Service Developer Guide .
- For cryptographic operations on KMS keys in custom key stores, this exception represents a general failure with many possible causes. To identify the cause, see the error message that accompanies the exception.

HTTP Status Code: 400

LimitExceededException

The request was rejected because a length constraint or quota was exceeded. For more information, see Quotas in the *AWS Key Management Service Developer Guide*.

HTTP Status Code: 400

NotFoundException

The request was rejected because the specified entity or resource could not be found.

HTTP Status Code: 400

Examples

Example Request

The following example is formatted for legibility.

```
POST / HTTP/1.1
Host: kms.us-west-2.amazonaws.com
Content-Length: 87
X-Amz-Target: TrentService.CreateAlias
X-Amz-Date: 20160517T204220Z
Content-Type: application/x-amz-json-1.1
Authorization: AWS4-HMAC-SHA256\
 Credential=AKIAI44QH8DHBEXAMPLE/20160517/us-west-2/kms/aws4_request,\
 SignedHeaders=content-type;host;x-amz-date;x-amz-target,\
 Signature=ca7bcf1e8d5364dc3f0d881c05bdadf36f498c6c6a8b576a060142d9b2199123

{
  "TargetKeyId": "1234abcd-12ab-34cd-56ef-1234567890ab",
  "AliasName": "alias/ExampleAlias"
}
```

Example Response

This example illustrates one usage of CreateAlias.

```
HTTP/1.1 200 OK
Server: Server
Date: Tue, 17 May 2016 20:42:25 GMT
Content-Type: application/x-amz-json-1.1
Content-Length: 0
Connection: keep-alive
x-amzn-RequestId: dcb07ca7-1c6f-11e6-8540-77c363708b91
```

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface V2
- AWS SDK for .NET V4
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- AWS SDK for PHP V3
- AWS SDK for Python (Boto3)
- AWS SDK for Ruby V3

CreateCustomKeyStore

Creates a custom key store backed by a key store that you own and manage. When you use a KMS key in a custom key store for a cryptographic operation, the cryptographic operation is actually performed in your key store using your keys. AWS KMS supports AWS CloudHSM key stores backed by an AWS CloudHSM cluster and external key stores backed by an external key store proxy and external key manager outside of AWS.

This operation is part of the custom key stores feature in AWS KMS, which combines the convenience and extensive integration of AWS KMS with the isolation and control of a key store that you own and manage.

Before you create the custom key store, the required elements must be in place and operational. We recommend that you use the test tools that AWS KMS provides to verify the configuration your external key store proxy. For details about the required elements and verification tests, see Assemble the prerequisites (for AWS CloudHSM key stores) or Assemble the prerequisites (for external key stores) in the *AWS Key Management Service Developer Guide*.

To create a custom key store, use the following parameters.

- To create an AWS CloudHSM key store, specify the `CustomKeyStoreName`, `CloudHsmClusterId`, `KeyStorePassword`, and `TrustAnchorCertificate`. The `CustomKeyStoreType` parameter is optional for AWS CloudHSM key stores. If you include it, set it to the default value, `AWS_CLOUDHSM`. For help with failures, see Troubleshooting an AWS CloudHSM key store in the *AWS Key Management Service Developer Guide*.
- To create an external key store, specify the `CustomKeyStoreName` and a `CustomKeyStoreType` of `EXTERNAL_KEY_STORE`. Also, specify values for `XksProxyConnectivity`, `XksProxyAuthenticationCredential`, `XksProxyUriEndpoint`, and `XksProxyUriPath`. If your `XksProxyConnectivity` value is `VPC_ENDPOINT_SERVICE`, specify the `XksProxyVpcEndpointServiceName` parameter. For help with failures, see Troubleshooting an external key store in the *AWS Key Management Service Developer Guide*.

> **Note**
>
> For external key stores:
> Some external key managers provide a simpler method for creating an external key store. For details, see your external key manager documentation.

When creating an external key store in the AWS KMS console, you can upload a JSON-based proxy configuration file with the desired values. You cannot use a proxy configuration with the `CreateCustomKeyStore` operation. However, you can use the values in the file to help you determine the correct values for the `CreateCustomKeyStore` parameters.

When the operation completes successfully, it returns the ID of the new custom key store. Before you can use your new custom key store, you need to use the ConnectCustomKeyStore operation to connect a new AWS CloudHSM key store to its AWS CloudHSM cluster, or to connect a new external key store to the external key store proxy for your external key manager. Even if you are not going to use your custom key store immediately, you might want to connect it to verify that all settings are correct and then disconnect it until you are ready to use it.

Cross-account use: No. You cannot perform this operation on a custom key store in a different AWS account.

Required permissions: kms:CreateCustomKeyStore (IAM policy).

Related operations:

- ConnectCustomKeyStore
- DeleteCustomKeyStore
- DescribeCustomKeyStores
- DisconnectCustomKeyStore
- UpdateCustomKeyStore

Eventual consistency: The AWS KMS API follows an eventual consistency model. For more information, see AWS KMS eventual consistency.

Request Syntax

```
{
   "CloudHsmClusterId": "string",
   "CustomKeyStoreName": "string",
   "CustomKeyStoreType": "string",
   "KeyStorePassword": "string",
   "TrustAnchorCertificate": "string",
```

```
    "XksProxyAuthenticationCredential": {
       "AccessKeyId": "string",
       "RawSecretAccessKey": "string"
    },
    "XksProxyConnectivity": "string",
    "XksProxyUriEndpoint": "string",
    "XksProxyUriPath": "string",
    "XksProxyVpcEndpointServiceName": "string",
    "XksProxyVpcEndpointServiceOwner": "string"
}
```

Request Parameters

For information about the parameters that are common to all actions, see Common Parameters.

The request accepts the following data in JSON format.

> **Note**
>
> In the following list, the required parameters are described first.

CustomKeyStoreName

Specifies a friendly name for the custom key store. The name must be unique in your AWS account and Region. This parameter is required for all custom key stores.

> **Important**
>
> Do not include confidential or sensitive information in this field. This field may be displayed in plaintext in CloudTrail logs and other output.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 256.

Required: Yes

CloudHsmClusterId

Identifies the AWS CloudHSM cluster for an AWS CloudHSM key store. This parameter is required for custom key stores with `CustomKeyStoreType` of `AWS_CLOUDHSM`.

Enter the cluster ID of any active AWS CloudHSM cluster that is not already associated with a custom key store. To find the cluster ID, use the DescribeClusters operation.

Type: String

Length Constraints: Minimum length of 19. Maximum length of 24.

Pattern: `cluster-[2-7a-zA-Z]{11,16}`

Required: No

CustomKeyStoreType

Specifies the type of custom key store. The default value is `AWS_CLOUDHSM`.

For a custom key store backed by an AWS CloudHSM cluster, omit the parameter or enter `AWS_CLOUDHSM`. For a custom key store backed by an external key manager outside of AWS, enter `EXTERNAL_KEY_STORE`. You cannot change this property after the key store is created.

Type: String

Valid Values: `AWS_CLOUDHSM | EXTERNAL_KEY_STORE`

Required: No

KeyStorePassword

Specifies the `kmsuser` password for an AWS CloudHSM key store. This parameter is required for custom key stores with a `CustomKeyStoreType` of `AWS_CLOUDHSM`.

Enter the password of the `kmsuser` crypto user (CU) account in the specified AWS CloudHSM cluster. AWS KMS logs into the cluster as this user to manage key material on your behalf.

The password must be a string of 7 to 32 characters. Its value is case sensitive.

This parameter tells AWS KMS the `kmsuser` account password; it does not change the password in the AWS CloudHSM cluster.

Type: String

Length Constraints: Minimum length of 7. Maximum length of 32.

Required: No

TrustAnchorCertificate

Specifies the certificate for an AWS CloudHSM key store. This parameter is required for custom key stores with a `CustomKeyStoreType` of `AWS_CLOUDHSM`.

Enter the content of the trust anchor certificate for the AWS CloudHSM cluster. This is the content of the `customerCA.crt` file that you created when you initialized the cluster.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 5000.

Required: No

XksProxyAuthenticationCredential

Specifies an authentication credential for the external key store proxy (XKS proxy). This parameter is required for all custom key stores with a `CustomKeyStoreType` of `EXTERNAL_KEY_STORE`.

The `XksProxyAuthenticationCredential` has two required elements: `RawSecretAccessKey`, a secret key, and `AccessKeyId`, a unique identifier for the `RawSecretAccessKey`. For character requirements, see XksProxyAuthenticationCredentialType.

AWS KMS uses this authentication credential to sign requests to the external key store proxy on your behalf. This credential is unrelated to AWS Identity and Access Management (IAM) and AWS credentials.

This parameter doesn't set or change the authentication credentials on the XKS proxy. It just tells AWS KMS the credential that you established on your external key store proxy. If you rotate your proxy authentication credential, use the UpdateCustomKeyStore operation to provide the new credential to AWS KMS.

Type: XksProxyAuthenticationCredentialType object

Required: No

XksProxyConnectivity

Indicates how AWS KMS communicates with the external key store proxy. This parameter is required for custom key stores with a `CustomKeyStoreType` of `EXTERNAL_KEY_STORE`.

If the external key store proxy uses a public endpoint, specify `PUBLIC_ENDPOINT`. If the external key store proxy uses a Amazon VPC endpoint service for communication with AWS KMS, specify `VPC_ENDPOINT_SERVICE`. For help making this choice, see Choosing a connectivity option in the *AWS Key Management Service Developer Guide*.

An Amazon VPC endpoint service keeps your communication with AWS KMS in a private address space entirely within AWS, but it requires more configuration, including establishing a Amazon VPC with multiple subnets, a VPC endpoint service, a network load balancer, and a verified private DNS name. A public endpoint is simpler to set up, but it might be slower and might not fulfill your security requirements. You might consider testing with a public endpoint, and then establishing a VPC endpoint service for production tasks. Note that this choice does not determine the location of the external key store proxy. Even if you choose a VPC endpoint service, the proxy can be hosted within the VPC or outside of AWS such as in your corporate data center.

Type: String

Valid Values: `PUBLIC_ENDPOINT | VPC_ENDPOINT_SERVICE`

Required: No

XksProxyUriEndpoint

Specifies the endpoint that AWS KMS uses to send requests to the external key store proxy (XKS proxy). This parameter is required for custom key stores with a `CustomKeyStoreType` of `EXTERNAL_KEY_STORE`.

The protocol must be HTTPS. AWS KMS communicates on port 443. Do not specify the port in the `XksProxyUriEndpoint` value.

For external key stores with `XksProxyConnectivity` value of `VPC_ENDPOINT_SERVICE`, specify `https://` followed by the private DNS name of the VPC endpoint service.

For external key stores with `PUBLIC_ENDPOINT` connectivity, this endpoint must be reachable before you create the custom key store. AWS KMS connects to the external key store proxy while creating the custom key store. For external key stores with `VPC_ENDPOINT_SERVICE` connectivity, AWS KMS connects when you call the ConnectCustomKeyStore operation.

The value of this parameter must begin with `https://`. The remainder can contain upper and lower case letters (A-Z and a-z), numbers (0-9), dots (`.`), and hyphens (-). Additional slashes (`/` and `\`) are not permitted.

Uniqueness requirements:

- The combined `XksProxyUriEndpoint` and `XksProxyUriPath` values must be unique in the AWS account and Region.
- An external key store with `PUBLIC_ENDPOINT` connectivity cannot use the same `XksProxyUriEndpoint` value as an external key store with `VPC_ENDPOINT_SERVICE` connectivity in this AWS Region.
- Each external key store with `VPC_ENDPOINT_SERVICE` connectivity must have its own private DNS name. The `XksProxyUriEndpoint` value for external key stores with `VPC_ENDPOINT_SERVICE` connectivity (private DNS name) must be unique in the AWS account and Region.

Type: String

Length Constraints: Minimum length of 10. Maximum length of 128.

Pattern: `^https://[a-zA-Z0-9.-]+$`

Required: No

XksProxyUriPath

Specifies the base path to the proxy APIs for this external key store. To find this value, see the documentation for your external key store proxy. This parameter is required for all custom key stores with a `CustomKeyStoreType` of `EXTERNAL_KEY_STORE`.

The value must start with `/` and must end with `/kms/xks/v1` where `v1` represents the version of the AWS KMS external key store proxy API. This path can include an optional prefix between the required elements such as */prefix*`/kms/xks/v1`.

Uniqueness requirements:

- The combined `XksProxyUriEndpoint` and `XksProxyUriPath` values must be unique in the AWS account and Region.

Type: String

Length Constraints: Minimum length of 10. Maximum length of 128.

Pattern: `^(/[a-zA-Z0-9\/_-]+/kms/xks/v\d{1,2})$|^(/kms/xks/v\d{1,2})$`

Required: No

XksProxyVpcEndpointServiceName

Specifies the name of the Amazon VPC endpoint service for interface endpoints that is used to communicate with your external key store proxy (XKS proxy). This parameter is required when the value of `CustomKeyStoreType` is `EXTERNAL_KEY_STORE` and the value of `XksProxyConnectivity` is `VPC_ENDPOINT_SERVICE`.

The Amazon VPC endpoint service must fulfill all requirements for use with an external key store.

Uniqueness requirements:

- External key stores with `VPC_ENDPOINT_SERVICE` connectivity can share an Amazon VPC, but each external key store must have its own VPC endpoint service and private DNS name.

Type: String

Length Constraints: Minimum length of 20. Maximum length of 64.

Pattern: `^(com|eu)\.amazonaws\.vpce\.([a-z]+-){2,3}\d+\.vpce-svc-[0-9a-z]+$`

Required: No

XksProxyVpcEndpointServiceOwner

Specifies the AWS account ID that owns the Amazon VPC service endpoint for the interface that is used to communicate with your external key store proxy (XKS proxy). This parameter is optional. If not provided, the AWS account ID calling the action will be used.

Type: String

Length Constraints: Fixed length of 12.

Pattern: `[0-9]{12}`

Required: No

Response Syntax

```
{
   "CustomKeyStoreId": "string"
```

```
}
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

CustomKeyStoreId

A unique identifier for the new custom key store.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 64.

Errors

For information about the errors that are common to all actions, see Common Error Types.

CloudHsmClusterInUseException

The request was rejected because the specified AWS CloudHSM cluster is already associated with an AWS CloudHSM key store in the account, or it shares a backup history with an AWS CloudHSM key store in the account. Each AWS CloudHSM key store in the account must be associated with a different AWS CloudHSM cluster.

AWS CloudHSM clusters that share a backup history have the same cluster certificate. To view the cluster certificate of an AWS CloudHSM cluster, use the DescribeClusters operation.

HTTP Status Code: 400

CloudHsmClusterInvalidConfigurationException

The request was rejected because the associated AWS CloudHSM cluster did not meet the configuration requirements for an AWS CloudHSM key store.

- The AWS CloudHSM cluster must be configured with private subnets in at least two different Availability Zones in the Region.
- The security group for the cluster (cloudhsm-cluster-*<cluster-id>*-sg) must include inbound rules and outbound rules that allow TCP traffic on ports 2223-2225. The **Source** in the

inbound rules and the **Destination** in the outbound rules must match the security group ID. These rules are set by default when you create the AWS CloudHSM cluster. Do not delete or change them. To get information about a particular security group, use the DescribeSecurityGroups operation.

- The AWS CloudHSM cluster must contain at least as many HSMs as the operation requires. To add HSMs, use the AWS CloudHSM CreateHsm operation.

  For the CreateCustomKeyStore, UpdateCustomKeyStore, and CreateKey operations, the AWS CloudHSM cluster must have at least two active HSMs, each in a different Availability Zone. For the ConnectCustomKeyStore operation, the AWS CloudHSM must contain at least one active HSM.

For information about the requirements for an AWS CloudHSM cluster that is associated with an AWS CloudHSM key store, see Assemble the Prerequisites in the *AWS Key Management Service Developer Guide*. For information about creating a private subnet for an AWS CloudHSM cluster, see Create a Private Subnet in the *AWS CloudHSM User Guide*. For information about cluster security groups, see Configure a Default Security Group in the AWS CloudHSM User Guide .

HTTP Status Code: 400

CloudHsmClusterNotActiveException

The request was rejected because the AWS CloudHSM cluster associated with the AWS CloudHSM key store is not active. Initialize and activate the cluster and try the command again. For detailed instructions, see Getting Started in the *AWS CloudHSM User Guide*.

HTTP Status Code: 400

CloudHsmClusterNotFoundException

The request was rejected because AWS KMS cannot find the AWS CloudHSM cluster with the specified cluster ID. Retry the request with a different cluster ID.

HTTP Status Code: 400

CustomKeyStoreNameInUseException

The request was rejected because the specified custom key store name is already assigned to another custom key store in the account. Try again with a custom key store name that is unique in the account.

HTTP Status Code: 400

IncorrectTrustAnchorException

The request was rejected because the trust anchor certificate in the request to create an AWS CloudHSM key store is not the trust anchor certificate for the specified AWS CloudHSM cluster.

When you initialize the AWS CloudHSM cluster, you create the trust anchor certificate and save it in the `customerCA.crt` file.

HTTP Status Code: 400

KMSInternalException

The request was rejected because an internal exception occurred. The request can be retried.

HTTP Status Code: 500

LimitExceededException

The request was rejected because a length constraint or quota was exceeded. For more information, see Quotas in the *AWS Key Management Service Developer Guide*.

HTTP Status Code: 400

XksProxyIncorrectAuthenticationCredentialException

The request was rejected because the proxy credentials failed to authenticate to the specified external key store proxy. The specified external key store proxy rejected a status request from AWS KMS due to invalid credentials. This can indicate an error in the credentials or in the identification of the external key store proxy.

HTTP Status Code: 400

XksProxyInvalidConfigurationException

The request was rejected because the external key store proxy is not configured correctly. To identify the cause, see the error message that accompanies the exception.

HTTP Status Code: 400

XksProxyInvalidResponseException

AWS KMS cannot interpret the response it received from the external key store proxy. The problem might be a poorly constructed response, but it could also be a transient network issue. If you see this error repeatedly, report it to the proxy vendor.

HTTP Status Code: 400

XksProxyUriEndpointInUseException

The request was rejected because the `XksProxyUriEndpoint` is already associated with another external key store in this AWS Region. To identify the cause, see the error message that accompanies the exception.

HTTP Status Code: 400

XksProxyUriInUseException

The request was rejected because the concatenation of the `XksProxyUriEndpoint` and `XksProxyUriPath` is already associated with another external key store in this AWS Region. Each external key store in a Region must use a unique external key store proxy API address.

HTTP Status Code: 400

XksProxyUriUnreachableException

AWS KMS was unable to reach the specified `XksProxyUriPath`. The path must be reachable before you create the external key store or update its settings.

This exception is also thrown when the external key store proxy response to a `GetHealthStatus` request indicates that all external key manager instances are unavailable.

HTTP Status Code: 400

XksProxyVpcEndpointServiceInUseException

The request was rejected because the specified Amazon VPC endpoint service is already associated with another external key store in this AWS Region. Each external key store in a Region must use a different Amazon VPC endpoint service.

HTTP Status Code: 400

XksProxyVpcEndpointServiceInvalidConfigurationException

The request was rejected because the Amazon VPC endpoint service configuration does not fulfill the requirements for an external key store. To identify the cause, see the error message that accompanies the exception and review the requirements for Amazon VPC endpoint service connectivity for an external key store.

HTTP Status Code: 400

XksProxyVpcEndpointServiceNotFoundException

The request was rejected because AWS KMS could not find the specified VPC endpoint service. Use DescribeCustomKeyStores to verify the VPC endpoint service name for the external key store. Also, confirm that the `Allow principals` list for the VPC endpoint service includes the AWS KMS service principal for the Region, such as `cks.kms.us-east-1.amazonaws.com`.

HTTP Status Code: 400

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface V2
- AWS SDK for .NET V4
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- AWS SDK for PHP V3
- AWS SDK for Python (Boto3)
- AWS SDK for Ruby V3

CreateGrant

Adds a grant to a KMS key.

A *grant* is a policy instrument that allows AWS principals to use KMS keys in cryptographic operations. It also can allow them to view a KMS key (DescribeKey) and create and manage grants. When authorizing access to a KMS key, grants are considered along with key policies and IAM policies. Grants are often used for temporary permissions because you can create one, use its permissions, and delete it without changing your key policies or IAM policies.

You can create a grant for an AWS principal (IAM user, IAM role, or AWS account) by specifying the `GranteePrincipal` parameter. You can also create a grant for an AWS service principal by specifying the `GranteeServicePrincipal` parameter.

For detailed information about grants, including grant terminology, see Grants in AWS KMS in the AWS Key Management Service Developer Guide . For examples of creating grants in several programming languages, see Use CreateGrant with an AWS SDK or CLI.

The `CreateGrant` operation returns a `GrantToken` and a `GrantId`.

- When you create, retire, or revoke a grant, there might be a brief delay, usually less than five minutes, until the grant is available throughout AWS KMS. This state is known as *eventual consistency*. Once the grant has achieved eventual consistency, the grantee principal can use the permissions in the grant without identifying the grant.

  However, to use the permissions in the grant immediately, use the `GrantToken` that `CreateGrant` returns. For details, see Using a grant token in the AWS Key Management Service Developer Guide .
- The `CreateGrant` operation also returns a `GrantId`. You can use the `GrantId` and a key identifier to identify the grant in the RetireGrant and RevokeGrant operations. To find the grant ID, use the ListGrants or ListRetirableGrants operations.

The KMS key that you use for this operation must be in a compatible key state. For details, see Key states of AWS KMS keys in the *AWS Key Management Service Developer Guide*.

Cross-account use: Yes. To perform this operation on a KMS key in a different AWS account, specify the key ARN in the value of the `KeyId` parameter.

Required permissions: kms:CreateGrant (key policy)

Related operations:

- ListGrants
- ListRetirableGrants
- RetireGrant
- RevokeGrant

Eventual consistency: The AWS KMS API follows an eventual consistency model. For more information, see AWS KMS eventual consistency.

Request Syntax

```
{
   "Constraints": {
      "EncryptionContextEquals": {
         "string" : "string"
      },
      "EncryptionContextSubset": {
         "string" : "string"
      },
      "SourceArn": "string"
   },
   "DryRun": boolean,
   "GranteePrincipal": "string",
   "GranteeServicePrincipal": "string",
   "GrantTokens": [ "string" ],
   "KeyId": "string",
   "Name": "string",
   "Operations": [ "string" ],
   "RetiringPrincipal": "string",
   "RetiringServicePrincipal": "string"
}
```

Request Parameters

For information about the parameters that are common to all actions, see Common Parameters.

The request accepts the following data in JSON format.

> **Note**
>
> In the following list, the required parameters are described first.

KeyId

Identifies the KMS key for the grant. The grant gives principals permission to use this KMS key.

Specify the key ID or key ARN of the KMS key. To specify a KMS key in a different AWS account, you must use the key ARN.

For example:

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

To get the key ID and key ARN for a KMS key, use ListKeys or DescribeKey.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 2048.

Required: Yes

Operations

A list of operations that the grant permits.

This list must include only operations that are permitted in a grant. Also, the operation must be supported on the KMS key. For example, you cannot create a grant for a symmetric encryption KMS key that allows the Sign operation, or a grant for an asymmetric KMS key that allows the GenerateDataKey operation. If you try, AWS KMS returns a `ValidationError` exception. For details, see Grant operations in the *AWS Key Management Service Developer Guide*.

Type: Array of strings

Valid Values: `Decrypt | Encrypt | GenerateDataKey | GenerateDataKeyWithoutPlaintext | ReEncryptFrom | ReEncryptTo | Sign | Verify | GetPublicKey | CreateGrant | RetireGrant | DescribeKey |`

`GenerateDataKeyPair | GenerateDataKeyPairWithoutPlaintext | GenerateMac | VerifyMac | DeriveSharedSecret | Encapsulate | Decapsulate`

Required: Yes

Constraints

Specifies a grant constraint.

> **Important**
>
> Do not include confidential or sensitive information in this field. This field may be displayed in plaintext in CloudTrail logs and other output.

AWS KMS supports the following grant constraints.

- `EncryptionContextEquals` and `EncryptionContextSubset` — These encryption context grant constraints allow the permissions in the grant only when the encryption context in the request matches (`EncryptionContextEquals`) or includes (`EncryptionContextSubset`) the encryption context specified in the constraint.

  Encryption context grant constraints are supported only on grant operations that include an `EncryptionContext` parameter, such as cryptographic operations on symmetric encryption KMS keys. You cannot use an encryption context grant constraint for cryptographic operations with asymmetric KMS keys or HMAC KMS keys. Operations with these keys don't support an encryption context. Grants with encryption context grant constraints can include the DescribeKey and RetireGrant operations, but the constraint doesn't apply to these operations. If a grant with an encryption context grant constraint includes the `CreateGrant` operation, the constraint requires that any grants created with the `CreateGrant` permission have an equally strict or stricter encryption context constraint.

  Each constraint value can include up to 8 encryption context pairs. The encryption context value in each constraint cannot exceed 384 characters. For more information about encryption context, see Encryption context in the AWS Key Management Service Developer Guide .

- `SourceArn` — This grant constraint allows the permissions in the grant only when the request is made on behalf of a specific AWS resource, identified by its Amazon Resource Name (ARN). This is effectively the same as having the aws:SourceArn global condition key in the grant. The SourceArn constraint is supported on grants for all types of KMS keys and can

also be applied to the DescribeKey operation when specified in the request. However, it does not apply to RetireGrant operation.

For information about grant constraints, see Using grant constraints in the *AWS Key Management Service Developer Guide*.

Type: GrantConstraints object

Required: No

DryRun

Checks if your request will succeed. `DryRun` is an optional parameter.

To learn more about how to use this parameter, see Testing your permissions in the *AWS Key Management Service Developer Guide*.

Type: Boolean

Required: No

GranteePrincipal

The identity that gets the permissions specified in the grant.

To specify the grantee principal, use the Amazon Resource Name (ARN) of an AWS principal. Valid principals include AWS accounts, IAM users, IAM roles, federated users, and assumed role users. For help with the ARN syntax for a principal, see IAM ARNs in the AWS Identity and Access Management User Guide .

You must specify either `GranteePrincipal` or `GranteeServicePrincipal`, but not both.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 256.

Pattern: `^[\w+=,.@:/-]+$`

Required: No

GranteeServicePrincipal

The AWS service principal that gets the permissions specified in the grant.

When you specify a `GranteeServicePrincipal`, you must also specify a `SourceArn` grant constraint. In addition, you must specify either a `RetiringPrincipal` or a `RetiringServicePrincipal`.

You must specify either `GranteePrincipal` or `GranteeServicePrincipal`, but not both.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 128.

Pattern: `^([A-Za-z0-9\-]+)\.([A-Za-z0-9\-]+)(\.[A-Za-z0-9\-]+)+$`

Required: No

GrantTokens

A list of grant tokens.

Use a grant token when your permission to call this operation comes from a new grant that has not yet achieved *eventual consistency*. For more information, see Grant token and Using a grant token in the *AWS Key Management Service Developer Guide*.

Type: Array of strings

Array Members: Minimum number of 0 items. Maximum number of 10 items.

Length Constraints: Minimum length of 1. Maximum length of 8192.

Required: No

Name

A friendly name for the grant. Use this value to prevent the unintended creation of duplicate grants when retrying this request.

> **Important**
>
> Do not include confidential or sensitive information in this field. This field may be displayed in plaintext in CloudTrail logs and other output.

When this value is absent, all `CreateGrant` requests result in a new grant with a unique `GrantId` even if all the supplied parameters are identical. This can result in unintended duplicates when you retry the `CreateGrant` request.

When this value is present, you can retry a `CreateGrant` request with identical parameters; if the grant already exists, the original `GrantId` is returned without creating a new grant. Note that the returned grant token is unique with every `CreateGrant` request, even when a duplicate `GrantId` is returned. All grant tokens for the same grant ID can be used interchangeably.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 256.

Pattern: `^[a-zA-Z0-9:/_-]+$`

Required: No

RetiringPrincipal

The principal that has permission to use the RetireGrant operation to retire the grant.

To specify the principal, use the Amazon Resource Name (ARN) of an AWS principal. Valid principals include AWS accounts, IAM users, IAM roles, federated users, and assumed role users. For help with the ARN syntax for a principal, see IAM ARNs in the AWS Identity and Access Management User Guide .

The grant determines the retiring principal. Other principals might have permission to retire the grant or revoke the grant. For details, see RevokeGrant and Retiring and revoking grants in the *AWS Key Management Service Developer Guide*.

You can specify either `RetiringPrincipal` or `RetiringServicePrincipal`, but not both.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 256.

Pattern: `^[\w+=,.@:/-]+$`

Required: No

RetiringServicePrincipal

The AWS service principal that has permission to use the RetireGrant operation to retire the grant.

You can specify either `RetiringPrincipal` or `RetiringServicePrincipal`, but not both.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 128.

Pattern: `^([A-Za-z0-9\-]+)\.([A-Za-z0-9\-]+)(\.[A-Za-z0-9\-]+)+$`

Required: No

Response Syntax

```
{
   "GrantId": "string",
   "GrantToken": "string"
}
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

GrantId

The unique identifier for the grant.

You can use the `GrantId` in a ListGrants, RetireGrant, or RevokeGrant operation.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 128.

GrantToken

The grant token.

Use a grant token when your permission to call this operation comes from a new grant that has not yet achieved *eventual consistency*. For more information, see Grant token and Using a grant token in the *AWS Key Management Service Developer Guide*.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 8192.

Errors

For information about the errors that are common to all actions, see Common Error Types.

DependencyTimeoutException

The system timed out while trying to fulfill the request. You can retry the request.

HTTP Status Code: 500

DisabledException

The request was rejected because the specified KMS key is not enabled.

HTTP Status Code: 400

DryRunOperationException

The request was rejected because the DryRun parameter was specified.

HTTP Status Code: 400

InvalidArnException

The request was rejected because a specified ARN, or an ARN in a key policy, is not valid.

HTTP Status Code: 400

InvalidGrantTokenException

The request was rejected because the specified grant token is not valid.

HTTP Status Code: 400

KMSInternalException

The request was rejected because an internal exception occurred. The request can be retried.

HTTP Status Code: 500

KMSInvalidStateException

The request was rejected because the state of the specified resource is not valid for this request.

This exceptions means one of the following:

- The key state of the KMS key is not compatible with the operation.

To find the key state, use the DescribeKey operation. For more information about which key states are compatible with each AWS KMS operation, see Key states of AWS KMS keys in the AWS Key Management Service Developer Guide .

- For cryptographic operations on KMS keys in custom key stores, this exception represents a general failure with many possible causes. To identify the cause, see the error message that accompanies the exception.

HTTP Status Code: 400

LimitExceededException

The request was rejected because a length constraint or quota was exceeded. For more information, see Quotas in the *AWS Key Management Service Developer Guide*.

HTTP Status Code: 400

NotFoundException

The request was rejected because the specified entity or resource could not be found.

HTTP Status Code: 400

Examples

The following examples are formatted for legibility.

Example Request

This example illustrates one usage of CreateGrant.

```
POST / HTTP/1.1
Host: kms.us-east-2.amazonaws.com
Content-Length: 176
X-Amz-Target: TrentService.CreateGrant
X-Amz-Date: 20161031T202851Z
Content-Type: application/x-amz-json-1.1
Authorization: AWS4-HMAC-SHA256\
 Credential=AKIAI44QH8DHBEXAMPLE/20161031/us-east-2/kms/aws4_request,\
 SignedHeaders=content-type;host;x-amz-date;x-amz-target,\
 Signature=84a2b3b8eb50b9bf34ba844cd5e59649fb315a16b447357ae49bf8b87774c8f7

{
```

```
  "Operations": [
    "Encrypt",
    "Decrypt"
  ],
  "GranteePrincipal": "arn:aws:iam::111122223333:role/ExampleRole",
  "KeyId": "arn:aws:kms:us-
east-2:444455556666:key/1234abcd-12ab-34cd-56ef-1234567890ab"
}
```

Example Response

This example illustrates one usage of CreateGrant.

```
HTTP/1.1 200 OK
Server: Server
Date: Mon, 31 Oct 2016 20:28:51 GMT
Content-Type: application/x-amz-json-1.1
Content-Length: 585
Connection: keep-alive
x-amzn-RequestId: a2d8d452identity center-9fa8-11e6-b30c-dbb8ea4d97c5

{
  "GrantId": "0c237476b39f8bc44e45212e08498fbe3151305030726c0590dd8d3e9f3d6a60",
  "GrantToken":
 "AQpAM2RhZTk1MGMyNTk2ZmZmMzEyYWVhOWViN2I1MWM4Mzc0MWFiYjc0ZDE1ODkyNGFlNTIzODZhMzgyZjBlNGY3NiKIA
ZJP7m6f1g8GzV47HX5phdtONAP7K_HQIflcgpkoCqd_fUnE114mSmiagWkbQ5sqAVV3ov-
VeqgrvMe5ZFEWLMSluvBAqdjHEdMIkHMlhlj4ENZbzBfo9Wxk8b8SnwP4kc4gGivedzFXo-
dwN8fxjjq_ZZ9JFOj2ijIbj5FyogDCN0drOfi8RORSEuCEmPvjFRMFAwcmwFkN2NPp89amA"
}
```

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface V2
- AWS SDK for .NET V4
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2

- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- AWS SDK for PHP V3
- AWS SDK for Python (Boto3)
- AWS SDK for Ruby V3

CreateKey

Creates a unique customer managed KMS key in your AWS account and Region. You can use a KMS key in cryptographic operations, such as encryption and signing. Some AWS services let you use KMS keys that you create and manage to protect your service resources.

A KMS key is a logical representation of a cryptographic key. In addition to the key material used in cryptographic operations, a KMS key includes metadata, such as the key ID, key policy, creation date, description, and key state.

Use the parameters of `CreateKey` to specify the type of KMS key, the source of its key material, its key policy, description, tags, and other properties.

> **Note**
>
> AWS KMS has replaced the term *customer master key (CMK)* with *AWS Key Management Service key* and *KMS key*. The concept has not changed. To prevent breaking changes, AWS KMS is keeping some variations of this term.

To create different types of KMS keys, use the following guidance:

Symmetric encryption KMS key

By default, `CreateKey` creates a symmetric encryption KMS key with key material that KMS generates. This is the basic and most widely used type of KMS key, and provides the best performance.

To create a symmetric encryption KMS key, you don't need to specify any parameters. The default value for `KeySpec`, `SYMMETRIC_DEFAULT`, the default value for `KeyUsage`, `ENCRYPT_DECRYPT`, and the default value for `Origin`, `AWS_KMS`, create a symmetric encryption KMS key with KMS key material.

If you need a key for basic encryption and decryption or you are creating a KMS key to protect your resources in an AWS service, create a symmetric encryption KMS key. The key material in a symmetric encryption key never leaves AWS KMS unencrypted. You can use a symmetric encryption KMS key to encrypt and decrypt data up to 4,096 bytes, but they are typically used to generate data keys and data keys pairs. For details, see GenerateDataKey and GenerateDataKeyPair.

Asymmetric KMS keys

To create an asymmetric KMS key, use the `KeySpec` parameter to specify the type of key material in the KMS key. Then, use the `KeyUsage` parameter to determine whether the KMS key will be used to encrypt and decrypt or sign and verify. You can't change these properties after the KMS key is created.

Asymmetric KMS keys contain an RSA key pair, Elliptic Curve (ECC) key pair, ML-DSA key pair or an SM2 key pair (China Regions only). The private key in an asymmetric KMS key never leaves AWS KMS unencrypted. However, you can use the GetPublicKey operation to download the public key so it can be used outside of AWS KMS. Each KMS key can have only one key usage. KMS keys with RSA key pairs can be used to encrypt and decrypt data or sign and verify messages (but not both). KMS keys with NIST-standard ECC key pairs can be used to sign and verify messages or derive shared secrets (but not both). KMS keys with `ECC_SECG_P256K1` can be used only to sign and verify messages. KMS keys with ML-DSA key pairs can be used to sign and verify messages. KMS keys with SM2 key pairs (China Regions only) can be used to either encrypt and decrypt data, sign and verify messages, or derive shared secrets (you must choose one key usage type). For information about asymmetric KMS keys, see Asymmetric KMS keys in the *AWS Key Management Service Developer Guide*.

HMAC KMS key

To create an HMAC KMS key, set the `KeySpec` parameter to a key spec value for HMAC KMS keys. Then set the `KeyUsage` parameter to `GENERATE_VERIFY_MAC`. You must set the key usage even though `GENERATE_VERIFY_MAC` is the only valid key usage value for HMAC KMS keys. You can't change these properties after the KMS key is created.

HMAC KMS keys are symmetric keys that never leave AWS KMS unencrypted. You can use HMAC keys to generate (GenerateMac) and verify (VerifyMac) HMAC codes for messages up to 4096 bytes.

Multi-Region primary keys

To create a multi-Region *primary key* in the local AWS Region, use the `MultiRegion` parameter with a value of `True`. To create a multi-Region *replica key*, that is, a KMS key with the same key ID and key material as a primary key, but in a different AWS Region, use the ReplicateKey operation. To change a replica key to a primary key, and its primary key to a replica key, use the UpdatePrimaryRegion operation.

You can create multi-Region KMS keys for all supported KMS key types: symmetric encryption KMS keys, HMAC KMS keys, asymmetric encryption KMS keys, and asymmetric signing KMS keys. You can also create multi-Region keys with imported key material. However, you can't create multi-Region keys in a custom key store.

This operation supports *multi-Region keys*, an AWS KMS feature that lets you create multiple interoperable KMS keys in different AWS Regions. Because these KMS keys have the same key ID, key material, and other metadata, you can use them interchangeably to encrypt data in one AWS Region and decrypt it in a different AWS Region without re-encrypting the data or making a cross-Region call. For more information about multi-Region keys, see Multi-Region keys in AWS KMS in the *AWS Key Management Service Developer Guide*.

Imported key material

To import your own key material into a KMS key, begin by creating a KMS key with no key material. To do this, use the `Origin` parameter of `CreateKey` with a value of `EXTERNAL`. Next, use GetParametersForImport operation to get a public key and import token. Use the wrapping public key to encrypt your key material. Then, use ImportKeyMaterial with your import token to import the key material. For step-by-step instructions, see Importing Key Material in the AWS Key Management Service Developer Guide .

You can import key material into KMS keys of all supported KMS key types: symmetric encryption KMS keys, HMAC KMS keys, asymmetric encryption KMS keys, and asymmetric signing KMS keys. You can also create multi-Region keys with imported key material. However, you can't import key material into a KMS key in a custom key store.

To create a multi-Region primary key with imported key material, use the `Origin` parameter of `CreateKey` with a value of `EXTERNAL` and the `MultiRegion` parameter with a value of `True`. To create replicas of the multi-Region primary key, use the ReplicateKey operation. For instructions, see Importing key material step 1. For more information about multi-Region keys, see Multi-Region keys in AWS KMS in the *AWS Key Management Service Developer Guide*.

Custom key store

A custom key store lets you protect your AWS resources using keys in a backing key store that you own and manage. When you request a cryptographic operation with a KMS key in a custom key store, the operation is performed in the backing key store using its cryptographic keys.

AWS KMS supports AWS CloudHSM key stores backed by an AWS CloudHSM cluster and external key stores backed by an external key manager outside of AWS. When you create a

KMS key in an AWS CloudHSM key store, AWS KMS generates an encryption key in the AWS CloudHSM cluster and associates it with the KMS key. When you create a KMS key in an external key store, you specify an existing encryption key in the external key manager.

> **Note**
>
> Some external key managers provide a simpler method for creating a KMS key in an external key store. For details, see your external key manager documentation.

Before you create a KMS key in a custom key store, the `ConnectionState` of the key store must be `CONNECTED`. To connect the custom key store, use the ConnectCustomKeyStore operation. To find the `ConnectionState`, use the DescribeCustomKeyStores operation.

To create a KMS key in a custom key store, use the `CustomKeyStoreId`. Use the default `KeySpec` value, `SYMMETRIC_DEFAULT`, and the default `KeyUsage` value, `ENCRYPT_DECRYPT` to create a symmetric encryption key. No other key type is supported in a custom key store.

To create a KMS key in an AWS CloudHSM key store, use the `Origin` parameter with a value of `AWS_CLOUDHSM`. The AWS CloudHSM cluster that is associated with the custom key store must have at least two active HSMs in different Availability Zones in the AWS Region.

To create a KMS key in an external key store, use the `Origin` parameter with a value of `EXTERNAL_KEY_STORE` and an `XksKeyId` parameter that identifies an existing external key.

> **Note**
>
> Some external key managers provide a simpler method for creating a KMS key in an external key store. For details, see your external key manager documentation.

Cross-account use: No. You cannot use this operation to create a KMS key in a different AWS account.

Required permissions: kms:CreateKey (IAM policy). To use the `Tags` parameter, kms:TagResource (IAM policy). For examples and information about related permissions, see Allow a user to create KMS keys in the *AWS Key Management Service Developer Guide*.

Related operations:

- DescribeKey
- ListKeys
- ScheduleKeyDeletion

Eventual consistency: The AWS KMS API follows an eventual consistency model. For more information, see AWS KMS eventual consistency.

Request Syntax

```
{
   "BypassPolicyLockoutSafetyCheck": boolean,
   "CustomerMasterKeySpec": "string",
   "CustomKeyStoreId": "string",
   "Description": "string",
   "KeySpec": "string",
   "KeyUsage": "string",
   "MultiRegion": boolean,
   "Origin": "string",
   "Policy": "string",
   "Tags": [
      {
         "TagKey": "string",
         "TagValue": "string"
      }
   ],
   "XksKeyId": "string"
}
```

Request Parameters

For information about the parameters that are common to all actions, see Common Parameters.

The request accepts the following data in JSON format.

> **Note**
>
> In the following list, the required parameters are described first.

BypassPolicyLockoutSafetyCheck

Skips ("bypasses") the key policy lockout safety check. The default value is false.

> **Important**
>
> Setting this value to true increases the risk that the KMS key becomes unmanageable. Do not set this value to true indiscriminately.
> For more information, see Default key policy in the *AWS Key Management Service Developer Guide*.

Use this parameter only when you intend to prevent the principal that is making the request from making a subsequent PutKeyPolicy request on the KMS key.

Type: Boolean

Required: No

CustomerMasterKeySpec

This parameter has been deprecated.

Instead, use the `KeySpec` parameter.

The `KeySpec` and `CustomerMasterKeySpec` parameters work the same way. Only the names differ. We recommend that you use `KeySpec` parameter in your code. However, to avoid breaking changes, AWS KMS supports both parameters.

Type: String

Valid Values: `RSA_2048 | RSA_3072 | RSA_4096 | ECC_NIST_P256 | ECC_NIST_P384 | ECC_NIST_P521 | ECC_SECG_P256K1 | SYMMETRIC_DEFAULT | HMAC_224 | HMAC_256 | HMAC_384 | HMAC_512 | SM2`

Required: No

CustomKeyStoreId

Creates the KMS key in the specified custom key store. The `ConnectionState` of the custom key store must be `CONNECTED`. To find the CustomKeyStoreID and ConnectionState use the DescribeCustomKeyStores operation.

This parameter is valid only for symmetric encryption KMS keys in a single Region. You cannot create any other type of KMS key in a custom key store.

When you create a KMS key in an AWS CloudHSM key store, AWS KMS generates a non-exportable 256-bit symmetric key in its associated AWS CloudHSM cluster and associates it with the KMS key. When you create a KMS key in an external key store, you must use the `XksKeyId` parameter to specify an external key that serves as key material for the KMS key.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 64.

Required: No

Description

A description of the KMS key. Use a description that helps you decide whether the KMS key is appropriate for a task. The default value is an empty string (no description).

> **Important**
>
> Do not include confidential or sensitive information in this field. This field may be displayed in plaintext in CloudTrail logs and other output.

To set or change the description after the key is created, use UpdateKeyDescription.

Type: String

Length Constraints: Minimum length of 0. Maximum length of 8192.

Required: No

KeySpec

Specifies the type of KMS key to create. The default value, `SYMMETRIC_DEFAULT`, creates a KMS key with a 256-bit AES-GCM key that is used for encryption and decryption, except in China Regions, where it creates a 128-bit symmetric key that uses SM4 encryption. For a detailed description of all supported key specs, see Key spec reference in the AWS Key Management Service Developer Guide .

The `KeySpec` determines whether the KMS key contains a symmetric key or an asymmetric key pair. It also determines the algorithms that the KMS key supports. You can't change

the KeySpec after the KMS key is created. To further restrict the algorithms that can be used with the KMS key, use a condition key in its key policy or IAM policy. For more information, see kms:EncryptionAlgorithm, kms:MacAlgorithm, kms:KeyAgreementAlgorithm, or kms:SigningAlgorithm in the AWS Key Management Service Developer Guide .

> **Important**
>
> AWS services that are integrated with AWS KMS use symmetric encryption KMS keys to protect your data. These services do not support asymmetric KMS keys or HMAC KMS keys.

AWS KMS supports the following key specs for KMS keys:

- Symmetric encryption key (default)
  - SYMMETRIC_DEFAULT
- HMAC keys (symmetric)
  - HMAC_224
  - HMAC_256
  - HMAC_384
  - HMAC_512
- Asymmetric RSA key pairs (encryption and decryption -or- signing and verification)
  - RSA_2048
  - RSA_3072
  - RSA_4096
- Asymmetric NIST-standard elliptic curve key pairs (signing and verification -or- deriving shared secrets)
  - ECC_NIST_P256 (secp256r1)
  - ECC_NIST_P384 (secp384r1)
  - ECC_NIST_P521 (secp521r1)
  - ECC_NIST_EDWARDS25519 (ed25519) - signing and verification only
    - **Note:** For ECC_NIST_EDWARDS25519 KMS keys, the ED25519_SHA_512 signing algorithm requires MessageType:RAW, while ED25519_PH_SHA_512 requires MessageType:DIGEST. These message types cannot be used interchangeably.

- Other asymmetric elliptic curve key pairs (signing and verification)
  - ECC_SECG_P256K1 (secp256k1), commonly used for cryptocurrencies.
- Asymmetric ML-DSA key pairs (signing and verification)
  - ML_DSA_44
  - ML_DSA_65
  - ML_DSA_87
- Asymmetric ML-KEM key pairs (establishing shared secrets)
  - ML_KEM_512
  - ML_KEM_768
  - ML_KEM_1024

  The ML-KEM key specs are not available in China Regions.
- SM2 key pairs (encryption and decryption -or- signing and verification -or- deriving shared secrets)
  - SM2 (China Regions only)

Type: String

Valid Values: RSA_2048 | RSA_3072 | RSA_4096 | ECC_NIST_P256 | ECC_NIST_P384 | ECC_NIST_P521 | ECC_SECG_P256K1 | SYMMETRIC_DEFAULT | HMAC_224 | HMAC_256 | HMAC_384 | HMAC_512 | SM2 | ML_DSA_44 | ML_DSA_65 | ML_DSA_87 | ECC_NIST_EDWARDS25519 | ML_KEM_512 | ML_KEM_768 | ML_KEM_1024

Required: No

KeyUsage

Determines the cryptographic operations for which you can use the KMS key. The default value is ENCRYPT_DECRYPT. This parameter is optional when you are creating a symmetric encryption KMS key; otherwise, it is required. You can't change the KeyUsage value after the KMS key is created. Each AWS KMS key can have only one key usage. This follows key usage best practices according to NIST SP 800-57 Recommendations for Key Management, section 5.2, Key usage.

Select only one valid value.

- For symmetric encryption KMS keys, omit the parameter or specify ENCRYPT_DECRYPT.
- For HMAC KMS keys (symmetric), specify GENERATE_VERIFY_MAC.

- For asymmetric KMS keys with RSA key pairs, specify `ENCRYPT_DECRYPT` or `SIGN_VERIFY`.
- For asymmetric KMS keys with NIST-standard elliptic curve key pairs, specify `SIGN_VERIFY` or `KEY_AGREEMENT`.
- For asymmetric KMS keys with `ECC_SECG_P256K1` key pairs, specify `SIGN_VERIFY`.
- For asymmetric KMS keys with ML-DSA key pairs, specify `SIGN_VERIFY`.
- For asymmetric KMS keys with ML-KEM key pairs, specify `ENCAPSULATE_DECAPSULATE`.
- For asymmetric KMS keys with SM2 key pairs (China Regions only), specify `ENCRYPT_DECRYPT`, `SIGN_VERIFY`, or `KEY_AGREEMENT`.

Type: String

Valid Values: `SIGN_VERIFY | ENCRYPT_DECRYPT | GENERATE_VERIFY_MAC | KEY_AGREEMENT | ENCAPSULATE_DECAPSULATE`

Required: No

MultiRegion

Creates a multi-Region primary key that you can replicate into other AWS Regions. You cannot change this value after you create the KMS key.

For a multi-Region key, set this parameter to `True`. For a single-Region KMS key, omit this parameter or set it to `False`. The default value is `False`.

This operation supports *multi-Region keys*, an AWS KMS feature that lets you create multiple interoperable KMS keys in different AWS Regions. Because these KMS keys have the same key ID, key material, and other metadata, you can use them interchangeably to encrypt data in one AWS Region and decrypt it in a different AWS Region without re-encrypting the data or making a cross-Region call. For more information about multi-Region keys, see Multi-Region keys in AWS KMS in the *AWS Key Management Service Developer Guide*.

This value creates a *primary key*, not a replica. To create a *replica key*, use the ReplicateKey operation.

You can create a symmetric or asymmetric multi-Region key, and you can create a multi-Region key with imported key material. However, you cannot create a multi-Region key in a custom key store.

Type: Boolean

Required: No

Origin

The source of the key material for the KMS key. You cannot change the origin after you create the KMS key. The default is `AWS_KMS`, which means that AWS KMS creates the key material.

To create a KMS key with no key material (for imported key material), set this value to `EXTERNAL`. For more information about importing key material into AWS KMS, see Importing Key Material in the *AWS Key Management Service Developer Guide*. The `EXTERNAL` origin value is valid only for symmetric KMS keys.

To create a KMS key in an AWS CloudHSM key store and create its key material in the associated AWS CloudHSM cluster, set this value to `AWS_CLOUDHSM`. You must also use the `CustomKeyStoreId` parameter to identify the AWS CloudHSM key store. The `KeySpec` value must be `SYMMETRIC_DEFAULT`.

To create a KMS key in an external key store, set this value to `EXTERNAL_KEY_STORE`. You must also use the `CustomKeyStoreId` parameter to identify the external key store and the `XksKeyId` parameter to identify the associated external key. The `KeySpec` value must be `SYMMETRIC_DEFAULT`.

Type: String

Valid Values: `AWS_KMS | EXTERNAL | AWS_CLOUDHSM | EXTERNAL_KEY_STORE`

Required: No

Policy

The key policy to attach to the KMS key.

If you provide a key policy, it must meet the following criteria:

- The key policy must allow the calling principal to make a subsequent `PutKeyPolicy` request on the KMS key. This reduces the risk that the KMS key becomes unmanageable. For more information, see Default key policy in the *AWS Key Management Service Developer Guide*. (To omit this condition, set `BypassPolicyLockoutSafetyCheck` to true.)
- Each statement in the key policy must contain one or more principals. The principals in the key policy must exist and be visible to AWS KMS. When you create a new AWS principal, you might need to enforce a delay before including the new principal in a key policy because the new principal might not be immediately visible to AWS KMS. For more information, see Changes that I make are not always immediately visible in the *AWS Identity and Access Management User Guide*.

> **Note**
>
> If either of the required `Resource` or `Action` elements are missing from a key policy statement, the policy statement has no effect. When a key policy statement is missing one of these elements, the AWS KMS console correctly reports an error, but the `CreateKey` and `PutKeyPolicy` API requests succeed, even though the policy statement is ineffective.
> For more information on required key policy elements, see Elements in a key policy in the *AWS Key Management Service Developer Guide*.

If you do not provide a key policy, AWS KMS attaches a default key policy to the KMS key. For more information, see Default key policy in the *AWS Key Management Service Developer Guide*.

> **Note**
>
> If the key policy exceeds the length constraint, AWS KMS returns a `LimitExceededException`.

For help writing and formatting a JSON policy document, see the IAM JSON Policy Reference in the AWS Identity and Access Management User Guide .

Type: String

Length Constraints: Minimum length of 1. Maximum length of 32768.

Pattern: `[\u0009\u000A\u000D\u0020-\u00FF]+`

Required: No

Tags

Assigns one or more tags to the KMS key. Use this parameter to tag the KMS key when it is created. To tag an existing KMS key, use the TagResource operation.

> **Important**
>
> Do not include confidential or sensitive information in this field. This field may be displayed in plaintext in CloudTrail logs and other output.

> **Note**
>
> Tagging or untagging a KMS key can allow or deny permission to the KMS key. For details, see ABAC for AWS KMS in the *AWS Key Management Service Developer Guide*.

To use this parameter, you must have kms:TagResource permission in an IAM policy.

Each tag consists of a tag key and a tag value. Both the tag key and the tag value are required, but the tag value can be an empty (null) string. You cannot have more than one tag on a KMS key with the same tag key. If you specify an existing tag key with a different tag value, AWS KMS replaces the current tag value with the specified one.

When you add tags to an AWS resource, AWS generates a cost allocation report with usage and costs aggregated by tags. Tags can also be used to control access to a KMS key. For details, see Tags in AWS KMS.

Type: Array of Tag objects

Required: No

XksKeyId

Identifies the external key that serves as key material for the KMS key in an external key store. Specify the ID that the external key store proxy uses to refer to the external key. For help, see the documentation for your external key store proxy.

This parameter is required for a KMS key with an `Origin` value of `EXTERNAL_KEY_STORE`. It is not valid for KMS keys with any other `Origin` value.

The external key must be an existing 256-bit AES symmetric encryption key hosted outside of AWS in an external key manager associated with the external key store specified by the `CustomKeyStoreId` parameter. This key must be enabled and configured to perform encryption and decryption. Each KMS key in an external key store must use a different external key. For details, see Requirements for a KMS key in an external key store in the *AWS Key Management Service Developer Guide*.

Each KMS key in an external key store is associated two backing keys. One is key material that AWS KMS generates. The other is the external key specified by this parameter. When you use the KMS key in an external key store to encrypt data, the encryption operation is performed first by AWS KMS using the AWS KMS key material, and then by the external key manager

using the specified external key, a process known as *double encryption*. For details, see Double encryption in the *AWS Key Management Service Developer Guide*.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 128.

Pattern: `^[a-zA-Z0-9-_.]+$`

Required: No

Response Syntax

```
{
   "KeyMetadata": {
      "Arn": "string",
      "AWSAccountId": "string",
      "CloudHsmClusterId": "string",
      "CreationDate": number,
      "CurrentKeyMaterialId": "string",
      "CustomerMasterKeySpec": "string",
      "CustomKeyStoreId": "string",
      "DeletionDate": number,
      "Description": "string",
      "Enabled": boolean,
      "EncapsulationAlgorithms": [ "string" ],
      "EncryptionAlgorithms": [ "string" ],
      "ExpirationModel": "string",
      "KeyAgreementAlgorithms": [ "string" ],
      "KeyId": "string",
      "KeyManager": "string",
      "KeySpec": "string",
      "KeyState": "string",
      "KeyUsage": "string",
      "MacAlgorithms": [ "string" ],
      "MultiRegion": boolean,
      "MultiRegionConfiguration": {
         "MultiRegionKeyType": "string",
         "PrimaryKey": {
            "Arn": "string",
            "Region": "string"
         },
         "ReplicaKeys": [
```

```
            {
               "Arn": "string",
               "Region": "string"
            }
         ]
      },
      "Origin": "string",
      "PendingDeletionWindowInDays": number,
      "SigningAlgorithms": [ "string" ],
      "ValidTo": number,
      "XksKeyConfiguration": {
         "Id": "string"
      }
   }
}
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

KeyMetadata

Metadata associated with the KMS key.

Type: KeyMetadata object

Errors

For information about the errors that are common to all actions, see Common Error Types.

CloudHsmClusterInvalidConfigurationException

The request was rejected because the associated AWS CloudHSM cluster did not meet the configuration requirements for an AWS CloudHSM key store.

- The AWS CloudHSM cluster must be configured with private subnets in at least two different Availability Zones in the Region.
- The security group for the cluster (cloudhsm-cluster-*<cluster-id>*-sg) must include inbound rules and outbound rules that allow TCP traffic on ports 2223-2225. The **Source** in the inbound rules and the **Destination** in the outbound rules must match the security group

ID. These rules are set by default when you create the AWS CloudHSM cluster. Do not delete or change them. To get information about a particular security group, use the DescribeSecurityGroups operation.

- The AWS CloudHSM cluster must contain at least as many HSMs as the operation requires. To add HSMs, use the AWS CloudHSM CreateHsm operation.

  For the CreateCustomKeyStore, UpdateCustomKeyStore, and CreateKey operations, the AWS CloudHSM cluster must have at least two active HSMs, each in a different Availability Zone. For the ConnectCustomKeyStore operation, the AWS CloudHSM must contain at least one active HSM.

For information about the requirements for an AWS CloudHSM cluster that is associated with an AWS CloudHSM key store, see Assemble the Prerequisites in the *AWS Key Management Service Developer Guide*. For information about creating a private subnet for an AWS CloudHSM cluster, see Create a Private Subnet in the *AWS CloudHSM User Guide*. For information about cluster security groups, see Configure a Default Security Group in the AWS CloudHSM User Guide .

HTTP Status Code: 400

CustomKeyStoreInvalidStateException

The request was rejected because of the `ConnectionState` of the custom key store. To get the `ConnectionState` of a custom key store, use the DescribeCustomKeyStores operation.

This exception is thrown under the following conditions:

- You requested the ConnectCustomKeyStore operation on a custom key store with a `ConnectionState` of `DISCONNECTING` or `FAILED`. This operation is valid for all other `ConnectionState` values. To reconnect a custom key store in a `FAILED` state, disconnect it (DisconnectCustomKeyStore), then connect it (`ConnectCustomKeyStore`).
- You requested the CreateKey operation in a custom key store that is not connected. This operations is valid only when the custom key store `ConnectionState` is `CONNECTED`.
- You requested the DisconnectCustomKeyStore operation on a custom key store with a `ConnectionState` of `DISCONNECTING` or `DISCONNECTED`. This operation is valid for all other `ConnectionState` values.
- You requested the UpdateCustomKeyStore or DeleteCustomKeyStore operation on a custom key store that is not disconnected. `UpdateCustomKeyStore` can be called on a custom key store in the `CONNECTED` state only to update `NewCustomKeyStoreName`. For all other properties, the custom key store `ConnectionState` must be `DISCONNECTED`.

- You requested the GenerateRandom operation in an AWS CloudHSM key store that is not connected. This operation is valid only when the AWS CloudHSM key store `ConnectionState` is `CONNECTED`.

HTTP Status Code: 400

CustomKeyStoreNotFoundException

The request was rejected because AWS KMS cannot find a custom key store with the specified key store name or ID.

HTTP Status Code: 400

DependencyTimeoutException

The system timed out while trying to fulfill the request. You can retry the request.

HTTP Status Code: 500

InvalidArnException

The request was rejected because a specified ARN, or an ARN in a key policy, is not valid.

HTTP Status Code: 400

KMSInternalException

The request was rejected because an internal exception occurred. The request can be retried.

HTTP Status Code: 500

LimitExceededException

The request was rejected because a length constraint or quota was exceeded. For more information, see Quotas in the *AWS Key Management Service Developer Guide*.

HTTP Status Code: 400

MalformedPolicyDocumentException

The request was rejected because the specified policy is not syntactically or semantically correct.

HTTP Status Code: 400

TagException

The request was rejected because one or more tags are not valid.

HTTP Status Code: 400

UnsupportedOperationException

The request was rejected because a specified parameter is not supported or a specified resource is not valid for this operation.

HTTP Status Code: 400

XksKeyAlreadyInUseException

The request was rejected because the (`XksKeyId`) is already associated with another KMS key in this external key store. Each KMS key in an external key store must be associated with a different external key.

HTTP Status Code: 400

XksKeyInvalidConfigurationException

The request was rejected because the external key specified by the `XksKeyId` parameter did not meet the configuration requirements for an external key store.

The external key must be an AES-256 symmetric key that is enabled and performs encryption and decryption.

HTTP Status Code: 400

XksKeyNotFoundException

The request was rejected because the external key store proxy could not find the external key. This exception is thrown when the value of the `XksKeyId` parameter doesn't identify a key in the external key manager associated with the external key proxy.

Verify that the `XksKeyId` represents an existing key in the external key manager. Use the key identifier that the external key store proxy uses to identify the key. For details, see the documentation provided with your external key store proxy or key manager.

HTTP Status Code: 400

Examples

The following examples are formatted for legibility.

Example Request

This example illustrates one usage of CreateKey.

```
POST / HTTP/1.1
Host: kms.us-east-2.amazonaws.com
Content-Type: application/x-amz-json-1.1
Authorization: AWS4-HMAC-SHA256\
 Credential=AKIAI44QH8DHBEXAMPLE/20170705/us-east-2/kms/aws4_request,\
 SignedHeaders=content-type;host;x-amz-date;x-amz-target,\
 Signature=8fb59aa17854a97df47aae69f560b66178ed0b5e1ebe334be516c4f3f59acedc
X-Amz-Target: TrentService.CreateKey
X-Amz-Date: 20170705T210455Z
Content-Length: 62

{
  "Tags": [{
    "TagValue": "ExampleUser",
    "TagKey": "CreatedBy"
  }]
}
```

Example Response

This example illustrates one usage of CreateKey.

```
HTTP/1.1 200 OK
Server: Server
Date: Wed, 05 Jul 2017 21:04:55 GMT
Content-Type: application/x-amz-json-1.1
Content-Length: 335
Connection: keep-alive
x-amzn-RequestId: 98b2de61-61c5-11e7-bd87-9fc4a74e147b

{
  "KeyMetadata": {
    "AWSAccountId": "111122223333",
    "Arn": "arn:aws:kms:us-
east-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
    "CreationDate": "2025-04-16T16:03:04.060000-07:00",
    "CustomerMasterKeySpec": "SYMMETRIC_DEFAULT",
    "Description": "",
    "Enabled": true,
```

```
    "KeyId": "1234abcd-12ab-34cd-56ef-1234567890ab",
    "KeyManager": "CUSTOMER",
    "KeySpec": "SYMMETRIC_DEFAULT",
    "KeyState": "Enabled",
    "KeyUsage": "ENCRYPT_DECRYPT",
    "MultiRegion": false,
    "Origin": "AWS_KMS",
    "EncryptionAlgorithms": [
      "SYMMETRIC_DEFAULT"
    ],
    "CurrentKeyMaterialId":
 "0b7fd7ddbac6eef27907413567cad8c810e2883dc8a7534067a82ee1142fc1e6"
  }
}
```

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface V2
- AWS SDK for .NET V4
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- AWS SDK for PHP V3
- AWS SDK for Python (Boto3)
- AWS SDK for Ruby V3

Decapsulate

Recovers the shared secret that was encapsulated with the public key of an ML-KEM KMS key. Module-Lattice-Based Key-Encapsulation Mechanism (ML-KEM), specified in FIPS 203 on the NIST website, is a post-quantum algorithm for establishing a shared secret between two parties. For an example of how two parties establish a shared secret over an insecure channel using ML-KEM, see ML-KEM keys in AWS KMS in the *AWS Key Management Service Developer Guide*.

The Decapsulate operation is not available in China Regions.

`Decapsulate` returns the shared secret as a 32-byte value. You can use it directly as key material, or run it through a key derivation function to derive multiple keys and nonces.

Important

`Decapsulate` returns an `InvalidCiphertextException` when the length of the `SharedSecretCiphertextBlob` is inconsistent with the ML-KEM key spec. In all other cases, `Decapsulate` returns a 32-byte pseudo-random value. This is true even when the `SharedSecretCiphertextBlob` was created using a different ML-KEM key or altered from its original value. This behavior is a security feature called *implicit rejection* that helps protect against chosen-ciphertext attacks.
Because a response does not indicate that the ciphertext and key are valid, confirm the two parties derived the same shared secret before using it.

The KMS key that you use for this operation must be in a compatible key state. For details, see Key states of AWS KMS keys in the *AWS Key Management Service Developer Guide*.

Cross-account use: Yes. To perform this operation with a KMS key in a different AWS account, specify the key ARN or alias ARN in the value of the `KeyId` parameter.

Required permissions: kms:Decapsulate (key policy)

Related operations:

- Encapsulate
- CreateKey
- GetPublicKey
- DescribeKey

Eventual consistency: The AWS KMS API follows an eventual consistency model. For more information, see AWS KMS eventual consistency.

Request Syntax

```
{
   "DryRun": boolean,
   "EncapsulationAlgorithm": "string",
   "GrantTokens": [ "string" ],
   "KeyId": "string",
   "SharedSecretCiphertextBlob": blob
}
```

Request Parameters

For information about the parameters that are common to all actions, see Common Parameters.

The request accepts the following data in JSON format.

> **Note**
>
> In the following list, the required parameters are described first.

EncapsulationAlgorithm

Specifies the key encapsulation algorithm used to recover the shared secret. The only valid value is `ML_KEM`.

Type: String

Valid Values: `ML_KEM`

Required: Yes

KeyId

Identifies the asymmetric ML-KEM KMS key whose public key was used to produce the encapsulated shared secret. AWS KMS uses the private key in the specified key pair to decapsulate the shared secret.

To specify a KMS key, use its key ID, key ARN, alias name, or alias ARN. When using an alias name, prefix it with `"alias/"`. To specify a KMS key in a different AWS account, you must use the key ARN or alias ARN.

For example:

- Key ID: `1234abcd-12ab-34cd-56ef-1234567890ab`
- Key ARN: `arn:aws:kms:us-east-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab`
- Alias name: `alias/ExampleAlias`
- Alias ARN: `arn:aws:kms:us-east-2:111122223333:alias/ExampleAlias`

To get the key ID and key ARN for a KMS key, use ListKeys or DescribeKey. To get the alias name and alias ARN, use ListAliases.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 2048.

Required: Yes

SharedSecretCiphertextBlob

The encapsulated (ciphertext) form of the shared secret to recover. This is the `SharedSecretCiphertextBlob` value that the Encapsulate operation returns. When you use the HTTP API or the AWS CLI, the value is Base64-encoded. Otherwise, it is not Base64-encoded.

Type: Base64-encoded binary data object

Length Constraints: Minimum length of 1. Maximum length of 6144.

Required: Yes

DryRun

Checks if your request will succeed. `DryRun` is an optional parameter.

To learn more about how to use this parameter, see Testing your permissions in the *AWS Key Management Service Developer Guide*.

Type: Boolean

Required: No

GrantTokens

A list of grant tokens.

Use a grant token when your permission to call this operation comes from a new grant that has not yet achieved *eventual consistency*. For more information, see Grant token and Using a grant token in the *AWS Key Management Service Developer Guide*.

Type: Array of strings

Array Members: Minimum number of 0 items. Maximum number of 10 items.

Length Constraints: Minimum length of 1. Maximum length of 8192.

Required: No

Response Syntax

```
{
   "EncapsulationAlgorithm": "string",
   "KeyId": "string",
   "SharedSecret": blob
}
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

EncapsulationAlgorithm

The key encapsulation algorithm used to recover the shared secret.

Type: String

Valid Values: `ML_KEM`

KeyId

The identifier of the ML-KEM KMS key used to decapsulate the shared secret.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 2048.

SharedSecret

The plaintext shared secret recovered from the encapsulated form. It is a 32-byte value that matches the shared secret held by the encapsulating party. When you use the HTTP API or the AWS CLI, the value is Base64-encoded. Otherwise, it is not Base64-encoded.

Type: Base64-encoded binary data object

Length Constraints: Minimum length of 1. Maximum length of 4096.

Errors

For information about the errors that are common to all actions, see Common Error Types.

DependencyTimeoutException

The system timed out while trying to fulfill the request. You can retry the request.

HTTP Status Code: 500

DisabledException

The request was rejected because the specified KMS key is not enabled.

HTTP Status Code: 400

DryRunOperationException

The request was rejected because the DryRun parameter was specified.

HTTP Status Code: 400

InvalidCiphertextException

From the Decrypt or ReEncrypt operation, the request was rejected because the specified ciphertext, or additional authenticated data incorporated into the ciphertext, such as the encryption context, is corrupted, missing, or otherwise invalid.

From the ImportKeyMaterial operation, the request was rejected because AWS KMS could not decrypt the encrypted (wrapped) key material.

From the Decapsulate operation, the request was rejected because the length of the `SharedSecretCiphertextBlob` is inconsistent with the specified ML-KEM key spec.

HTTP Status Code: 400

InvalidGrantTokenException

The request was rejected because the specified grant token is not valid.

HTTP Status Code: 400

InvalidKeyUsageException

The request was rejected for one of the following reasons:

- The `KeyUsage` value of the KMS key is incompatible with the API operation.
- The encryption algorithm or signing algorithm specified for the operation is incompatible with the type of key material in the KMS key (`KeySpec`).

For encrypting, decrypting, re-encrypting, and generating data keys, the `KeyUsage` must be `ENCRYPT_DECRYPT`. For signing and verifying messages, the `KeyUsage` must be `SIGN_VERIFY`. For generating and verifying message authentication codes (MACs), the `KeyUsage` must be `GENERATE_VERIFY_MAC`. For deriving key agreement secrets, the `KeyUsage` must be `KEY_AGREEMENT`. For encapsulating and decapsulating shared secrets, the `KeyUsage` must be `ENCAPSULATE_DECAPSULATE`. To find the `KeyUsage` of a KMS key, use the DescribeKey operation.

To find the encryption or signing algorithms supported for a particular KMS key, use the DescribeKey operation.

HTTP Status Code: 400

KeyUnavailableException

The request was rejected because the specified KMS key was not available. You can retry the request.

HTTP Status Code: 500

KMSInternalException

The request was rejected because an internal exception occurred. The request can be retried.

HTTP Status Code: 500

KMSInvalidStateException

The request was rejected because the state of the specified resource is not valid for this request.

This exceptions means one of the following:

- The key state of the KMS key is not compatible with the operation.

  To find the key state, use the DescribeKey operation. For more information about which key states are compatible with each AWS KMS operation, see Key states of AWS KMS keys in the AWS Key Management Service Developer Guide .
- For cryptographic operations on KMS keys in custom key stores, this exception represents a general failure with many possible causes. To identify the cause, see the error message that accompanies the exception.

HTTP Status Code: 400

NotFoundException

The request was rejected because the specified entity or resource could not be found.

HTTP Status Code: 400

Examples

Example Request

The following example is formatted for legibility.

```
POST / HTTP/1.1
Host: kms.us-east-2.amazonaws.com
Content-Length: 1571
X-Amz-Target: TrentService.Decapsulate
X-Amz-Date: 20261018T233819Z
Content-Type: application/x-amz-json-1.1
Authorization: AWS4-HMAC-SHA256\
 Credential=AKIAI44QH8DHBEXAMPLE/20261018/us-east-2/kms/aws4_request,\
 SignedHeaders=content-type;host;x-amz-date;x-amz-target,\
 Signature=2cea34fe55d5858295a377448a1e053d0edd45ce571da7cf69b202905759f272

{
    "KeyId": "1234abcd-12ab-34cd-56ef-1234567890ab",
    "SharedSecretCiphertextBlob": "kGuVROOrf4ZLP3rR7DB64O803VeoevUmGCDvJ/
vc2vBEYkYwZoc2NmNYeX0g55i/
Pz4uvqnwhVFC2KDZJD5Rdob4GoaRphStzM87F30tq7Xui3zfHO8X5aZCXJpPB3lh/DTiRoitmjo
+mwleYyL6IOG8PurEPASSROB3SdfhmVPsTY6ZaUARqtY6BFBIyHbnWzI0a2HLdl1DGnMuvy3yTdhXzE5/
```

```
Z7ZUAe5edL+WLao7vWWl2InU1TDlPE+kDu540aCoIX37ke4dmTJIe/
+e20nzINEmuhCzUVIoRGrYyT8ZnuyMq9kZtd
+SmhfKRA5PQT0vrp0PSHE2653724YPKvLSuEoNbMSXPTbB8rBeP3kXg0p+4ZLjRda3SoMEvqbQdWLVRclMI
+PtXUSCJkKWrbmg9/2eyO9fPNCsck36zVEpZaw4irP1bZguM+7qAYDhxPupO+Ks7/
vkgDh6myzjslSBHCGWqnMLmoiatJinsI2NdFLprGvqZqzJrbIABI54OfFs6B7pSqLdNATbXuRRMUMRX/2+c2WcL6YR62tQU
lgbmiyeZfhsvpUPPAX8QDfkZUPG1xeTn2ao7it+jO48OOPFbzj
+zwt2ZCiIpxN1lR41R0ubD18WQynI9jIz5R3nY6RTtx8kJ4Y
+br62NdpsczIr6tWDgpGfqY6XGyW5SncB2EJ85AIrQrEUFbPva2inHPgxdkCOEgLp8VYwabNXiLJf7D8BiU90QyPUyFdoZW
+8EbfiSLwoDqDbgGgbAbItCrrdwFA6vUBXG2AQfvsdq+9L3OcUmPajCwnTnf/BTMR4mnHT
+0Eu5gamKmb86SjCa9uRzIosVGTvaMLuQBMD
+507rKJwr5+Ia5k9TMjtfNO4oQglNKDVORumiyxOvTDQDYbWcT9B4g9ygOA0GZts7qmJzM44wGtXpfrQpZCWX8X0zKE9Les
+W7h/LqG0qzisKX97vLXIdGUSC14On2iPQWgO3qyXfkt8xwXohc/
veWoRsS9MFUtLHabYJ25o8VZdtwWJfVcDSrCaSFDHJeAFFihA/
F3FlVZuMoWy2DwadoL4xMG2ZmGgs68rPIeLq27dXZFaF5PemCsrQKMhlo4X1tGQROgehoj0Kr8Dab17DgBZJrzvvrecWzoT
WJqGDxkLVlwP9z6fIbEy4vbhe2vhbVtetjlbh0F0JQFnMMet29j6bs
+506K88MPYyHRZBC5HIODmrYDlMjUdPaUGOUYM5VBLqEwnUvCY82SdFMKeuUxVSgQ2dyF2W41Sybf4pzRWfLMzkWuExbIbt
+eXb4nhIGJVGQBXSWefN61Q=",
    "EncapsulationAlgorithm": "ML_KEM"
}
```

Example Response

This example illustrates one usage of Decapsulate.

```
HTTP/1.1 200 OK
Server: Server
Date: Sun, 18 Oct 2026 23:38:19 GMT
Content-Type: application/x-amz-json-1.1
Content-Length: 149
Connection: keep-alive
x-amzn-RequestId: 1e76aa81-a532-11e6-a265-d3aef78e1a90

{
    "KeyId": "1234abcd-12ab-34cd-56ef-1234567890ab",
    "SharedSecret": "oJ2H4lIMwHD2VgqkBf63bGY86fwcyxe2DOu/+EGN5bU=",
    "EncapsulationAlgorithm": "ML_KEM"
}
```

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface V2
- AWS SDK for .NET V4
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- AWS SDK for PHP V3
- AWS SDK for Python (Boto3)
- AWS SDK for Ruby V3

Decrypt

Decrypts ciphertext that was encrypted by a KMS key using any of the following operations:

- Encrypt
- GenerateDataKey
- GenerateDataKeyPair
- GenerateDataKeyWithoutPlaintext
- GenerateDataKeyPairWithoutPlaintext

You can use this operation to decrypt ciphertext that was encrypted under a symmetric encryption KMS key or an asymmetric encryption KMS key. When the KMS key is asymmetric, you must specify the KMS key and the encryption algorithm that was used to encrypt the ciphertext. For information about asymmetric KMS keys, see Asymmetric KMS keys in the *AWS Key Management Service Developer Guide*.

The `Decrypt` operation also decrypts ciphertext that was encrypted outside of AWS KMS by the public key in an AWS KMS asymmetric KMS key. However, it cannot decrypt symmetric ciphertext produced by other libraries, such as the AWS Encryption SDK or Amazon S3 client-side encryption. These libraries return a ciphertext format that is incompatible with AWS KMS.

If the ciphertext was encrypted under a symmetric encryption KMS key, the `KeyId` parameter is optional. AWS KMS can get this information from metadata that it adds to the symmetric ciphertext blob. This feature adds durability to your implementation by ensuring that authorized users can decrypt ciphertext decades after it was encrypted, even if they've lost track of the key ID. However, specifying the KMS key is always recommended as a best practice. When you use the `KeyId` parameter to specify a KMS key, AWS KMS only uses the KMS key you specify. If the ciphertext was encrypted under a different KMS key, the `Decrypt` operation fails. This practice ensures that you use the KMS key that you intend.

Whenever possible, use key policies to give users permission to call the `Decrypt` operation on a particular KMS key, instead of using IAM policies. Otherwise, you might create an IAM policy that gives the user `Decrypt` permission on all KMS keys. This user could decrypt ciphertext that was encrypted by KMS keys in other accounts if the key policy for the cross-account KMS key permits it. If you must use an IAM policy for `Decrypt` permissions, limit the user to particular KMS keys or particular trusted accounts. For details, see Best practices for IAM policies in the *AWS Key Management Service Developer Guide*.

`Decrypt` also supports AWS Nitro Enclaves and NitroTPM, which provide attested environments in Amazon EC2. To call `Decrypt` for a Nitro enclave or NitroTPM, use the AWS Nitro Enclaves SDK or any AWS SDK. Use the `Recipient` parameter to provide the attestation document for the attested environment. Instead of the plaintext data, the response includes the plaintext data encrypted with the public key from the attestation document (`CiphertextForRecipient`). For information about the interaction between AWS KMS and AWS Nitro Enclaves or AWS NitroTPM, see Cryptographic attestation support in AWS KMS in the *AWS Key Management Service Developer Guide*.

The KMS key that you use for this operation must be in a compatible key state. For details, see Key states of AWS KMS keys in the *AWS Key Management Service Developer Guide*.

Cross-account use: Yes. To specify a KMS key in a different AWS account, use the key ARN or alias ARN. A short key ID is also acceptable when decrypting symmetric ciphertexts, though using a full key ARN is recommended to be more explicit about the intended KMS key.

Required permissions: kms:Decrypt (key policy)

Related operations:

- Encrypt
- GenerateDataKey
- GenerateDataKeyPair
- ReEncrypt

Eventual consistency: The AWS KMS API follows an eventual consistency model. For more information, see AWS KMS eventual consistency.

Request Syntax

```
{
   "CiphertextBlob": blob,
   "DryRun": boolean,
   "DryRunModifiers": [ "string" ],
   "EncryptionAlgorithm": "string",
   "EncryptionContext": {
      "string" : "string"
   },
   "GrantTokens": [ "string" ],
```

```
   "KeyId": "string",
   "Recipient": {
      "AttestationDocument": blob,
      "KeyEncryptionAlgorithm": "string"
   }
}
```

Request Parameters

For information about the parameters that are common to all actions, see Common Parameters.

The request accepts the following data in JSON format.

> **Note**
>
> In the following list, the required parameters are described first.

CiphertextBlob

Ciphertext to be decrypted. The blob includes metadata.

This parameter is required in all cases except when `DryRun` is `true` and `DryRunModifiers` is set to `IGNORE_CIPHERTEXT`.

Type: Base64-encoded binary data object

Length Constraints: Minimum length of 1. Maximum length of 6144.

Required: No

DryRun

Checks if your request will succeed. `DryRun` is an optional parameter.

To learn more about how to use this parameter, see Testing your permissions in the *AWS Key Management Service Developer Guide*.

Type: Boolean

Required: No

DryRunModifiers

Specifies the modifiers to apply to the dry run operation. `DryRunModifiers` is an optional parameter that only applies when `DryRun` is set to `true`.

When set to `IGNORE_CIPHERTEXT`, AWS KMS performs only authorization validation without ciphertext validation. This allows you to test permissions without requiring a valid ciphertext blob.

To learn more about how to use this parameter, see Testing your permissions in the *AWS Key Management Service Developer Guide*.

Type: Array of strings

Valid Values: `IGNORE_CIPHERTEXT`

Required: No

EncryptionAlgorithm

Specifies the encryption algorithm that will be used to decrypt the ciphertext. Specify the same algorithm that was used to encrypt the data. If you specify a different algorithm, the `Decrypt` operation fails.

This parameter is required only when the ciphertext was encrypted under an asymmetric KMS key. The default value, `SYMMETRIC_DEFAULT`, represents the only supported algorithm that is valid for symmetric encryption KMS keys.

Type: String

Valid Values: `SYMMETRIC_DEFAULT | RSAES_OAEP_SHA_1 | RSAES_OAEP_SHA_256 | SM2PKE`

Required: No

EncryptionContext

Specifies the encryption context to use when decrypting the data. An encryption context is valid only for cryptographic operations with a symmetric encryption KMS key. The standard asymmetric encryption algorithms and HMAC algorithms that AWS KMS uses do not support an encryption context.

An *encryption context* is a collection of non-secret key-value pairs that represent additional authenticated data. When you use an encryption context to encrypt data, you must specify the same (an exact case-sensitive match) encryption context to decrypt the data. An encryption context is supported only on operations with symmetric encryption KMS keys. On operations with symmetric encryption KMS keys, an encryption context is optional, but it is strongly recommended.

For more information, see Encryption context in the *AWS Key Management Service Developer Guide*.

Type: String to string map

Required: No

GrantTokens

A list of grant tokens.

Use a grant token when your permission to call this operation comes from a new grant that has not yet achieved *eventual consistency*. For more information, see Grant token and Using a grant token in the *AWS Key Management Service Developer Guide*.

Type: Array of strings

Array Members: Minimum number of 0 items. Maximum number of 10 items.

Length Constraints: Minimum length of 1. Maximum length of 8192.

Required: No

KeyId

Specifies the KMS key that AWS KMS uses to decrypt the ciphertext.

Enter a key ID of the KMS key that was used to encrypt the ciphertext. If you identify a different KMS key, the `Decrypt` operation throws an `IncorrectKeyException`.

This parameter is required only when the ciphertext was encrypted under an asymmetric KMS key or when `DryRun` is `true` and `DryRunModifiers` is set to `IGNORE_CIPHERTEXT`. If you used a symmetric encryption KMS key, AWS KMS can get the KMS key from metadata that it adds to the symmetric ciphertext blob. However, it is always recommended as a best practice. This practice ensures that you use the KMS key that you intend.

To specify a KMS key, use its key ID, key ARN, alias name, or alias ARN. When using an alias name, prefix it with `"alias/"`. To specify a KMS key in a different AWS account, you should use the key ARN or alias ARN.

For example:

- Key ID: `1234abcd-12ab-34cd-56ef-1234567890ab`
- Key ARN: `arn:aws:kms:us-east-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab`
- Alias name: `alias/ExampleAlias`
- Alias ARN: `arn:aws:kms:us-east-2:111122223333:alias/ExampleAlias`

To get the key ID and key ARN for a KMS key, use ListKeys or DescribeKey. To get the alias name and alias ARN, use ListAliases.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 2048.

Required: No

Recipient

A signed attestation document from an AWS Nitro enclave or NitroTPM, and the encryption algorithm to use with the public key in the attestation document. The only valid encryption algorithm is `RSAES_OAEP_SHA_256`.

This parameter supports the AWS Nitro Enclaves SDK or any AWS SDK for AWS Nitro Enclaves. It supports any AWS SDK for AWS NitroTPM.

When you use this parameter, instead of returning the plaintext data, AWS KMS encrypts the plaintext data with the public key in the attestation document, and returns the resulting ciphertext in the `CiphertextForRecipient` field in the response. This ciphertext can be decrypted only with the private key in the attested environment. The `Plaintext` field in the response is null or empty.

For information about the interaction between AWS KMS and AWS Nitro Enclaves or AWS NitroTPM, see Cryptographic attestation support in AWS KMS in the *AWS Key Management Service Developer Guide*.

Type: RecipientInfo object

Required: No

Response Syntax

```
{
   "CiphertextForRecipient": blob,
   "EncryptionAlgorithm": "string",
   "KeyId": "string",
   "KeyMaterialId": "string",
   "Plaintext": blob
}
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

CiphertextForRecipient

The plaintext data encrypted with the public key from the attestation document. This ciphertext can be decrypted only by using a private key from the attested environment.

This field is included in the response only when the `Recipient` parameter in the request includes a valid attestation document from an AWS Nitro enclave or NitroTPM. For information about the interaction between AWS KMS and AWS Nitro Enclaves or AWS NitroTPM, see Cryptographic attestation support in AWS KMS in the *AWS Key Management Service Developer Guide*.

Type: Base64-encoded binary data object

Length Constraints: Minimum length of 1. Maximum length of 6144.

EncryptionAlgorithm

The encryption algorithm that was used to decrypt the ciphertext.

Type: String

Valid Values: `SYMMETRIC_DEFAULT | RSAES_OAEP_SHA_1 | RSAES_OAEP_SHA_256 | SM2PKE`

KeyId

The Amazon Resource Name (key ARN) of the KMS key that was used to decrypt the ciphertext.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 2048.

KeyMaterialId

The identifier of the key material used to decrypt the ciphertext. This field is present only when the operation uses a symmetric encryption KMS key. This field is omitted if the request includes the `Recipient` parameter.

Type: String

Length Constraints: Fixed length of 64.

Pattern: `^[a-f0-9]+$`

Plaintext

Decrypted plaintext data. When you use the HTTP API or the AWS CLI, the value is Base64-encoded. Otherwise, it is not Base64-encoded.

If the response includes the `CiphertextForRecipient` field, the `Plaintext` field is null or empty.

Type: Base64-encoded binary data object

Length Constraints: Minimum length of 1. Maximum length of 4096.

Errors

For information about the errors that are common to all actions, see Common Error Types.

DependencyTimeoutException

The system timed out while trying to fulfill the request. You can retry the request.

HTTP Status Code: 500

DisabledException

The request was rejected because the specified KMS key is not enabled.

HTTP Status Code: 400

DryRunOperationException

The request was rejected because the DryRun parameter was specified.

HTTP Status Code: 400

IncorrectKeyException

The request was rejected because the specified KMS key cannot decrypt the data. The `KeyId` in a Decrypt request and the `SourceKeyId` in a ReEncrypt request must identify the same KMS key that was used to encrypt the ciphertext.

HTTP Status Code: 400

InvalidCiphertextException

From the Decrypt or ReEncrypt operation, the request was rejected because the specified ciphertext, or additional authenticated data incorporated into the ciphertext, such as the encryption context, is corrupted, missing, or otherwise invalid.

From the ImportKeyMaterial operation, the request was rejected because AWS KMS could not decrypt the encrypted (wrapped) key material.

From the Decapsulate operation, the request was rejected because the length of the `SharedSecretCiphertextBlob` is inconsistent with the specified ML-KEM key spec.

HTTP Status Code: 400

InvalidGrantTokenException

The request was rejected because the specified grant token is not valid.

HTTP Status Code: 400

InvalidKeyUsageException

The request was rejected for one of the following reasons:

- The `KeyUsage` value of the KMS key is incompatible with the API operation.
- The encryption algorithm or signing algorithm specified for the operation is incompatible with the type of key material in the KMS key (`KeySpec`).

For encrypting, decrypting, re-encrypting, and generating data keys, the `KeyUsage` must be `ENCRYPT_DECRYPT`. For signing and verifying messages, the `KeyUsage` must be `SIGN_VERIFY`.

For generating and verifying message authentication codes (MACs), the KeyUsage must be GENERATE_VERIFY_MAC. For deriving key agreement secrets, the KeyUsage must be KEY_AGREEMENT. For encapsulating and decapsulating shared secrets, the KeyUsage must be ENCAPSULATE_DECAPSULATE. To find the KeyUsage of a KMS key, use the DescribeKey operation.

To find the encryption or signing algorithms supported for a particular KMS key, use the DescribeKey operation.

HTTP Status Code: 400

KeyUnavailableException

The request was rejected because the specified KMS key was not available. You can retry the request.

HTTP Status Code: 500

KMSInternalException

The request was rejected because an internal exception occurred. The request can be retried.

HTTP Status Code: 500

KMSInvalidStateException

The request was rejected because the state of the specified resource is not valid for this request.

This exceptions means one of the following:

- The key state of the KMS key is not compatible with the operation.

  To find the key state, use the DescribeKey operation. For more information about which key states are compatible with each AWS KMS operation, see Key states of AWS KMS keys in the AWS Key Management Service Developer Guide .
- For cryptographic operations on KMS keys in custom key stores, this exception represents a general failure with many possible causes. To identify the cause, see the error message that accompanies the exception.

HTTP Status Code: 400

NotFoundException

The request was rejected because the specified entity or resource could not be found.

HTTP Status Code: 400

Examples

The following examples are formatted for legibility.

Example Request

This example illustrates one usage of Decrypt.

```
POST / HTTP/1.1
Host: kms.us-west-2.amazonaws.com
Content-Length: 293
X-Amz-Target: TrentService.Decrypt
X-Amz-Date: 20160517T204035Z
Content-Type: application/x-amz-json-1.1
Authorization: AWS4-HMAC-SHA256\
 Credential=AKIAI44QH8DHBEXAMPLE/20160517/us-west-2/kms/aws4_request,\
 SignedHeaders=content-type;host;x-amz-date;x-amz-target,\
 Signature=545b0c3bfd9223b8ef7e6293ef3ccac37a83d415ee3112d2e5c70727d2a49c46

{
  "KeyId": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
  "CiphertextBlob":
 "CiDPoCH188S65r5Cy7pAhIFJMXDlU7mewhSlYUpuQIVBrhKmAQEBAgB4z6Ah9fPEuua
+Qsu6QISBSTFw5VO5nsIUpWFKbkCFQa4AAAB9MHsGCSqGSIb3DQEHBqBuMGwCAQAwZwYJKoZIhvcNAQcBMB4GCWCGSAFlAw
ZjYCARCAOt8la8qXLO5wB3JH2NlwWWzWRU2RKqpO9A/0psE5UWwkK6CnwoeC3Zj9Q0A66apZkbRglFfY1lTY
+Tc="
}
```

Example Response

This example illustrates one usage of Decrypt.

```
HTTP/1.1 200 OK
Server: Server
Date: Tue, 17 May 2016 20:40:40 GMT
Content-Type: application/x-amz-json-1.1
Content-Length: 146
Connection: keep-alive
x-amzn-RequestId: 9e02f41f-1c6f-11e6-af63-ab8791945da7
```

```
{
  "KeyId": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
  "KeyMaterialId": "0b7fd7ddbac6eef27907413567cad8c810e2883dc8a7534067a82ee1142fc1e6",
  "Plaintext": "VGhpcyBpcyBEYXkgMSBmb3IgdGhlIEludGVybmV0Cg==",
  "EncryptionAlgorithm": "SYMMETRIC_DEFAULT"
}
```

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface V2
- AWS SDK for .NET V4
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- AWS SDK for PHP V3
- AWS SDK for Python (Boto3)
- AWS SDK for Ruby V3

DeleteAlias

Deletes the specified alias.

> **Note**
>
> Adding, deleting, or updating an alias can allow or deny permission to the KMS key. For details, see ABAC for AWS KMS in the *AWS Key Management Service Developer Guide*.

Because an alias is not a property of a KMS key, you can delete and change the aliases of a KMS key without affecting the KMS key. Also, aliases do not appear in the response from the DescribeKey operation. To get the aliases of all KMS keys, use the ListAliases operation.

Each KMS key can have multiple aliases. To change the alias of a KMS key, use DeleteAlias to delete the current alias and CreateAlias to create a new alias. To associate an existing alias with a different KMS key, call UpdateAlias.

Cross-account use: No. You cannot perform this operation on an alias in a different AWS account.

Required permissions

- kms:DeleteAlias on the alias (IAM policy).
- kms:DeleteAlias on the KMS key (key policy).

For details, see Controlling access to aliases in the *AWS Key Management Service Developer Guide*.

Related operations:

- CreateAlias
- ListAliases
- UpdateAlias

Eventual consistency: The AWS KMS API follows an eventual consistency model. For more information, see AWS KMS eventual consistency.

Request Syntax

```
{
```

```
   "AliasName": "string"
}
```

Request Parameters

For information about the parameters that are common to all actions, see Common Parameters.

The request accepts the following data in JSON format.

> **Note**
>
> In the following list, the required parameters are described first.

AliasName

The alias to be deleted. The alias name must begin with `alias/` followed by the alias name, such as `alias/ExampleAlias`.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 256.

Pattern: `^[a-zA-Z0-9:/_-]+$`

Required: Yes

Response Elements

If the action is successful, the service sends back an HTTP 200 response with an empty HTTP body.

Errors

For information about the errors that are common to all actions, see Common Error Types.

DependencyTimeoutException

The system timed out while trying to fulfill the request. You can retry the request.

HTTP Status Code: 500

KMSInternalException

The request was rejected because an internal exception occurred. The request can be retried.

HTTP Status Code: 500

KMSInvalidStateException

The request was rejected because the state of the specified resource is not valid for this request.

This exceptions means one of the following:

- The key state of the KMS key is not compatible with the operation.

  To find the key state, use the DescribeKey operation. For more information about which key states are compatible with each AWS KMS operation, see Key states of AWS KMS keys in the AWS Key Management Service Developer Guide .
- For cryptographic operations on KMS keys in custom key stores, this exception represents a general failure with many possible causes. To identify the cause, see the error message that accompanies the exception.

HTTP Status Code: 400

NotFoundException

The request was rejected because the specified entity or resource could not be found.

HTTP Status Code: 400

Examples

Example Request

The following example is formatted for legibility.

```
POST / HTTP/1.1
Host: kms.us-east-2.amazonaws.com
Content-Length: 34
X-Amz-Target: TrentService.DeleteAlias
X-Amz-Date: 20161104T183415Z
Content-Type: application/x-amz-json-1.1
Authorization: AWS4-HMAC-SHA256\
 Credential=AKIAI44QH8DHBEXAMPLE/20161104/us-east-2/kms/aws4_request,\
```

```
 SignedHeaders=content-type;host;x-amz-date;x-amz-target,\
 Signature=a57d9c76f60733ea93fe92ac4fa90ca82058a72913e4b8e52c262ffc96704d53

{"AliasName": "alias/ExampleAlias"}
```

Example Response

This example illustrates one usage of DeleteAlias.

```
HTTP/1.1 200 OK
Server: Server
Date: Fri, 04 Nov 2016 18:34:15 GMT
Content-Type: application/x-amz-json-1.1
Content-Length: 0
Connection: keep-alive
x-amzn-RequestId: 4a2313ae-a2bd-11e6-aea3-9bf897a0ae69
```

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface V2
- AWS SDK for .NET V4
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- AWS SDK for PHP V3
- AWS SDK for Python (Boto3)
- AWS SDK for Ruby V3

DeleteCustomKeyStore

Deletes a custom key store. This operation does not affect any backing elements of the custom key store. It does not delete the AWS CloudHSM cluster that is associated with an AWS CloudHSM key store, or affect any users or keys in the cluster. For an external key store, it does not affect the external key store proxy, external key manager, or any external keys.

This operation is part of the custom key stores feature in AWS KMS, which combines the convenience and extensive integration of AWS KMS with the isolation and control of a key store that you own and manage.

The custom key store that you delete cannot contain any KMS keys. Before deleting the key store, verify that you will never need to use any of the KMS keys in the key store for any cryptographic operations. Then, use ScheduleKeyDeletion to delete the KMS keys from the key store. After the required waiting period expires and all KMS keys are deleted from the custom key store, use DisconnectCustomKeyStore to disconnect the key store from AWS KMS. Then, you can delete the custom key store.

For keys in an AWS CloudHSM key store, the `ScheduleKeyDeletion` operation makes a best effort to delete the key material from the associated cluster. However, you might need to manually delete the orphaned key material from the cluster and its backups. AWS KMS never creates, manages, or deletes cryptographic keys in the external key manager associated with an external key store. You must manage them using your external key manager tools.

Instead of deleting the custom key store, consider using the DisconnectCustomKeyStore operation to disconnect the custom key store from its backing key store. While the key store is disconnected, you cannot create or use the KMS keys in the key store. But, you do not need to delete KMS keys and you can reconnect a disconnected custom key store at any time.

If the operation succeeds, it returns a JSON object with no properties.

Cross-account use: No. You cannot perform this operation on a custom key store in a different AWS account.

Required permissions: kms:DeleteCustomKeyStore (IAM policy)

Related operations:

- ConnectCustomKeyStore
- CreateCustomKeyStore

- DescribeCustomKeyStores
- DisconnectCustomKeyStore
- UpdateCustomKeyStore

Eventual consistency: The AWS KMS API follows an eventual consistency model. For more information, see AWS KMS eventual consistency.

Request Syntax

```
{
   "CustomKeyStoreId": "string"
}
```

Request Parameters

For information about the parameters that are common to all actions, see Common Parameters.

The request accepts the following data in JSON format.

> **Note**
>
> In the following list, the required parameters are described first.

CustomKeyStoreId

Enter the ID of the custom key store you want to delete. To find the ID of a custom key store, use the DescribeCustomKeyStores operation.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 64.

Required: Yes

Response Elements

If the action is successful, the service sends back an HTTP 200 response with an empty HTTP body.

Errors

For information about the errors that are common to all actions, see Common Error Types.

CustomKeyStoreHasCMKsException

The request was rejected because the custom key store contains KMS keys. After verifying that you do not need to use the KMS keys, use the ScheduleKeyDeletion operation to delete the KMS keys. After they are deleted, you can delete the custom key store.

HTTP Status Code: 400

CustomKeyStoreInvalidStateException

The request was rejected because of the `ConnectionState` of the custom key store. To get the `ConnectionState` of a custom key store, use the DescribeCustomKeyStores operation.

This exception is thrown under the following conditions:

- You requested the ConnectCustomKeyStore operation on a custom key store with a `ConnectionState` of `DISCONNECTING` or `FAILED`. This operation is valid for all other `ConnectionState` values. To reconnect a custom key store in a `FAILED` state, disconnect it (DisconnectCustomKeyStore), then connect it (`ConnectCustomKeyStore`).
- You requested the CreateKey operation in a custom key store that is not connected. This operations is valid only when the custom key store `ConnectionState` is `CONNECTED`.
- You requested the DisconnectCustomKeyStore operation on a custom key store with a `ConnectionState` of `DISCONNECTING` or `DISCONNECTED`. This operation is valid for all other `ConnectionState` values.
- You requested the UpdateCustomKeyStore or DeleteCustomKeyStore operation on a custom key store that is not disconnected. `UpdateCustomKeyStore` can be called on a custom key store in the `CONNECTED` state only to update `NewCustomKeyStoreName`. For all other properties, the custom key store `ConnectionState` must be `DISCONNECTED`.
- You requested the GenerateRandom operation in an AWS CloudHSM key store that is not connected. This operation is valid only when the AWS CloudHSM key store `ConnectionState` is `CONNECTED`.

HTTP Status Code: 400

CustomKeyStoreNotFoundException

The request was rejected because AWS KMS cannot find a custom key store with the specified key store name or ID.

HTTP Status Code: 400

KMSInternalException

The request was rejected because an internal exception occurred. The request can be retried.

HTTP Status Code: 500

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface V2
- AWS SDK for .NET V4
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- AWS SDK for PHP V3
- AWS SDK for Python (Boto3)
- AWS SDK for Ruby V3

DeleteImportedKeyMaterial

Deletes key material that was previously imported. This operation makes the specified KMS key temporarily unusable. To restore the usability of the KMS key, reimport the same key material. For more information about importing key material into AWS KMS, see Importing Key Material in the *AWS Key Management Service Developer Guide*.

When the specified KMS key is in the `PendingDeletion` state, this operation does not change the KMS key's state. Otherwise, it changes the KMS key's state to `PendingImport`.

Considerations for multi-Region symmetric encryption keys

- When you delete the key material of a primary Region key that is in `PENDING_ROTATION` or `PENDING_MULTI_REGION_IMPORT_AND_ROTATION`state, you'll also be deleting the key materials for the replica Region keys.
- If you delete any key material of a replica Region key, the primary Region key and other replica Region keys remain unchanged.

The KMS key that you use for this operation must be in a compatible key state. For details, see Key states of AWS KMS keys in the *AWS Key Management Service Developer Guide*.

Cross-account use: No. You cannot perform this operation on a KMS key in a different AWS account.

Required permissions: kms:DeleteImportedKeyMaterial (key policy)

Related operations:

- GetParametersForImport
- ListKeyRotations
- ImportKeyMaterial

Eventual consistency: The AWS KMS API follows an eventual consistency model. For more information, see AWS KMS eventual consistency.

Request Syntax

```
{
```

```
    "KeyId": "string",
    "KeyMaterialId": "string"
}
```

Request Parameters

For information about the parameters that are common to all actions, see Common Parameters.

The request accepts the following data in JSON format.

> **Note**
>
> In the following list, the required parameters are described first.

KeyId

Identifies the KMS key from which you are deleting imported key material. The `Origin` of the KMS key must be `EXTERNAL`.

Specify the key ID or key ARN of the KMS key.

For example:

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

To get the key ID and key ARN for a KMS key, use ListKeys or DescribeKey.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 2048.

Required: Yes

KeyMaterialId

Identifies the imported key material you are deleting.

> **Important**
>
> If no KeyMaterialId is specified, AWS KMS deletes the current key material.

To get the list of key material IDs associated with a KMS key, use ListKeyRotations.

Type: String

Length Constraints: Fixed length of 64.

Pattern: ^[a-f0-9]+$

Required: No

Response Syntax

```
{
   "KeyId": "string",
   "KeyMaterialId": "string"
}
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

KeyId

The Amazon Resource Name (key ARN) of the KMS key from which the key material was deleted.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 2048.

KeyMaterialId

Identifies the deleted key material.

Type: String

Length Constraints: Minimum length of 0. Maximum length of 64.

Pattern: ^[a-f0-9]+$

Errors

For information about the errors that are common to all actions, see Common Error Types.

DependencyTimeoutException

The system timed out while trying to fulfill the request. You can retry the request.

HTTP Status Code: 500

InvalidArnException

The request was rejected because a specified ARN, or an ARN in a key policy, is not valid.

HTTP Status Code: 400

KMSInternalException

The request was rejected because an internal exception occurred. The request can be retried.

HTTP Status Code: 500

KMSInvalidStateException

The request was rejected because the state of the specified resource is not valid for this request.

This exceptions means one of the following:

- The key state of the KMS key is not compatible with the operation.

  To find the key state, use the DescribeKey operation. For more information about which key states are compatible with each AWS KMS operation, see Key states of AWS KMS keys in the AWS Key Management Service Developer Guide .
- For cryptographic operations on KMS keys in custom key stores, this exception represents a general failure with many possible causes. To identify the cause, see the error message that accompanies the exception.

HTTP Status Code: 400

NotFoundException

The request was rejected because the specified entity or resource could not be found.

HTTP Status Code: 400

UnsupportedOperationException

The request was rejected because a specified parameter is not supported or a specified resource is not valid for this operation.

HTTP Status Code: 400

Examples

Example Request

The following example is formatted for legibility.

```
POST / HTTP/1.1
Host: kms.us-east-2.amazonaws.com
Content-Length: 48
X-Amz-Target: TrentService.DeleteImportedKeyMaterial
X-Amz-Date: 20161107T213532Z
Content-Type: application/x-amz-json-1.1
Authorization: AWS4-HMAC-SHA256\
 Credential=AKIAI44QH8DHBEXAMPLE/20161107/us-east-2/kms/aws4_request,\
 SignedHeaders=content-type;host;x-amz-date;x-amz-target,\
 Signature=2cea34fe55d5858295a377448a1e053d0edd45ce571da7cf69b202905759f272

{"KeyId": "1234abcd-12ab-34cd-56ef-1234567890ab"}
```

Example Response

This example illustrates one usage of DeleteImportedKeyMaterial.

```
HTTP/1.1 200 OK
Server: Server
Date: Mon, 07 Nov 2016 21:35:35 GMT
Content-Type: application/x-amz-json-1.1
Content-Length: 0
Connection: keep-alive
x-amzn-RequestId: 1e76aa81-a532-11e6-a265-d3aef78e1a90

{
   "KeyId": "1234abcd-12ab-34cd-56ef-1234567890ab",
   "KeyMaterialId": "0b7fd7ddbac6eef27907413567cad8c810e2883dc8a7534067a82ee1142fc1e6",
```

```
}
```

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface V2
- AWS SDK for .NET V4
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- AWS SDK for PHP V3
- AWS SDK for Python (Boto3)
- AWS SDK for Ruby V3

DeriveSharedSecret

Derives a shared secret using a key agreement algorithm.

> **Note**
>
> You must use an asymmetric NIST-standard elliptic curve (ECC) or SM2 (China Regions only) KMS key pair with a `KeyUsage` value of `KEY_AGREEMENT` to call DeriveSharedSecret.

DeriveSharedSecret uses the Elliptic Curve Cryptography Cofactor Diffie-Hellman Primitive (ECDH) to establish a key agreement between two peers by deriving a shared secret from their elliptic curve public-private key pairs. You can use the raw shared secret that DeriveSharedSecret returns to derive a symmetric key that can encrypt and decrypt data that is sent between the two peers, or that can generate and verify HMACs. AWS KMS recommends that you follow NIST recommendations for key derivation when using the raw shared secret to derive a symmetric key.

The following workflow demonstrates how to establish key agreement over an insecure communication channel using DeriveSharedSecret.

1. **Alice** calls CreateKey to create an asymmetric KMS key pair with a `KeyUsage` value of `KEY_AGREEMENT`.

   The asymmetric KMS key must use a NIST-standard elliptic curve (ECC) or SM2 (China Regions only) key spec.

2. **Bob** creates an elliptic curve key pair.

   Bob can call CreateKey to create an asymmetric KMS key pair or generate a key pair outside of AWS KMS. Bob's key pair must use the same NIST-standard elliptic curve (ECC) or SM2 (China Regions ony) curve as Alice.

3. Alice and Bob **exchange their public keys** through an insecure communication channel (like the internet).

   Use GetPublicKey to download the public key of your asymmetric KMS key pair.

> **Note**
>
> AWS KMS strongly recommends verifying that the public key you receive came from the expected party before using it to derive a shared secret.

4. **Alice** calls DeriveSharedSecret.

   AWS KMS uses the private key from the KMS key pair generated in **Step 1**, Bob's public key, and the Elliptic Curve Cryptography Cofactor Diffie-Hellman Primitive to derive the shared secret. The private key in your KMS key pair never leaves AWS KMS unencrypted. DeriveSharedSecret returns the raw shared secret.

5. **Bob** uses the Elliptic Curve Cryptography Cofactor Diffie-Hellman Primitive to calculate the same raw secret using his private key and Alice's public key.

To derive a shared secret you must provide a key agreement algorithm, the private key of the caller's asymmetric NIST-standard elliptic curve or SM2 (China Regions only) KMS key pair, and the public key from your peer's NIST-standard elliptic curve or SM2 (China Regions only) key pair. The public key can be from another asymmetric KMS key pair or from a key pair generated outside of AWS KMS, but both key pairs must be on the same elliptic curve.

The KMS key that you use for this operation must be in a compatible key state. For details, see Key states of AWS KMS keys in the *AWS Key Management Service Developer Guide*.

Cross-account use: Yes. To perform this operation with a KMS key in a different AWS account, specify the key ARN or alias ARN in the value of the `KeyId` parameter.

Required permissions: kms:DeriveSharedSecret (key policy)

Related operations:

- CreateKey
- GetPublicKey
- DescribeKey

Eventual consistency: The AWS KMS API follows an eventual consistency model. For more information, see AWS KMS eventual consistency.

Request Syntax

```
{
   "DryRun": boolean,
   "GrantTokens": [ "string" ],
   "KeyAgreementAlgorithm": "string",
   "KeyId": "string",
   "PublicKey": blob,
   "Recipient": {
      "AttestationDocument": blob,
      "KeyEncryptionAlgorithm": "string"
   }
}
```

Request Parameters

For information about the parameters that are common to all actions, see Common Parameters.

The request accepts the following data in JSON format.

> **Note**
>
> In the following list, the required parameters are described first.

KeyAgreementAlgorithm

Specifies the key agreement algorithm used to derive the shared secret. The only valid value is ECDH.

Type: String

Valid Values: ECDH

Required: Yes

KeyId

Identifies an asymmetric NIST-standard ECC or SM2 (China Regions only) KMS key. AWS KMS uses the private key in the specified key pair to derive the shared secret. The key usage of the KMS key must be KEY_AGREEMENT. To find the KeyUsage of a KMS key, use the DescribeKey operation.

To specify a KMS key, use its key ID, key ARN, alias name, or alias ARN. When using an alias name, prefix it with `"alias/"`. To specify a KMS key in a different AWS account, you must use the key ARN or alias ARN.

For example:

- Key ID: `1234abcd-12ab-34cd-56ef-1234567890ab`
- Key ARN: `arn:aws:kms:us-east-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab`
- Alias name: `alias/ExampleAlias`
- Alias ARN: `arn:aws:kms:us-east-2:111122223333:alias/ExampleAlias`

To get the key ID and key ARN for a KMS key, use ListKeys or DescribeKey. To get the alias name and alias ARN, use ListAliases.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 2048.

Required: Yes

PublicKey

Specifies the public key in your peer's NIST-standard elliptic curve (ECC) or SM2 (China Regions only) key pair.

The public key must be a DER-encoded X.509 public key, also known as `SubjectPublicKeyInfo` (SPKI), as defined in RFC 5280.

GetPublicKey returns the public key of an asymmetric KMS key pair in the required DER-encoded format.

> **Note**
>
> If you use AWS CLI version 1, you must provide the DER-encoded X.509 public key in a file. Otherwise, the AWS CLI Base64-encodes the public key a second time, resulting in a `ValidationException`.

You can specify the public key as binary data in a file using fileb (`fileb://<path-to-file>`) or in-line using a Base64 encoded string.

Type: Base64-encoded binary data object

Length Constraints: Minimum length of 1. Maximum length of 8192.

Required: Yes

DryRun

Checks if your request will succeed. `DryRun` is an optional parameter.

To learn more about how to use this parameter, see Testing your permissions in the *AWS Key Management Service Developer Guide*.

Type: Boolean

Required: No

GrantTokens

A list of grant tokens.

Use a grant token when your permission to call this operation comes from a new grant that has not yet achieved *eventual consistency*. For more information, see Grant token and Using a grant token in the *AWS Key Management Service Developer Guide*.

Type: Array of strings

Array Members: Minimum number of 0 items. Maximum number of 10 items.

Length Constraints: Minimum length of 1. Maximum length of 8192.

Required: No

Recipient

A signed attestation document from an AWS Nitro enclave or NitroTPM, and the encryption algorithm to use with the public key in the attestation document. The only valid encryption algorithm is `RSAES_OAEP_SHA_256`.

This parameter only supports attestation documents for AWS Nitro Enclaves or AWS NitroTPM. To call DeriveSharedSecret generate an attestation document use either AWS Nitro Enclaves SDK for an AWS Nitro Enclaves or AWS NitroTPM tools for AWS NitroTPM. Then use the Recipient parameter from any AWS SDK to provide the attestation document for the attested environment.

When you use this parameter, instead of returning a plaintext copy of the shared secret, AWS KMS encrypts the plaintext shared secret under the public key in the attestation document, and returns the resulting ciphertext in the `CiphertextForRecipient` field in the response. This ciphertext can be decrypted only with the private key in the attested environment. The `CiphertextBlob` field in the response contains the encrypted shared secret derived from the KMS key specified by the `KeyId` parameter and public key specified by the `PublicKey` parameter. The `SharedSecret` field in the response is null or empty.

For information about the interaction between AWS KMS and AWS Nitro Enclaves or AWS NitroTPM, see Cryptographic attestation support in AWS KMS in the *AWS Key Management Service Developer Guide*.

Type: RecipientInfo object

Required: No

Response Syntax

```
{
   "CiphertextForRecipient": blob,
   "KeyAgreementAlgorithm": "string",
   "KeyId": "string",
   "KeyOrigin": "string",
   "SharedSecret": blob
}
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

CiphertextForRecipient

The plaintext shared secret encrypted with the public key from the attestation document. This ciphertext can be decrypted only by using a private key from the attested environment.

This field is included in the response only when the `Recipient` parameter in the request includes a valid attestation document from an AWS Nitro enclave or NitroTPM. For information about the interaction between AWS KMS and AWS Nitro Enclaves or AWS NitroTPM, see

Cryptographic attestation support in AWS KMS in the *AWS Key Management Service Developer Guide*.

Type: Base64-encoded binary data object

Length Constraints: Minimum length of 1. Maximum length of 6144.

KeyAgreementAlgorithm

Identifies the key agreement algorithm used to derive the shared secret.

Type: String

Valid Values: ECDH

KeyId

Identifies the KMS key used to derive the shared secret.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 2048.

KeyOrigin

The source of the key material for the specified KMS key.

When this value is AWS_KMS, AWS KMS created the key material. When this value is EXTERNAL, the key material was imported or the KMS key doesn't have any key material.

The only valid values for DeriveSharedSecret are AWS_KMS and EXTERNAL. DeriveSharedSecret does not support KMS keys with a KeyOrigin value of AWS_CLOUDHSM or EXTERNAL_KEY_STORE.

Type: String

Valid Values: AWS_KMS | EXTERNAL | AWS_CLOUDHSM | EXTERNAL_KEY_STORE

SharedSecret

The raw secret derived from the specified key agreement algorithm, private key in the asymmetric KMS key, and your peer's public key.

If the response includes the CiphertextForRecipient field, the SharedSecret field is null or empty.

Type: Base64-encoded binary data object

Length Constraints: Minimum length of 1. Maximum length of 4096.

Errors

For information about the errors that are common to all actions, see Common Error Types.

DependencyTimeoutException

The system timed out while trying to fulfill the request. You can retry the request.

HTTP Status Code: 500

DisabledException

The request was rejected because the specified KMS key is not enabled.

HTTP Status Code: 400

DryRunOperationException

The request was rejected because the DryRun parameter was specified.

HTTP Status Code: 400

InvalidGrantTokenException

The request was rejected because the specified grant token is not valid.

HTTP Status Code: 400

InvalidKeyUsageException

The request was rejected for one of the following reasons:

- The `KeyUsage` value of the KMS key is incompatible with the API operation.
- The encryption algorithm or signing algorithm specified for the operation is incompatible with the type of key material in the KMS key (`KeySpec`).

For encrypting, decrypting, re-encrypting, and generating data keys, the `KeyUsage` must be `ENCRYPT_DECRYPT`. For signing and verifying messages, the `KeyUsage` must be `SIGN_VERIFY`. For generating and verifying message authentication codes (MACs), the `KeyUsage` must be `GENERATE_VERIFY_MAC`. For deriving key agreement secrets, the `KeyUsage` must be

KEY_AGREEMENT. For encapsulating and decapsulating shared secrets, the KeyUsage must be ENCAPSULATE_DECAPSULATE. To find the KeyUsage of a KMS key, use the DescribeKey operation.

To find the encryption or signing algorithms supported for a particular KMS key, use the DescribeKey operation.

HTTP Status Code: 400

KeyUnavailableException

The request was rejected because the specified KMS key was not available. You can retry the request.

HTTP Status Code: 500

KMSInternalException

The request was rejected because an internal exception occurred. The request can be retried.

HTTP Status Code: 500

KMSInvalidStateException

The request was rejected because the state of the specified resource is not valid for this request.

This exceptions means one of the following:

- The key state of the KMS key is not compatible with the operation.

  To find the key state, use the DescribeKey operation. For more information about which key states are compatible with each AWS KMS operation, see Key states of AWS KMS keys in the AWS Key Management Service Developer Guide .
- For cryptographic operations on KMS keys in custom key stores, this exception represents a general failure with many possible causes. To identify the cause, see the error message that accompanies the exception.

HTTP Status Code: 400

NotFoundException

The request was rejected because the specified entity or resource could not be found.

HTTP Status Code: 400

Examples

Example Request

The following example is formatted for legibility.

```
POST / HTTP/1.1
Host: kms.us-east-2.amazonaws.com
Content-Length: 48
X-Amz-Target: TrentService.DeriveSharedSecret
X-Amz-Date: 20161107T213532Z
Content-Type: application/x-amz-json-1.1
Authorization: AWS4-HMAC-SHA256\
 Credential=AKIAI44QH8DHBEXAMPLE/20161107/us-east-2/kms/aws4_request,\
 SignedHeaders=content-type;host;x-amz-date;x-amz-target,\
 Signature=2cea34fe55d5858295a377448a1e053d0edd45ce571da7cf69b202905759f272

{
    "KeyId": "1234abcd-12ab-34cd-56ef-1234567890ab",
    "PublicKey":
 "MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAvH3Yj0wbkLEpUl95Cv1cJVjsVNSjwGq3tCLnzXfhVwVvmzGN8
+iSK341kr2kFTpINN7T1ZaX9vfXBdGR+VtkRKMWoHQeWzHrPZ+3irvpXNCKxGUxmPNsJSjPUhuSXT5+0VrY/
LEYLQ5lUTrhU6z5/OK0kzaCc66DXc5ipSloS4Xyg
+QcYSMxe9xuqO5HtzFImUSKBm1W6eDT6lHnSbpi7vXzNbIX7pWxKw9nmQvQIDAQAB",
    "KeyAgreementAlgorithm": "ECDH"
}
```

Example Response

This example illustrates one usage of DeriveSharedSecret.

```
HTTP/1.1 200 OK
Server: Server
Date: Mon, 07 Nov 2016 21:35:35 GMT
Content-Type: application/x-amz-json-1.1
Content-Length: 0
Connection: keep-alive
x-amzn-RequestId: 1e76aa81-a532-11e6-a265-d3aef78e1a90

{
   "KeyId": "1234abcd-12ab-34cd-56ef-1234567890ab",
   "KeyAgreementAlgorithm": "ECDH",
```

```
   "SharedSecret": "MEYCIQCKZLWyTk5runarx6XiAkU9gv3lbwPO/pHa
+DXFehzdDwIhANwpsIV2g/9SPWLLsF6p/hiSskuIXMTRwqrMdVKWTMHG"
}
```

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface V2
- AWS SDK for .NET V4
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- AWS SDK for PHP V3
- AWS SDK for Python (Boto3)
- AWS SDK for Ruby V3

DescribeCustomKeyStores

Gets information about custom key stores in the account and Region.

This operation is part of the custom key stores feature in AWS KMS, which combines the convenience and extensive integration of AWS KMS with the isolation and control of a key store that you own and manage.

By default, this operation returns information about all custom key stores in the account and Region. To get only information about a particular custom key store, use either the `CustomKeyStoreName` or `CustomKeyStoreId` parameter (but not both).

To determine whether the custom key store is connected to its AWS CloudHSM cluster or external key store proxy, use the `ConnectionState` element in the response. If an attempt to connect the custom key store failed, the `ConnectionState` value is `FAILED` and the `ConnectionErrorCode` element in the response indicates the cause of the failure. For help interpreting the `ConnectionErrorCode`, see CustomKeyStoresListEntry.

Custom key stores have a `DISCONNECTED` connection state if the key store has never been connected or you used the DisconnectCustomKeyStore operation to disconnect it. Otherwise, the connection state is CONNECTED. If your custom key store connection state is `CONNECTED` but you are having trouble using it, verify that the backing store is active and available. For an AWS CloudHSM key store, verify that the associated AWS CloudHSM cluster is active and contains the minimum number of HSMs required for the operation, if any. For an external key store, verify that the external key store proxy and its associated external key manager are reachable and enabled.

For help repairing your AWS CloudHSM key store, see the Troubleshooting AWS CloudHSM key stores. For help repairing your external key store, see the Troubleshooting external key stores. Both topics are in the *AWS Key Management Service Developer Guide*.

Cross-account use: No. You cannot perform this operation on a custom key store in a different AWS account.

Required permissions: kms:DescribeCustomKeyStores (IAM policy)

Related operations:

- ConnectCustomKeyStore
- CreateCustomKeyStore

- DeleteCustomKeyStore
- DisconnectCustomKeyStore
- UpdateCustomKeyStore

Eventual consistency: The AWS KMS API follows an eventual consistency model. For more information, see AWS KMS eventual consistency.

Request Syntax

```
{
   "CustomKeyStoreId": "string",
   "CustomKeyStoreName": "string",
   "Limit": number,
   "Marker": "string"
}
```

Request Parameters

For information about the parameters that are common to all actions, see Common Parameters.

The request accepts the following data in JSON format.

> **Note**
>
> In the following list, the required parameters are described first.

CustomKeyStoreId

Gets only information about the specified custom key store. Enter the key store ID.

By default, this operation gets information about all custom key stores in the account and Region. To limit the output to a particular custom key store, provide either the `CustomKeyStoreId` or `CustomKeyStoreName` parameter, but not both.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 64.

Required: No

CustomKeyStoreName

Gets only information about the specified custom key store. Enter the friendly name of the custom key store.

By default, this operation gets information about all custom key stores in the account and Region. To limit the output to a particular custom key store, provide either the `CustomKeyStoreId` or `CustomKeyStoreName` parameter, but not both.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 256.

Required: No

Limit

Use this parameter to specify the maximum number of items to return. When this value is present, AWS KMS does not return more than the specified number of items, but it might return fewer.

Type: Integer

Valid Range: Minimum value of 1. Maximum value of 1000.

Required: No

Marker

Use this parameter in a subsequent request after you receive a response with truncated results. Set it to the value of `NextMarker` from the truncated response you just received.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 1024.

Pattern: `[\u0020-\u00FF]*`

Required: No

Response Syntax

```
{
```

```
   "CustomKeyStores": [
      {
         "CloudHsmClusterId": "string",
         "ConnectionErrorCode": "string",
         "ConnectionState": "string",
         "CreationDate": number,
         "CustomKeyStoreId": "string",
         "CustomKeyStoreName": "string",
         "CustomKeyStoreType": "string",
         "TrustAnchorCertificate": "string",
         "XksProxyConfiguration": {
            "AccessKeyId": "string",
            "Connectivity": "string",
            "UriEndpoint": "string",
            "UriPath": "string",
            "VpcEndpointServiceName": "string",
            "VpcEndpointServiceOwner": "string"
         }
      }
   ],
   "NextMarker": "string",
   "Truncated": boolean
}
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

CustomKeyStores

Contains metadata about each custom key store.

Type: Array of CustomKeyStoresListEntry objects

NextMarker

When `Truncated` is true, this element is present and contains the value to use for the `Marker` parameter in a subsequent request.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 1024.

Pattern: `[\u0020-\u00FF]*`

Truncated

A flag that indicates whether there are more items in the list. When this value is true, the list in this response is truncated. To get more items, pass the value of the `NextMarker` element in this response to the `Marker` parameter in a subsequent request.

Type: Boolean

Errors

For information about the errors that are common to all actions, see Common Error Types.

CustomKeyStoreNotFoundException

The request was rejected because AWS KMS cannot find a custom key store with the specified key store name or ID.

HTTP Status Code: 400

InvalidMarkerException

The request was rejected because the marker that specifies where pagination should next begin is not valid.

HTTP Status Code: 400

KMSInternalException

The request was rejected because an internal exception occurred. The request can be retried.

HTTP Status Code: 500

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface V2
- AWS SDK for .NET V4
- AWS SDK for C++

- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- AWS SDK for PHP V3
- AWS SDK for Python (Boto3)
- AWS SDK for Ruby V3

DescribeKey

Provides detailed information about a KMS key. You can run `DescribeKey` on a customer managed key or an AWS managed key.

This detailed information includes the key ARN, creation date (and deletion date, if applicable), the key state, and the origin and expiration date (if any) of the key material. It includes fields, like `KeySpec`, that help you distinguish different types of KMS keys. It also displays the key usage (encryption, signing, or generating and verifying MACs) and the algorithms that the KMS key supports.

For multi-Region keys, `DescribeKey` displays the primary key and all related replica keys. For KMS keys in AWS CloudHSM key stores, it includes information about the key store, such as the key store ID and the AWS CloudHSM cluster ID. For KMS keys in external key stores, it includes the custom key store ID and the ID of the external key.

`DescribeKey` does not return the following information:

- Aliases associated with the KMS key. To get this information, use ListAliases.
- Whether automatic key rotation is enabled on the KMS key. To get this information, use GetKeyRotationStatus. Also, some key states prevent a KMS key from being automatically rotated. For details, see How key rotation works in the *AWS Key Management Service Developer Guide*.
- Tags on the KMS key. To get this information, use ListResourceTags.
- Key policies and grants on the KMS key. To get this information, use GetKeyPolicy and ListGrants.

In general, `DescribeKey` is a non-mutating operation. It returns data about KMS keys, but doesn't change them. However, AWS services use `DescribeKey` to create AWS managed keys from a *predefined AWS alias* with no key ID.

Cross-account use: Yes. To perform this operation with a KMS key in a different AWS account, specify the key ARN or alias ARN in the value of the `KeyId` parameter.

Required permissions: kms:DescribeKey (key policy)

Related operations:

- GetKeyPolicy

- GetKeyRotationStatus
- ListAliases
- ListGrants
- ListKeys
- ListResourceTags
- ListRetirableGrants

Eventual consistency: The AWS KMS API follows an eventual consistency model. For more information, see AWS KMS eventual consistency.

Request Syntax

```
{
   "GrantTokens": [ "string" ],
   "KeyId": "string"
}
```

Request Parameters

For information about the parameters that are common to all actions, see Common Parameters.

The request accepts the following data in JSON format.

> **Note**
>
> In the following list, the required parameters are described first.

KeyId

Describes the specified KMS key.

If you specify a predefined AWS alias (an AWS alias with no key ID), AWS KMS associates the alias with an AWS managed key and returns its `KeyId` and `Arn` in the response.

To specify a KMS key, use its key ID, key ARN, alias name, or alias ARN. When using an alias name, prefix it with `"alias/"`. To specify a KMS key in a different AWS account, you must use the key ARN or alias ARN.

For example:

- Key ID: `1234abcd-12ab-34cd-56ef-1234567890ab`
- Key ARN: `arn:aws:kms:us-east-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab`
- Alias name: `alias/ExampleAlias`
- Alias ARN: `arn:aws:kms:us-east-2:111122223333:alias/ExampleAlias`

To get the key ID and key ARN for a KMS key, use ListKeys or DescribeKey. To get the alias name and alias ARN, use ListAliases.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 2048.

Required: Yes

GrantTokens

A list of grant tokens.

Use a grant token when your permission to call this operation comes from a new grant that has not yet achieved *eventual consistency*. For more information, see Grant token and Using a grant token in the *AWS Key Management Service Developer Guide*.

Type: Array of strings

Array Members: Minimum number of 0 items. Maximum number of 10 items.

Length Constraints: Minimum length of 1. Maximum length of 8192.

Required: No

Response Syntax

```
{
   "KeyMetadata": {
      "Arn": "string",
      "AWSAccountId": "string",
      "CloudHsmClusterId": "string",
      "CreationDate": number,
      "CurrentKeyMaterialId": "string",
      "CustomerMasterKeySpec": "string",
```

```
      "CustomKeyStoreId": "string",
      "DeletionDate": number,
      "Description": "string",
      "Enabled": boolean,
      "EncapsulationAlgorithms": [ "string" ],
      "EncryptionAlgorithms": [ "string" ],
      "ExpirationModel": "string",
      "KeyAgreementAlgorithms": [ "string" ],
      "KeyId": "string",
      "KeyManager": "string",
      "KeySpec": "string",
      "KeyState": "string",
      "KeyUsage": "string",
      "MacAlgorithms": [ "string" ],
      "MultiRegion": boolean,
      "MultiRegionConfiguration": {
         "MultiRegionKeyType": "string",
         "PrimaryKey": {
            "Arn": "string",
            "Region": "string"
         },
         "ReplicaKeys": [
            {
               "Arn": "string",
               "Region": "string"
            }
         ]
      },
      "Origin": "string",
      "PendingDeletionWindowInDays": number,
      "SigningAlgorithms": [ "string" ],
      "ValidTo": number,
      "XksKeyConfiguration": {
         "Id": "string"
      }
   }
}
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

KeyMetadata

Metadata associated with the key.

Type: KeyMetadata object

Errors

For information about the errors that are common to all actions, see Common Error Types.

DependencyTimeoutException

The system timed out while trying to fulfill the request. You can retry the request.

HTTP Status Code: 500

InvalidArnException

The request was rejected because a specified ARN, or an ARN in a key policy, is not valid.

HTTP Status Code: 400

KMSInternalException

The request was rejected because an internal exception occurred. The request can be retried.

HTTP Status Code: 500

NotFoundException

The request was rejected because the specified entity or resource could not be found.

HTTP Status Code: 400

Examples

The following examples are formatted for legibility.

Example Request

This example illustrates one usage of DescribeKey.

```
POST / HTTP/1.1
Host: kms.us-east-2.amazonaws.com
Content-Length: 49
X-Amz-Target: TrentService.DescribeKey
X-Amz-Date: 20170705T211529Z
Authorization: AWS4-HMAC-SHA256\
 Credential=AKIAI44QH8DHBEXAMPLE/20170705/us-east-2/kms/aws4_request,\
 SignedHeaders=content-type;host;x-amz-date;x-amz-target,\
 Signature=6bcb6a5ef9ee7585d83955e8a5c3f6d47cf581596208fc0e436fa1de26ef3f6a
Content-Type: application/x-amz-json-1.1

{"KeyId": "1234abcd-12ab-34cd-56ef-1234567890ab"}
```

Example Response

This example illustrates one usage of DescribeKey.

```
HTTP/1.1 200 OK
Server: Server
Date: Wed, 05 Jul 2017 21:15:30 GMT
Content-Type: application/x-amz-json-1.1
Content-Length: 335
Connection: keep-alive
x-amzn-RequestId: 13230ddb-61c7-11e7-af6f-c5b105d7a982

{
  "KeyMetadata": {
    "AWSAccountId": "111122223333",
    "Arn": "arn:aws:kms:us-
east-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
    "CreationDate": "2025-04-16T16:03:04.060000-07:00",
    "CustomerMasterKeySpec": "SYMMETRIC_DEFAULT",
    "Description": "",
    "Enabled": true,
    "KeyId": "1234abcd-12ab-34cd-56ef-1234567890ab",
    "KeyManager": "CUSTOMER",
    "KeySpec": "SYMMETRIC_DEFAULT",
    "KeyState": "Enabled",
    "KeyUsage": "ENCRYPT_DECRYPT",
    "MultiRegion": false,
    "Origin": "AWS_KMS",
    "EncryptionAlgorithms": [
        "SYMMETRIC_DEFAULT"
```

```
    ],
    "CurrentKeyMaterialId":
 "0b7fd7ddbac6eef27907413567cad8c810e2883dc8a7534067a82ee1142fc1e6"
  }
}
```

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface V2
- AWS SDK for .NET V4
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- AWS SDK for PHP V3
- AWS SDK for Python (Boto3)
- AWS SDK for Ruby V3

DisableKey

Sets the state of a KMS key to disabled. This change temporarily prevents use of the KMS key for cryptographic operations.

The KMS key that you use for this operation must be in a compatible key state. For more information about how key state affects the use of a KMS key, see Key states of AWS KMS keys in the AWS Key Management Service Developer Guide .

Cross-account use: No. You cannot perform this operation on a KMS key in a different AWS account.

Required permissions: kms:DisableKey (key policy)

Related operations: EnableKey

Eventual consistency: The AWS KMS API follows an eventual consistency model. For more information, see AWS KMS eventual consistency.

Request Syntax

```
{
   "KeyId": "string"
}
```

Request Parameters

For information about the parameters that are common to all actions, see Common Parameters.

The request accepts the following data in JSON format.

> **Note**
>
> In the following list, the required parameters are described first.

KeyId

Identifies the KMS key to disable.

Specify the key ID or key ARN of the KMS key.

For example:

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

To get the key ID and key ARN for a KMS key, use ListKeys or DescribeKey.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 2048.

Required: Yes

Response Elements

If the action is successful, the service sends back an HTTP 200 response with an empty HTTP body.

Errors

For information about the errors that are common to all actions, see Common Error Types.

DependencyTimeoutException

The system timed out while trying to fulfill the request. You can retry the request.

HTTP Status Code: 500

InvalidArnException

The request was rejected because a specified ARN, or an ARN in a key policy, is not valid.

HTTP Status Code: 400

KMSInternalException

The request was rejected because an internal exception occurred. The request can be retried.

HTTP Status Code: 500

KMSInvalidStateException

The request was rejected because the state of the specified resource is not valid for this request.

This exceptions means one of the following:

- The key state of the KMS key is not compatible with the operation.

  To find the key state, use the DescribeKey operation. For more information about which key states are compatible with each AWS KMS operation, see Key states of AWS KMS keys in the AWS Key Management Service Developer Guide .
- For cryptographic operations on KMS keys in custom key stores, this exception represents a general failure with many possible causes. To identify the cause, see the error message that accompanies the exception.

HTTP Status Code: 400

NotFoundException

The request was rejected because the specified entity or resource could not be found.

HTTP Status Code: 400

Examples

Example Request

The following example is formatted for legibility.

```
POST / HTTP/1.1
Host: kms.us-east-2.amazonaws.com
Content-Length: 48
X-Amz-Target: TrentService.DisableKey
X-Amz-Date: 20161107T221459Z
Content-Type: application/x-amz-json-1.1
Authorization: AWS4-HMAC-SHA256\
 Credential=AKIAI44QH8DHBEXAMPLE/20161107/us-east-2/kms/aws4_request,\
 SignedHeaders=content-type;host;x-amz-date;x-amz-target,\
 Signature=de4ddbea732953d60c07d835a5dde9037c484ee3bec9313cbecd1d9420b41a7a

{"KeyId": "1234abcd-12ab-34cd-56ef-1234567890ab"}
```

Example Response

This example illustrates one usage of DisableKey.

```
HTTP/1.1 200 OK
```

```
Server: Server
Date: Mon, 07 Nov 2016 22:14:59 GMT
Content-Type: application/x-amz-json-1.1
Content-Length: 0
Connection: keep-alive
x-amzn-RequestId: 9f5f3560-a537-11e6-8185-8df6f2682323
```

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface V2
- AWS SDK for .NET V4
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- AWS SDK for PHP V3
- AWS SDK for Python (Boto3)
- AWS SDK for Ruby V3

DisableKeyRotation

Disables automatic rotation of the key material of the specified symmetric encryption KMS key.

Automatic key rotation is supported only on symmetric encryption KMS keys. You cannot enable automatic rotation of asymmetric KMS keys, HMAC KMS keys, KMS keys with imported key material, or KMS keys in a custom key store. To enable or disable automatic rotation of a set of related multi-Region keys, set the property on the primary key.

You can enable (EnableKeyRotation) and disable automatic rotation of the key material in customer managed KMS keys. Key material rotation of AWS managed KMS keys is not configurable. AWS KMS always rotates the key material for every year. Rotation of AWS owned KMS keys varies.

> **Note**
>
> In May 2022, AWS KMS changed the rotation schedule for AWS managed keys from every three years to every year. For details, see EnableKeyRotation.

The KMS key that you use for this operation must be in a compatible key state. For details, see Key states of AWS KMS keys in the *AWS Key Management Service Developer Guide*.

Cross-account use: No. You cannot perform this operation on a KMS key in a different AWS account.

Required permissions: kms:DisableKeyRotation (key policy)

Related operations:

- EnableKeyRotation
- GetKeyRotationStatus
- ListKeyRotations
- RotateKeyOnDemand

Eventual consistency: The AWS KMS API follows an eventual consistency model. For more information, see AWS KMS eventual consistency.

Request Syntax

```
{
   "KeyId": "string"
}
```

Request Parameters

For information about the parameters that are common to all actions, see Common Parameters.

The request accepts the following data in JSON format.

> **Note**
>
> In the following list, the required parameters are described first.

KeyId

Identifies a symmetric encryption KMS key. You cannot enable or disable automatic rotation of asymmetric KMS keys, HMAC KMS keys, KMS keys with imported key material, or KMS keys in a custom key store.

Specify the key ID or key ARN of the KMS key.

For example:

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

To get the key ID and key ARN for a KMS key, use ListKeys or DescribeKey.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 2048.

Required: Yes

Response Elements

If the action is successful, the service sends back an HTTP 200 response with an empty HTTP body.

Errors

For information about the errors that are common to all actions, see Common Error Types.

DependencyTimeoutException

The system timed out while trying to fulfill the request. You can retry the request.

HTTP Status Code: 500

DisabledException

The request was rejected because the specified KMS key is not enabled.

HTTP Status Code: 400

InvalidArnException

The request was rejected because a specified ARN, or an ARN in a key policy, is not valid.

HTTP Status Code: 400

KMSInternalException

The request was rejected because an internal exception occurred. The request can be retried.

HTTP Status Code: 500

KMSInvalidStateException

The request was rejected because the state of the specified resource is not valid for this request.

This exceptions means one of the following:

- The key state of the KMS key is not compatible with the operation.

  To find the key state, use the DescribeKey operation. For more information about which key states are compatible with each AWS KMS operation, see Key states of AWS KMS keys in the AWS Key Management Service Developer Guide .
- For cryptographic operations on KMS keys in custom key stores, this exception represents a general failure with many possible causes. To identify the cause, see the error message that accompanies the exception.

HTTP Status Code: 400

NotFoundException

The request was rejected because the specified entity or resource could not be found.

HTTP Status Code: 400

UnsupportedOperationException

The request was rejected because a specified parameter is not supported or a specified resource is not valid for this operation.

HTTP Status Code: 400

Examples

Example Request

The following example is formatted for legibility.

```
POST / HTTP/1.1
Host: kms.us-east-2.amazonaws.com
Content-Length: 48
X-Amz-Target: TrentService.DisableKeyRotation
X-Amz-Date: 20161107T222236Z
Content-Type: application/x-amz-json-1.1
Authorization: AWS4-HMAC-SHA256\
 Credential=AKIAI44QH8DHBEXAMPLE/20161107/us-east-2/kms/aws4_request,\
 SignedHeaders=content-type;host;x-amz-date;x-amz-target,\
 Signature=2304622be05af2afa8c75bf784fb87b280c194746418b05d7af947c8c2bd8f04

{"KeyId": "1234abcd-12ab-34cd-56ef-1234567890ab"}
```

Example Response

This example illustrates one usage of DisableKeyRotation.

```
HTTP/1.1 200 OK
Server: Server
Date: Mon, 07 Nov 2016 22:22:36 GMT
Content-Type: application/x-amz-json-1.1
Content-Length: 0
Connection: keep-alive
```

```
x-amzn-RequestId: afd1c328-a538-11e6-861b-ad130425efbf
```

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface V2
- AWS SDK for .NET V4
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- AWS SDK for PHP V3
- AWS SDK for Python (Boto3)
- AWS SDK for Ruby V3

DisconnectCustomKeyStore

Disconnects the custom key store from its backing key store. This operation disconnects an AWS CloudHSM key store from its associated AWS CloudHSM cluster or disconnects an external key store from the external key store proxy that communicates with your external key manager.

This operation is part of the custom key stores feature in AWS KMS, which combines the convenience and extensive integration of AWS KMS with the isolation and control of a key store that you own and manage.

While a custom key store is disconnected, you can manage the custom key store and its KMS keys, but you cannot create or use its KMS keys. You can reconnect the custom key store at any time.

> **Note**
>
> While a custom key store is disconnected, all attempts to create KMS keys in the custom key store or to use existing KMS keys in cryptographic operations will fail. This action can prevent users from storing and accessing sensitive data.

When you disconnect a custom key store, its `ConnectionState` changes to `Disconnected`. To find the connection state of a custom key store, use the DescribeCustomKeyStores operation. To reconnect a custom key store, use the ConnectCustomKeyStore operation.

If the operation succeeds, it returns a JSON object with no properties.

Cross-account use: No. You cannot perform this operation on a custom key store in a different AWS account.

Required permissions: kms:DisconnectCustomKeyStore (IAM policy)

Related operations:

- ConnectCustomKeyStore
- CreateCustomKeyStore
- DeleteCustomKeyStore
- DescribeCustomKeyStores
- UpdateCustomKeyStore

Eventual consistency: The AWS KMS API follows an eventual consistency model. For more information, see AWS KMS eventual consistency.

Request Syntax

```
{
   "CustomKeyStoreId": "string"
}
```

Request Parameters

For information about the parameters that are common to all actions, see Common Parameters.

The request accepts the following data in JSON format.

> **Note**
>
> In the following list, the required parameters are described first.

CustomKeyStoreId

Enter the ID of the custom key store you want to disconnect. To find the ID of a custom key store, use the DescribeCustomKeyStores operation.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 64.

Required: Yes

Response Elements

If the action is successful, the service sends back an HTTP 200 response with an empty HTTP body.

Errors

For information about the errors that are common to all actions, see Common Error Types.

CustomKeyStoreInvalidStateException

The request was rejected because of the `ConnectionState` of the custom key store. To get the `ConnectionState` of a custom key store, use the DescribeCustomKeyStores operation.

This exception is thrown under the following conditions:

- You requested the ConnectCustomKeyStore operation on a custom key store with a `ConnectionState` of `DISCONNECTING` or `FAILED`. This operation is valid for all other `ConnectionState` values. To reconnect a custom key store in a `FAILED` state, disconnect it (DisconnectCustomKeyStore), then connect it (`ConnectCustomKeyStore`).
- You requested the CreateKey operation in a custom key store that is not connected. This operations is valid only when the custom key store `ConnectionState` is `CONNECTED`.
- You requested the DisconnectCustomKeyStore operation on a custom key store with a `ConnectionState` of `DISCONNECTING` or `DISCONNECTED`. This operation is valid for all other `ConnectionState` values.
- You requested the UpdateCustomKeyStore or DeleteCustomKeyStore operation on a custom key store that is not disconnected. `UpdateCustomKeyStore` can be called on a custom key store in the `CONNECTED` state only to update `NewCustomKeyStoreName`. For all other properties, the custom key store `ConnectionState` must be `DISCONNECTED`.
- You requested the GenerateRandom operation in an AWS CloudHSM key store that is not connected. This operation is valid only when the AWS CloudHSM key store `ConnectionState` is `CONNECTED`.

HTTP Status Code: 400

CustomKeyStoreNotFoundException

The request was rejected because AWS KMS cannot find a custom key store with the specified key store name or ID.

HTTP Status Code: 400

KMSInternalException

The request was rejected because an internal exception occurred. The request can be retried.

HTTP Status Code: 500

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface V2
- AWS SDK for .NET V4
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- AWS SDK for PHP V3
- AWS SDK for Python (Boto3)
- AWS SDK for Ruby V3

EnableKey

Sets the key state of a KMS key to enabled. This allows you to use the KMS key for cryptographic operations.

The KMS key that you use for this operation must be in a compatible key state. For details, see Key states of AWS KMS keys in the *AWS Key Management Service Developer Guide*.

Cross-account use: No. You cannot perform this operation on a KMS key in a different AWS account.

Required permissions: kms:EnableKey (key policy)

Related operations: DisableKey

Eventual consistency: The AWS KMS API follows an eventual consistency model. For more information, see AWS KMS eventual consistency.

Request Syntax

```
{
   "KeyId": "string"
}
```

Request Parameters

For information about the parameters that are common to all actions, see Common Parameters.

The request accepts the following data in JSON format.

> **Note**
>
> In the following list, the required parameters are described first.

KeyId

Identifies the KMS key to enable.

Specify the key ID or key ARN of the KMS key.

For example:

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

To get the key ID and key ARN for a KMS key, use ListKeys or DescribeKey.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 2048.

Required: Yes

Response Elements

If the action is successful, the service sends back an HTTP 200 response with an empty HTTP body.

Errors

For information about the errors that are common to all actions, see Common Error Types.

DependencyTimeoutException

The system timed out while trying to fulfill the request. You can retry the request.

HTTP Status Code: 500

InvalidArnException

The request was rejected because a specified ARN, or an ARN in a key policy, is not valid.

HTTP Status Code: 400

KMSInternalException

The request was rejected because an internal exception occurred. The request can be retried.

HTTP Status Code: 500

KMSInvalidStateException

The request was rejected because the state of the specified resource is not valid for this request.

This exceptions means one of the following:

- The key state of the KMS key is not compatible with the operation.

  To find the key state, use the DescribeKey operation. For more information about which key states are compatible with each AWS KMS operation, see Key states of AWS KMS keys in the AWS Key Management Service Developer Guide .
- For cryptographic operations on KMS keys in custom key stores, this exception represents a general failure with many possible causes. To identify the cause, see the error message that accompanies the exception.

HTTP Status Code: 400

LimitExceededException

The request was rejected because a length constraint or quota was exceeded. For more information, see Quotas in the *AWS Key Management Service Developer Guide*.

HTTP Status Code: 400

NotFoundException

The request was rejected because the specified entity or resource could not be found.

HTTP Status Code: 400

Examples

Example Request

The following example is formatted for legibility.

```
POST / HTTP/1.1
Host: kms.us-east-2.amazonaws.com
Content-Length: 48
X-Amz-Target: TrentService.EnableKey
X-Amz-Date: 20161107T221800Z
Content-Type: application/x-amz-json-1.1
Authorization: AWS4-HMAC-SHA256\
 Credential=AKIAI44QH8DHBEXAMPLE/20161107/us-east-2/kms/aws4_request,\
 SignedHeaders=content-type;host;x-amz-date;x-amz-target,\
 Signature=74d02e36580c1759255dfef66f1e51f3542e469de8c7c8fa5fb21c042e518295
```

```
{"KeyId": "1234abcd-12ab-34cd-56ef-1234567890ab"}
```

Example Response

This example illustrates one usage of EnableKey.

```
HTTP/1.1 200 OK
Server: Server
Date: Mon, 07 Nov 2016 22:18:00 GMT
Content-Type: application/x-amz-json-1.1
Content-Length: 0
Connection: keep-alive
x-amzn-RequestId: 0b588162-a538-11e6-b4ed-059c103e7a90
```

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface V2
- AWS SDK for .NET V4
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- AWS SDK for PHP V3
- AWS SDK for Python (Boto3)
- AWS SDK for Ruby V3

EnableKeyRotation

Enables automatic rotation of the key material of the specified symmetric encryption KMS key.

By default, when you enable automatic rotation of a customer managed KMS key, AWS KMS rotates the key material of the KMS key one year (approximately 365 days) from the enable date and every year thereafter. You can use the optional `RotationPeriodInDays` parameter to specify a custom rotation period when you enable key rotation, or you can use `RotationPeriodInDays` to modify the rotation period of a key that you previously enabled automatic key rotation on.

You can monitor rotation of the key material for your KMS keys in AWS CloudTrail and Amazon CloudWatch. To disable rotation of the key material in a customer managed KMS key, use the DisableKeyRotation operation. You can use the GetKeyRotationStatus operation to identify any in progress rotations. You can use the ListKeyRotations operation to view the details of completed rotations.

Automatic key rotation is supported only on symmetric encryption KMS keys. You cannot enable automatic rotation of asymmetric KMS keys, HMAC KMS keys, KMS keys with imported key material, or KMS keys in a custom key store. To enable or disable automatic rotation of a set of related multi-Region keys, set the property on the primary key.

You cannot enable or disable automatic rotation of AWS managed KMS keys. AWS KMS always rotates the key material of AWS managed keys every year. Rotation of AWS owned KMS keys is managed by the AWS service that owns the key.

> **Note**
>
> In May 2022, AWS KMS changed the rotation schedule for AWS managed keys from every three years (approximately 1,095 days) to every year (approximately 365 days).
> New AWS managed keys are automatically rotated one year after they are created, and approximately every year thereafter.
> Existing AWS managed keys are automatically rotated one year after their most recent rotation, and every year thereafter.

The KMS key that you use for this operation must be in a compatible key state. For details, see Key states of AWS KMS keys in the *AWS Key Management Service Developer Guide*.

Cross-account use: No. You cannot perform this operation on a KMS key in a different AWS account.

Required permissions: kms:EnableKeyRotation (key policy)

Related operations:

- DisableKeyRotation
- GetKeyRotationStatus
- ListKeyRotations
- RotateKeyOnDemand

> **Note**
>
> You can perform on-demand (RotateKeyOnDemand) rotation of the key material in customer managed KMS keys, regardless of whether or not automatic key rotation is enabled.

Eventual consistency: The AWS KMS API follows an eventual consistency model. For more information, see AWS KMS eventual consistency.

Request Syntax

```
{
   "KeyId": "string",
   "RotationPeriodInDays": number
}
```

Request Parameters

For information about the parameters that are common to all actions, see Common Parameters.

The request accepts the following data in JSON format.

> **Note**
>
> In the following list, the required parameters are described first.

KeyId

Identifies a symmetric encryption KMS key. You cannot enable automatic rotation of asymmetric KMS keys, HMAC KMS keys, KMS keys with imported key material, or KMS keys in a custom key store. To enable or disable automatic rotation of a set of related multi-Region keys, set the property on the primary key.

Specify the key ID or key ARN of the KMS key.

For example:

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

To get the key ID and key ARN for a KMS key, use ListKeys or DescribeKey.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 2048.

Required: Yes

RotationPeriodInDays

Use this parameter to specify a custom period of time between each rotation date. If no value is specified, the default value is 365 days.

The rotation period defines the number of days after you enable automatic key rotation that AWS KMS will rotate your key material, and the number of days between each automatic rotation thereafter.

You can use the `kms:RotationPeriodInDays` condition key to further constrain the values that principals can specify in the `RotationPeriodInDays` parameter.

Type: Integer

Valid Range: Minimum value of 90. Maximum value of 2560.

Required: No

Response Elements

If the action is successful, the service sends back an HTTP 200 response with an empty HTTP body.

Errors

For information about the errors that are common to all actions, see Common Error Types.

DependencyTimeoutException

The system timed out while trying to fulfill the request. You can retry the request.

HTTP Status Code: 500

DisabledException

The request was rejected because the specified KMS key is not enabled.

HTTP Status Code: 400

InvalidArnException

The request was rejected because a specified ARN, or an ARN in a key policy, is not valid.

HTTP Status Code: 400

KMSInternalException

The request was rejected because an internal exception occurred. The request can be retried.

HTTP Status Code: 500

KMSInvalidStateException

The request was rejected because the state of the specified resource is not valid for this request.

This exceptions means one of the following:

- The key state of the KMS key is not compatible with the operation.

  To find the key state, use the DescribeKey operation. For more information about which key states are compatible with each AWS KMS operation, see Key states of AWS KMS keys in the AWS Key Management Service Developer Guide .
- For cryptographic operations on KMS keys in custom key stores, this exception represents a general failure with many possible causes. To identify the cause, see the error message that accompanies the exception.

HTTP Status Code: 400

NotFoundException

The request was rejected because the specified entity or resource could not be found.

HTTP Status Code: 400

UnsupportedOperationException

The request was rejected because a specified parameter is not supported or a specified resource is not valid for this operation.

HTTP Status Code: 400

Examples

Example Request

The following example is formatted for legibility.

```
POST / HTTP/1.1
Host: kms.us-east-2.amazonaws.com
Content-Length: 48
X-Amz-Target: TrentService.EnableKeyRotation
X-Amz-Date: 20161107T221835Z
Content-Type: application/x-amz-json-1.1
Authorization: AWS4-HMAC-SHA256\
 Credential=AKIAI44QH8DHBEXAMPLE/20161107/us-east-2/kms/aws4_request,\
 SignedHeaders=content-type;host;x-amz-date;x-amz-target,\
 Signature=4783e177036ca78627fe0cda9dcfdaf4ad7c8312d0e7c3d71d814b0c4cff1c0b

{
    "KeyId": "1234abcd-12ab-34cd-56ef-1234567890ab",
    "RotationPeriodInDays": 180
}
```

Example Response

This example illustrates one usage of EnableKeyRotation.

```
HTTP/1.1 200 OK
```

```
Server: Server
Date: Mon, 07 Nov 2016 22:18:36 GMT
Content-Type: application/x-amz-json-1.1
Content-Length: 0
Connection: keep-alive
x-amzn-RequestId: 2077c3bf-a538-11e6-b6fb-794e83344f84
```

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface V2
- AWS SDK for .NET V4
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- AWS SDK for PHP V3
- AWS SDK for Python (Boto3)
- AWS SDK for Ruby V3

Encapsulate

Generates a shared secret and its encapsulated (ciphertext) form using the public key of an ML-KEM KMS key pair. Module-Lattice-Based Key-Encapsulation Mechanism (ML-KEM), specified in FIPS 203 on the NIST website, is a post-quantum algorithm for establishing a shared secret between two parties.

For an example of how two parties establish a shared secret over an insecure channel using ML-KEM, see ML-KEM keys in AWS KMS in the *AWS Key Management Service Developer Guide*.

The Encapsulate operation is not available in China Regions.

The KMS key that you use for this operation must be in a compatible key state. For details, see Key states of AWS KMS keys in the *AWS Key Management Service Developer Guide*.

Cross-account use: Yes. To perform this operation with a KMS key in a different AWS account, specify the key ARN or alias ARN in the value of the `KeyId` parameter.

Required permissions: kms:Encapsulate (key policy)

Related operations:

- Decapsulate
- DeriveSharedSecret
- CreateKey
- GetPublicKey
- DescribeKey

Eventual consistency: The AWS KMS API follows an eventual consistency model. For more information, see AWS KMS eventual consistency.

Request Syntax

```
{
   "DryRun": boolean,
   "EncapsulationAlgorithm": "string",
   "GrantTokens": [ "string" ],
   "KeyId": "string"
```

```
}
```

Request Parameters

For information about the parameters that are common to all actions, see Common Parameters.

The request accepts the following data in JSON format.

> **Note**
>
> In the following list, the required parameters are described first.

EncapsulationAlgorithm

Specifies the key encapsulation algorithm used to generate the shared secret and its encapsulated form. The only valid value is `ML_KEM`.

Type: String

Valid Values: `ML_KEM`

Required: Yes

KeyId

Identifies an asymmetric ML-KEM KMS key. AWS KMS uses the public key in the specified key pair to encapsulate the shared secret. ML-KEM KMS keys have a `KeySpec` value of `ML_KEM_512`, `ML_KEM_768`, or `ML_KEM_1024` and a `KeyUsage` value of `ENCAPSULATE_DECAPSULATE`. To find the `KeySpec` and `KeyUsage` of a KMS key, use the DescribeKey operation.

To specify a KMS key, use its key ID, key ARN, alias name, or alias ARN. When using an alias name, prefix it with `"alias/"`. To specify a KMS key in a different AWS account, you must use the key ARN or alias ARN.

For example:

- Key ID: `1234abcd-12ab-34cd-56ef-1234567890ab`
- Key ARN: `arn:aws:kms:us-east-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab`
- Alias name: `alias/ExampleAlias`

- Alias ARN: `arn:aws:kms:us-east-2:111122223333:alias/ExampleAlias`

To get the key ID and key ARN for a KMS key, use ListKeys or DescribeKey. To get the alias name and alias ARN, use ListAliases.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 2048.

Required: Yes

DryRun

Checks if your request will succeed. `DryRun` is an optional parameter.

To learn more about how to use this parameter, see Testing your permissions in the *AWS Key Management Service Developer Guide*.

Type: Boolean

Required: No

GrantTokens

A list of grant tokens.

Use a grant token when your permission to call this operation comes from a new grant that has not yet achieved *eventual consistency*. For more information, see Grant token and Using a grant token in the *AWS Key Management Service Developer Guide*.

Type: Array of strings

Array Members: Minimum number of 0 items. Maximum number of 10 items.

Length Constraints: Minimum length of 1. Maximum length of 8192.

Required: No

Response Syntax

```
{
   "EncapsulationAlgorithm": "string",
   "KeyId": "string",
```

```
    "SharedSecret": blob,
    "SharedSecretCiphertextBlob": blob
}
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

EncapsulationAlgorithm

The key encapsulation algorithm used to generate the shared secret.

Type: String

Valid Values: `ML_KEM`

KeyId

The identifier of the ML-KEM KMS key used to encapsulate the shared secret.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 2048.

SharedSecret

The plaintext shared secret. It is a 32-byte cryptographically random value that both parties can use directly as key material or pass through a key derivation function to generate multiple keys and nonces. When you use the HTTP API or the AWS CLI, the value is Base64-encoded. Otherwise, it is not Base64-encoded.

Type: Base64-encoded binary data object

Length Constraints: Minimum length of 1. Maximum length of 4096.

SharedSecretCiphertextBlob

The encapsulated form of the shared secret. Pass this value to the Decapsulate operation to recover the shared secret. When you use the HTTP API or the AWS CLI, the value is Base64-encoded. Otherwise, it is not Base64-encoded. The length of `SharedSecretCiphertextBlob` depends on the ML-KEM key spec. Before Base64 encoding, the encapsulated shared secret is 768 bytes for `ML_KEM_512`, 1088 bytes for `ML_KEM_768`, and 1568 bytes for `ML_KEM_1024`.

Type: Base64-encoded binary data object

Length Constraints: Minimum length of 1. Maximum length of 6144.

Errors

For information about the errors that are common to all actions, see Common Error Types.

DependencyTimeoutException

The system timed out while trying to fulfill the request. You can retry the request.

HTTP Status Code: 500

DisabledException

The request was rejected because the specified KMS key is not enabled.

HTTP Status Code: 400

DryRunOperationException

The request was rejected because the DryRun parameter was specified.

HTTP Status Code: 400

InvalidGrantTokenException

The request was rejected because the specified grant token is not valid.

HTTP Status Code: 400

InvalidKeyUsageException

The request was rejected for one of the following reasons:

- The `KeyUsage` value of the KMS key is incompatible with the API operation.
- The encryption algorithm or signing algorithm specified for the operation is incompatible with the type of key material in the KMS key (`KeySpec`).

For encrypting, decrypting, re-encrypting, and generating data keys, the `KeyUsage` must be `ENCRYPT_DECRYPT`. For signing and verifying messages, the `KeyUsage` must be `SIGN_VERIFY`. For generating and verifying message authentication codes (MACs), the `KeyUsage` must be `GENERATE_VERIFY_MAC`. For deriving key agreement secrets, the `KeyUsage` must be

KEY_AGREEMENT. For encapsulating and decapsulating shared secrets, the KeyUsage must be ENCAPSULATE_DECAPSULATE. To find the KeyUsage of a KMS key, use the DescribeKey operation.

To find the encryption or signing algorithms supported for a particular KMS key, use the DescribeKey operation.

HTTP Status Code: 400

KeyUnavailableException

The request was rejected because the specified KMS key was not available. You can retry the request.

HTTP Status Code: 500

KMSInternalException

The request was rejected because an internal exception occurred. The request can be retried.

HTTP Status Code: 500

KMSInvalidStateException

The request was rejected because the state of the specified resource is not valid for this request.

This exceptions means one of the following:

- The key state of the KMS key is not compatible with the operation.

  To find the key state, use the DescribeKey operation. For more information about which key states are compatible with each AWS KMS operation, see Key states of AWS KMS keys in the AWS Key Management Service Developer Guide .
- For cryptographic operations on KMS keys in custom key stores, this exception represents a general failure with many possible causes. To identify the cause, see the error message that accompanies the exception.

HTTP Status Code: 400

NotFoundException

The request was rejected because the specified entity or resource could not be found.

HTTP Status Code: 400

Examples

Example Request

The following example is formatted for legibility.

```
POST / HTTP/1.1
Host: kms.us-east-2.amazonaws.com
Content-Length: 85
X-Amz-Target: TrentService.Encapsulate
X-Amz-Date: 20261018T233819Z
Content-Type: application/x-amz-json-1.1
Authorization: AWS4-HMAC-SHA256\
 Credential=AKIAI44QH8DHBEXAMPLE/20261018/us-east-2/kms/aws4_request,\
 SignedHeaders=content-type;host;x-amz-date;x-amz-target,\
 Signature=2cea34fe55d5858295a377448a1e053d0edd45ce571da7cf69b202905759f272

{
    "KeyId": "1234abcd-12ab-34cd-56ef-1234567890ab",
    "EncapsulationAlgorithm": "ML_KEM"
}
```

Example Response

This example illustrates one usage of Encapsulate.

```
HTTP/1.1 200 OK
Server: Server
Date: Sun, 18 Oct 2026 23:38:19 GMT
Content-Type: application/x-amz-json-1.1
Content-Length: 1635
Connection: keep-alive
x-amzn-RequestId: 1e76aa81-a532-11e6-a265-d3aef78e1a90

{
    "KeyId": "1234abcd-12ab-34cd-56ef-1234567890ab",
    "SharedSecret": "oJ2H4lIMwHD2VgqkBf63bGY86fwcyxe2DOu/+EGN5bU=",
    "SharedSecretCiphertextBlob": "kGuVROOrf4ZLP3rR7DB64O803VeoevUmGCDvJ/
vc2vBEYkYwZoc2NmNYeX0g55i/
Pz4uvqnwhVFC2KDZJD5Rdob4GoaRphStzM87F30tq7Xui3zfHO8X5aZCXJpPB3lh/DTiRoitmjo
+mwleYyL6IOG8PurEPASSROB3SdfhmVPsTY6ZaUARqtY6BFBIyHbnWzI0a2HLdl1DGnMuvy3yTdhXzE5/
Z7ZUAe5edL+WLao7vWWl2InU1TDlPE+kDu540aCoIX37ke4dmTJIe/
```

```
+e20nzINEmuhCzUVIoRGrYyT8ZnuyMq9kZtd
+SmhfKRA5PQT0vrp0PSHE2653724YPKvLSuEoNbMSXPTbB8rBeP3kXg0p+4ZLjRda3SoMEvqbQdWLVRclMI
+PtXUSCJkKWrbmg9/2eyO9fPNCsck36zVEpZaw4irP1bZguM+7qAYDhxPupO+Ks7/
vkgDh6myzjslSBHCGWqnMLmoiatJinsI2NdFLprGvqZqzJrbIABI54OfFs6B7pSqLdNATbXuRRMUMRX/2+c2WcL6YR62tQU
lgbmiyeZfhsvpUPPAX8QDfkZUPG1xeTn2ao7it+jO48OOPFbzj
+zwt2ZCiIpxN1lR41R0ubD18WQynI9jIz5R3nY6RTtx8kJ4Y
+br62NdpsczIr6tWDgpGfqY6XGyW5SncB2EJ85AIrQrEUFbPva2inHPgxdkCOEgLp8VYwabNXiLJf7D8BiU90QyPUyFdoZW
+8EbfiSLwoDqDbgGgbAbItCrrdwFA6vUBXG2AQfvsdq+9L3OcUmPajCwnTnf/BTMR4mnHT
+0Eu5gamKmb86SjCa9uRzIosVGTvaMLuQBMD
+507rKJwr5+Ia5k9TMjtfNO4oQglNKDVORumiyxOvTDQDYbWcT9B4g9ygOA0GZts7qmJzM44wGtXpfrQpZCWX8X0zKE9Les
+W7h/LqG0qzisKX97vLXIdGUSC14On2iPQWgO3qyXfkt8xwXohc/
veWoRsS9MFUtLHabYJ25o8VZdtwWJfVcDSrCaSFDHJeAFFihA/
F3FlVZuMoWy2DwadoL4xMG2ZmGgs68rPIeLq27dXZFaF5PemCsrQKMhlo4X1tGQROgehoj0Kr8Dab17DgBZJrzvvrecWzoT
WJqGDxkLVlwP9z6fIbEy4vbhe2vhbVtetjlbh0F0JQFnMMet29j6bs
+506K88MPYyHRZBC5HIODmrYDlMjUdPaUGOUYM5VBLqEwnUvCY82SdFMKeuUxVSgQ2dyF2W41Sybf4pzRWfLMzkWuExbIbt
+eXb4nhIGJVGQBXSWefN61Q=",
    "EncapsulationAlgorithm": "ML_KEM"
}
```


See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface V2
- AWS SDK for .NET V4
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- AWS SDK for PHP V3
- AWS SDK for Python (Boto3)
- AWS SDK for Ruby V3

Encrypt

Encrypts plaintext of up to 4,096 bytes using a KMS key. You can use a symmetric or asymmetric KMS key with a `KeyUsage` of `ENCRYPT_DECRYPT`.

You can use this operation to encrypt small amounts of arbitrary data, such as a personal identifier or database password, or other sensitive information. You don't need to use the `Encrypt` operation to encrypt a data key. The GenerateDataKey and GenerateDataKeyPair operations return a plaintext data key and an encrypted copy of that data key.

If you use a symmetric encryption KMS key, you can use an encryption context to add additional security to your encryption operation. If you specify an `EncryptionContext` when encrypting data, you must specify the same encryption context (a case-sensitive exact match) when decrypting the data. Otherwise, the request to decrypt fails with an `InvalidCiphertextException`. For more information, see Encryption Context in the *AWS Key Management Service Developer Guide*.

If you specify an asymmetric KMS key, you must also specify the encryption algorithm. The algorithm must be compatible with the KMS key spec.

> **Important**
>
> When you use an asymmetric KMS key to encrypt or reencrypt data, be sure to record the KMS key and encryption algorithm that you choose. You will be required to provide the same KMS key and encryption algorithm when you decrypt the data. If the KMS key and algorithm do not match the values used to encrypt the data, the decrypt operation fails. You are not required to supply the key ID and encryption algorithm when you decrypt with symmetric encryption KMS keys because AWS KMS stores this information in the ciphertext blob. AWS KMS cannot store metadata in ciphertext generated with asymmetric keys. The standard format for asymmetric key ciphertext does not include configurable fields.

The maximum size of the data that you can encrypt varies with the type of KMS key and the encryption algorithm that you choose.

- Symmetric encryption KMS keys
  - `SYMMETRIC_DEFAULT`: 4096 bytes
- `RSA_2048`
  - `RSAES_OAEP_SHA_1`: 214 bytes

  - RSAES_OAEP_SHA_256: 190 bytes
- RSA_3072
  - RSAES_OAEP_SHA_1: 342 bytes
  - RSAES_OAEP_SHA_256: 318 bytes
- RSA_4096
  - RSAES_OAEP_SHA_1: 470 bytes
  - RSAES_OAEP_SHA_256: 446 bytes
- SM2PKE: 1024 bytes (China Regions only)

The KMS key that you use for this operation must be in a compatible key state. For details, see Key states of AWS KMS keys in the *AWS Key Management Service Developer Guide*.

Cross-account use: Yes. To perform this operation with a KMS key in a different AWS account, specify the key ARN or alias ARN in the value of the `KeyId` parameter.

Required permissions: kms:Encrypt (key policy)

Related operations:

- Decrypt
- GenerateDataKey
- GenerateDataKeyPair

Eventual consistency: The AWS KMS API follows an eventual consistency model. For more information, see AWS KMS eventual consistency.

Request Syntax

```
{
   "DryRun": boolean,
   "EncryptionAlgorithm": "string",
   "EncryptionContext": {
      "string" : "string"
   },
   "GrantTokens": [ "string" ],
   "KeyId": "string",
```

```
    "Plaintext": blob
}
```

Request Parameters

For information about the parameters that are common to all actions, see Common Parameters.

The request accepts the following data in JSON format.

> **Note**
>
> In the following list, the required parameters are described first.

KeyId

Identifies the KMS key to use in the encryption operation. The KMS key must have a KeyUsage of ENCRYPT_DECRYPT. To find the KeyUsage of a KMS key, use the DescribeKey operation.

To specify a KMS key, use its key ID, key ARN, alias name, or alias ARN. When using an alias name, prefix it with "alias/". To specify a KMS key in a different AWS account, you must use the key ARN or alias ARN.

For example:

- Key ID: 1234abcd-12ab-34cd-56ef-1234567890ab
- Key ARN: arn:aws:kms:us-east-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab
- Alias name: alias/ExampleAlias
- Alias ARN: arn:aws:kms:us-east-2:111122223333:alias/ExampleAlias

To get the key ID and key ARN for a KMS key, use ListKeys or DescribeKey. To get the alias name and alias ARN, use ListAliases.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 2048.

Required: Yes

Plaintext

Data to be encrypted.

Type: Base64-encoded binary data object

Length Constraints: Minimum length of 1. Maximum length of 4096.

Required: Yes

DryRun

Checks if your request will succeed. `DryRun` is an optional parameter.

To learn more about how to use this parameter, see Testing your permissions in the *AWS Key Management Service Developer Guide*.

Type: Boolean

Required: No

EncryptionAlgorithm

Specifies the encryption algorithm that AWS KMS will use to encrypt the plaintext message. The algorithm must be compatible with the KMS key that you specify.

This parameter is required only for asymmetric KMS keys. The default value, `SYMMETRIC_DEFAULT`, is the algorithm used for symmetric encryption KMS keys. If you are using an asymmetric KMS key, we recommend RSAES_OAEP_SHA_256.

The SM2PKE algorithm is only available in China Regions.

Type: String

Valid Values: `SYMMETRIC_DEFAULT | RSAES_OAEP_SHA_1 | RSAES_OAEP_SHA_256 | SM2PKE`

Required: No

EncryptionContext

Specifies the encryption context that will be used to encrypt the data. An encryption context is valid only for cryptographic operations with a symmetric encryption KMS key. The standard asymmetric encryption algorithms and HMAC algorithms that AWS KMS uses do not support an encryption context.

> **Important**
>
> Do not include confidential or sensitive information in this field. This field may be displayed in plaintext in CloudTrail logs and other output.

An *encryption context* is a collection of non-secret key-value pairs that represent additional authenticated data. When you use an encryption context to encrypt data, you must specify the same (an exact case-sensitive match) encryption context to decrypt the data. An encryption context is supported only on operations with symmetric encryption KMS keys. On operations with symmetric encryption KMS keys, an encryption context is optional, but it is strongly recommended.

For more information, see Encryption context in the *AWS Key Management Service Developer Guide*.

Type: String to string map

Required: No

GrantTokens

A list of grant tokens.

Use a grant token when your permission to call this operation comes from a new grant that has not yet achieved *eventual consistency*. For more information, see Grant token and Using a grant token in the *AWS Key Management Service Developer Guide*.

Type: Array of strings

Array Members: Minimum number of 0 items. Maximum number of 10 items.

Length Constraints: Minimum length of 1. Maximum length of 8192.

Required: No

Response Syntax

```
{
   "CiphertextBlob": blob,
```

```
   "EncryptionAlgorithm": "string",
   "KeyId": "string"
}
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

CiphertextBlob

The encrypted plaintext. When you use the HTTP API or the AWS CLI, the value is Base64-encoded. Otherwise, it is not Base64-encoded.

Type: Base64-encoded binary data object

Length Constraints: Minimum length of 1. Maximum length of 6144.

EncryptionAlgorithm

The encryption algorithm that was used to encrypt the plaintext.

Type: String

Valid Values: `SYMMETRIC_DEFAULT | RSAES_OAEP_SHA_1 | RSAES_OAEP_SHA_256 | SM2PKE`

KeyId

The Amazon Resource Name (key ARN) of the KMS key that was used to encrypt the plaintext.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 2048.

Errors

For information about the errors that are common to all actions, see Common Error Types.

DependencyTimeoutException

The system timed out while trying to fulfill the request. You can retry the request.

HTTP Status Code: 500

DisabledException

The request was rejected because the specified KMS key is not enabled.

HTTP Status Code: 400

DryRunOperationException

The request was rejected because the DryRun parameter was specified.

HTTP Status Code: 400

InvalidGrantTokenException

The request was rejected because the specified grant token is not valid.

HTTP Status Code: 400

InvalidKeyUsageException

The request was rejected for one of the following reasons:

- The `KeyUsage` value of the KMS key is incompatible with the API operation.
- The encryption algorithm or signing algorithm specified for the operation is incompatible with the type of key material in the KMS key (`KeySpec`).

For encrypting, decrypting, re-encrypting, and generating data keys, the `KeyUsage` must be `ENCRYPT_DECRYPT`. For signing and verifying messages, the `KeyUsage` must be `SIGN_VERIFY`. For generating and verifying message authentication codes (MACs), the `KeyUsage` must be `GENERATE_VERIFY_MAC`. For deriving key agreement secrets, the `KeyUsage` must be `KEY_AGREEMENT`. For encapsulating and decapsulating shared secrets, the `KeyUsage` must be `ENCAPSULATE_DECAPSULATE`. To find the `KeyUsage` of a KMS key, use the DescribeKey operation.

To find the encryption or signing algorithms supported for a particular KMS key, use the DescribeKey operation.

HTTP Status Code: 400

KeyUnavailableException

The request was rejected because the specified KMS key was not available. You can retry the request.

HTTP Status Code: 500

KMSInternalException

The request was rejected because an internal exception occurred. The request can be retried.

HTTP Status Code: 500

KMSInvalidStateException

The request was rejected because the state of the specified resource is not valid for this request.

This exceptions means one of the following:

- The key state of the KMS key is not compatible with the operation.

  To find the key state, use the DescribeKey operation. For more information about which key states are compatible with each AWS KMS operation, see Key states of AWS KMS keys in the AWS Key Management Service Developer Guide .
- For cryptographic operations on KMS keys in custom key stores, this exception represents a general failure with many possible causes. To identify the cause, see the error message that accompanies the exception.

HTTP Status Code: 400

NotFoundException

The request was rejected because the specified entity or resource could not be found.

HTTP Status Code: 400

Examples

The following examples are formatted for legibility.

Example Request

This example illustrates one usage of Encrypt.

```
POST / HTTP/1.1
Host: kms.us-west-2.amazonaws.com
Content-Length: 107
X-Amz-Target: TrentService.Encrypt
```

```
X-Amz-Date: 20160517T203825Z
Content-Type: application/x-amz-json-1.1
Authorization: AWS4-HMAC-SHA256\
 Credential=AKIAI44QH8DHBEXAMPLE/20160517/us-west-2/kms/aws4_request,\
 SignedHeaders=content-type;host;x-amz-date;x-amz-target,\
 Signature=67ccaa73c1af7fe83973ce8139104d55f3bdcebee323d2f2e65996d99015ace2

{
  "Plaintext": "VGhpcyBpcyBEYXkgMSBmb3IgdGhlIEludGVybmV0Cg==",
  "KeyId": "1234abcd-12ab-34cd-56ef-1234567890ab"
}
```

Example Response

This example illustrates one usage of Encrypt.

```
HTTP/1.1 200 OK
Server: Server
Date: Tue, 17 May 2016 20:38:30 GMT
Content-Type: application/x-amz-json-1.1
Content-Length: 379
Connection: keep-alive
x-amzn-RequestId: 50a0c603-1c6f-11e6-bb9e-3fadde80ce75

{
  "CiphertextBlob":
 "CiDPoCH188S65r5Cy7pAhIFJMXDlU7mewhSlYUpuQIVBrhKmAQEBAgB4z6Ah9fPEuua
+Qsu6QISBSTFw5VO5nsIUpWFKbkCFQa4AAAB9MHsGCSqGSIb3DQEHBqBuMGwCAQAwZwYJKoZIhvcNAQcBMB4GCWCGSAFlAw
ZjYCARCAOt8la8qXLO5wB3JH2NlwWWzWRU2RKqpO9A/0psE5UWwkK6CnwoeC3Zj9Q0A66apZkbRglFfY1lTY
+Tc=",
  "KeyId": "arn:aws:kms:us-
east-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
  "EncryptionAlgorithm": "SYMMETRIC_DEFAULT"
}
```

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface V2
- AWS SDK for .NET V4

- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- AWS SDK for PHP V3
- AWS SDK for Python (Boto3)
- AWS SDK for Ruby V3

GenerateDataKey

Returns a unique symmetric data key for use outside of AWS KMS. This operation returns a plaintext copy of the data key and a copy that is encrypted under a symmetric encryption KMS key that you specify. The bytes in the plaintext key are random; they are not related to the caller or the KMS key. You can use the plaintext key to encrypt your data outside of AWS KMS and store the encrypted data key with the encrypted data.

To generate a data key, specify the symmetric encryption KMS key that will be used to encrypt the data key. You cannot use an asymmetric KMS key to encrypt data keys. To get the type of your KMS key, use the DescribeKey operation.

You must also specify the length of the data key. Use either the `KeySpec` or `NumberOfBytes` parameters (but not both). For 128-bit and 256-bit data keys, use the `KeySpec` parameter.

To generate a 128-bit SM4 data key (China Regions only), specify a `KeySpec` value of `AES_128` or a `NumberOfBytes` value of 16. The symmetric encryption key used in China Regions to encrypt your data key is an SM4 encryption key.

To get only an encrypted copy of the data key, use GenerateDataKeyWithoutPlaintext.
To generate an asymmetric data key pair, use the GenerateDataKeyPair or GenerateDataKeyPairWithoutPlaintext operation. To get a cryptographically secure random byte string, use GenerateRandom.

You can use an optional encryption context to add additional security to the encryption operation. If you specify an `EncryptionContext`, you must specify the same encryption context (a case-sensitive exact match) when decrypting the encrypted data key. Otherwise, the request to decrypt fails with an `InvalidCiphertextException`. For more information, see Encryption Context in the *AWS Key Management Service Developer Guide*.

`GenerateDataKey` also supports AWS Nitro Enclaves, which provide an isolated compute environment in Amazon EC2. To call `GenerateDataKey` for an AWS Nitro enclave or NitroTPM, use the AWS Nitro Enclaves SDK or any AWS SDK. Use the `Recipient` parameter to provide the attestation document for the attested environment. `GenerateDataKey` returns a copy of the data key encrypted under the specified KMS key, as usual. But instead of a plaintext copy of the data key, the response includes a copy of the data key encrypted under the public key from the attestation document (`CiphertextForRecipient`). For information about the interaction between AWS KMS and AWS Nitro Enclaves or AWS NitroTPM, see Cryptographic attestation support in AWS KMS in the *AWS Key Management Service Developer Guide*.

The KMS key that you use for this operation must be in a compatible key state. For details, see Key states of AWS KMS keys in the *AWS Key Management Service Developer Guide*.

How to use your data key

We recommend that you use the following pattern to encrypt data locally in your application. You can write your own code or use a client-side encryption library, such as the AWS Encryption SDK, the Amazon DynamoDB Encryption Client, or Amazon S3 client-side encryption to do these tasks for you.

To encrypt data outside of AWS KMS:

1. Use the `GenerateDataKey` operation to get a data key.
2. Use the plaintext data key (in the `Plaintext` field of the response) to encrypt your data outside of AWS KMS. Then erase the plaintext data key from memory.
3. Store the encrypted data key (in the `CiphertextBlob` field of the response) with the encrypted data.

To decrypt data outside of AWS KMS:

1. Use the Decrypt operation to decrypt the encrypted data key. The operation returns a plaintext copy of the data key.
2. Use the plaintext data key to decrypt data outside of AWS KMS, then erase the plaintext data key from memory.

Cross-account use: Yes. To perform this operation with a KMS key in a different AWS account, specify the key ARN or alias ARN in the value of the `KeyId` parameter.

Required permissions: kms:GenerateDataKey (key policy)

Related operations:

- Decrypt
- Encrypt
- GenerateDataKeyPair
- GenerateDataKeyPairWithoutPlaintext
- GenerateDataKeyWithoutPlaintext

Eventual consistency: The AWS KMS API follows an eventual consistency model. For more information, see AWS KMS eventual consistency.

Request Syntax

```
{
   "DryRun": boolean,
   "EncryptionContext": {
      "string" : "string"
   },
   "GrantTokens": [ "string" ],
   "KeyId": "string",
   "KeySpec": "string",
   "NumberOfBytes": number,
   "Recipient": {
      "AttestationDocument": blob,
      "KeyEncryptionAlgorithm": "string"
   }
}
```

Request Parameters

For information about the parameters that are common to all actions, see Common Parameters.

The request accepts the following data in JSON format.

> **Note**
>
> In the following list, the required parameters are described first.

KeyId

Specifies the symmetric encryption KMS key that encrypts the data key. You cannot specify an asymmetric KMS key or a KMS key in a custom key store. To get the type and origin of your KMS key, use the DescribeKey operation.

To specify a KMS key, use its key ID, key ARN, alias name, or alias ARN. When using an alias name, prefix it with `"alias/"`. To specify a KMS key in a different AWS account, you must use the key ARN or alias ARN.

For example:

- Key ID: `1234abcd-12ab-34cd-56ef-1234567890ab`
- Key ARN: `arn:aws:kms:us-east-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab`
- Alias name: `alias/ExampleAlias`
- Alias ARN: `arn:aws:kms:us-east-2:111122223333:alias/ExampleAlias`

To get the key ID and key ARN for a KMS key, use ListKeys or DescribeKey. To get the alias name and alias ARN, use ListAliases.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 2048.

Required: Yes

DryRun

Checks if your request will succeed. `DryRun` is an optional parameter.

To learn more about how to use this parameter, see Testing your permissions in the *AWS Key Management Service Developer Guide*.

Type: Boolean

Required: No

EncryptionContext

Specifies the encryption context that will be used when encrypting the data key.

> **Important**
>
> Do not include confidential or sensitive information in this field. This field may be displayed in plaintext in CloudTrail logs and other output.

An *encryption context* is a collection of non-secret key-value pairs that represent additional authenticated data. When you use an encryption context to encrypt data, you must specify the same (an exact case-sensitive match) encryption context to decrypt the data. An encryption context is supported only on operations with symmetric encryption KMS keys. On operations

with symmetric encryption KMS keys, an encryption context is optional, but it is strongly recommended.

For more information, see Encryption context in the *AWS Key Management Service Developer Guide*.

Type: String to string map

Required: No

GrantTokens

A list of grant tokens.

Use a grant token when your permission to call this operation comes from a new grant that has not yet achieved *eventual consistency*. For more information, see Grant token and Using a grant token in the *AWS Key Management Service Developer Guide*.

Type: Array of strings

Array Members: Minimum number of 0 items. Maximum number of 10 items.

Length Constraints: Minimum length of 1. Maximum length of 8192.

Required: No

KeySpec

Specifies the length of the data key. Use `AES_128` to generate a 128-bit symmetric key, or `AES_256` to generate a 256-bit symmetric key.

You must specify either the `KeySpec` or the `NumberOfBytes` parameter (but not both) in every `GenerateDataKey` request.

Type: String

Valid Values: `AES_256 | AES_128`

Required: No

NumberOfBytes

Specifies the length of the data key in bytes. For example, use the value 64 to generate a 512-bit data key (64 bytes is 512 bits). For 128-bit (16-byte) and 256-bit (32-byte) data keys, use the `KeySpec` parameter.

You must specify either the `KeySpec` or the `NumberOfBytes` parameter (but not both) in every `GenerateDataKey` request.

Type: Integer

Valid Range: Minimum value of 1. Maximum value of 1024.

Required: No

Recipient

A signed attestation document from an AWS Nitro enclave or NitroTPM, and the encryption algorithm to use with the public key in the attestation document. The only valid encryption algorithm is `RSAES_OAEP_SHA_256`.

This parameter supports the AWS Nitro Enclaves SDK or any AWS SDK for AWS Nitro Enclaves. It supports any AWS SDK for AWS NitroTPM.

When you use this parameter, instead of returning the plaintext data key, AWS KMS encrypts the plaintext data key under the public key in the attestation document, and returns the resulting ciphertext in the `CiphertextForRecipient` field in the response. This ciphertext can be decrypted only with the private key in the enclave. The `CiphertextBlob` field in the response contains a copy of the data key encrypted under the KMS key specified by the `KeyId` parameter. The `Plaintext` field in the response is null or empty.

For information about the interaction between AWS KMS and AWS Nitro Enclaves or AWS NitroTPM, see Cryptographic attestation support in AWS KMS in the *AWS Key Management Service Developer Guide*.

Type: RecipientInfo object

Required: No

Response Syntax

```
{
   "CiphertextBlob": blob,
   "CiphertextForRecipient": blob,
   "KeyId": "string",
   "KeyMaterialId": "string",
   "Plaintext": blob
```

```
}
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

CiphertextBlob

The encrypted copy of the data key. When you use the HTTP API or the AWS CLI, the value is Base64-encoded. Otherwise, it is not Base64-encoded.

Type: Base64-encoded binary data object

Length Constraints: Minimum length of 1. Maximum length of 6144.

CiphertextForRecipient

The plaintext data key encrypted with the public key from the attestation document. This ciphertext can be decrypted only by using a private key from the attested environment.

This field is included in the response only when the `Recipient` parameter in the request includes a valid attestation document from an AWS Nitro enclave or NitroTPM. For information about the interaction between AWS KMS and AWS Nitro Enclaves or AWS NitroTPM, see Cryptographic attestation support in AWS KMS in the *AWS Key Management Service Developer Guide*.

Type: Base64-encoded binary data object

Length Constraints: Minimum length of 1. Maximum length of 6144.

KeyId

The Amazon Resource Name (key ARN) of the KMS key that encrypted the data key.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 2048.

KeyMaterialId

The identifier of the key material used to encrypt the data key. This field is omitted if the request includes the `Recipient` parameter.

Type: String

Length Constraints: Fixed length of 64.

Pattern: `^[a-f0-9]+$`

Plaintext

The plaintext data key. When you use the HTTP API or the AWS CLI, the value is Base64-encoded. Otherwise, it is not Base64-encoded. Use this data key to encrypt your data outside of KMS. Then, remove it from memory as soon as possible.

If the response includes the `CiphertextForRecipient` field, the `Plaintext` field is null or empty.

Type: Base64-encoded binary data object

Length Constraints: Minimum length of 1. Maximum length of 4096.

Errors

For information about the errors that are common to all actions, see Common Error Types.

DependencyTimeoutException

The system timed out while trying to fulfill the request. You can retry the request.

HTTP Status Code: 500

DisabledException

The request was rejected because the specified KMS key is not enabled.

HTTP Status Code: 400

DryRunOperationException

The request was rejected because the DryRun parameter was specified.

HTTP Status Code: 400

InvalidGrantTokenException

The request was rejected because the specified grant token is not valid.

HTTP Status Code: 400

InvalidKeyUsageException

The request was rejected for one of the following reasons:

- The KeyUsage value of the KMS key is incompatible with the API operation.
- The encryption algorithm or signing algorithm specified for the operation is incompatible with the type of key material in the KMS key (KeySpec).

For encrypting, decrypting, re-encrypting, and generating data keys, the KeyUsage must be ENCRYPT_DECRYPT. For signing and verifying messages, the KeyUsage must be SIGN_VERIFY. For generating and verifying message authentication codes (MACs), the KeyUsage must be GENERATE_VERIFY_MAC. For deriving key agreement secrets, the KeyUsage must be KEY_AGREEMENT. For encapsulating and decapsulating shared secrets, the KeyUsage must be ENCAPSULATE_DECAPSULATE. To find the KeyUsage of a KMS key, use the DescribeKey operation.

To find the encryption or signing algorithms supported for a particular KMS key, use the DescribeKey operation.

HTTP Status Code: 400

KeyUnavailableException

The request was rejected because the specified KMS key was not available. You can retry the request.

HTTP Status Code: 500

KMSInternalException

The request was rejected because an internal exception occurred. The request can be retried.

HTTP Status Code: 500

KMSInvalidStateException

The request was rejected because the state of the specified resource is not valid for this request.

This exceptions means one of the following:

- The key state of the KMS key is not compatible with the operation.

  To find the key state, use the DescribeKey operation. For more information about which key states are compatible with each AWS KMS operation, see Key states of AWS KMS keys in the AWS Key Management Service Developer Guide .

- For cryptographic operations on KMS keys in custom key stores, this exception represents a general failure with many possible causes. To identify the cause, see the error message that accompanies the exception.

HTTP Status Code: 400

NotFoundException

The request was rejected because the specified entity or resource could not be found.

HTTP Status Code: 400

Examples

The following examples are formatted for legibility.

Example Request

This example illustrates one usage of GenerateDataKey.

```
POST / HTTP/1.1
Host: kms.us-east-2.amazonaws.com
Content-Length: 50
X-Amz-Target: TrentService.GenerateDataKey
X-Amz-Date: 20161112T000940Z
Content-Type: application/x-amz-json-1.1
Authorization: AWS4-HMAC-SHA256\
 Credential=AKIAI44QH8DHBEXAMPLE/20161112/us-east-2/kms/aws4_request,\
 SignedHeaders=content-type;host;x-amz-date;x-amz-target,\
 Signature=815ac4ccbb5c53b8ca015f979704c7953bb0068bf53f4e0b7c6886ed5b0a8fe4

{
  "KeyId": "alias/ExampleAlias",
  "KeySpec": "AES_256"
}
```

Example Response

This example illustrates one usage of GenerateDataKey.

```
HTTP/1.1 200 OK
Server: Server
```

```
Date: Sat, 12 Nov 2016 00:09:40 GMT
Content-Type: application/x-amz-json-1.1
Content-Length: 390
Connection: keep-alive
x-amzn-RequestId: 4e6fc242-a86c-11e6-aff0-8333261e2fbd

{
  "CiphertextBlob":
 "AQEDAHjRYf5WytIc0C857tFSnBaPn2F8DgfmThbJlGfR8P3WlwAAAH4wfAYJKoZIhvcNAQcGoG8wbQIBADBoBgkqhkiG9
+YdhV8MrkBQPeac0ReRVNDt9qleAt+SHgIRF8P0H+7U=",
  "KeyId": "arn:aws:kms:us-
east-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
  "KeyMaterialId": "0b7fd7ddbac6eef27907413567cad8c810e2883dc8a7534067a82ee1142fc1e6",
  "Plaintext": "VdzKNHGzUAzJeRBVY+uUmofUGGiDzyB3+i9fVkh3piw="
}
```

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface V2
- AWS SDK for .NET V4
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- AWS SDK for PHP V3
- AWS SDK for Python (Boto3)
- AWS SDK for Ruby V3

GenerateDataKeyPair

Returns a unique asymmetric data key pair for use outside of AWS KMS. This operation returns a plaintext public key, a plaintext private key, and a copy of the private key that is encrypted under the symmetric encryption KMS key you specify. You can use the data key pair to perform asymmetric cryptography and implement digital signatures outside of AWS KMS. The bytes in the keys are random; they are not related to the caller or to the KMS key that is used to encrypt the private key.

You can use the public key that `GenerateDataKeyPair` returns to encrypt data or verify a signature outside of AWS KMS. Then, store the encrypted private key with the data. When you are ready to decrypt data or sign a message, you can use the Decrypt operation to decrypt the encrypted private key.

To generate a data key pair, you must specify a symmetric encryption KMS key to encrypt the private key in a data key pair. You cannot use an asymmetric KMS key or a KMS key in a custom key store. To get the type and origin of your KMS key, use the DescribeKey operation.

Use the `KeyPairSpec` parameter to choose an RSA or Elliptic Curve (ECC) data key pair. In China Regions, you can also choose an SM2 data key pair. AWS KMS recommends that you use ECC key pairs for signing, and use RSA and SM2 key pairs for either encryption or signing, but not both. However, AWS KMS cannot enforce any restrictions on the use of data key pairs outside of AWS KMS.

If you are using the data key pair to encrypt data, or for any operation where you don't immediately need a private key, consider using the GenerateDataKeyPairWithoutPlaintext operation. `GenerateDataKeyPairWithoutPlaintext` returns a plaintext public key and an encrypted private key, but omits the plaintext private key that you need only to decrypt ciphertext or sign a message. Later, when you need to decrypt the data or sign a message, use the Decrypt operation to decrypt the encrypted private key in the data key pair.

`GenerateDataKeyPair` returns a unique data key pair for each request. The bytes in the keys are random; they are not related to the caller or the KMS key that is used to encrypt the private key. The public key is a DER-encoded X.509 SubjectPublicKeyInfo, as specified in RFC 5280. The private key is a DER-encoded PKCS8 PrivateKeyInfo, as specified in RFC 5958.

Note

For ML-KEM data key pairs, the DER-encoded PKCS8 PrivateKeyInfo contains the private key in seed format. As a result, the length of the private key is the same for all ML-KEM key pair specs.

`GenerateDataKeyPair` also supports AWS Nitro Enclaves, which provide an isolated compute environment in Amazon EC2. To call `GenerateDataKeyPair` for an AWS Nitro enclave or NitroTPM, use the AWS Nitro Enclaves SDK or any AWS SDK. Use the `Recipient` parameter to provide the attestation document for the attested environment. `GenerateDataKeyPair` returns the public data key and a copy of the private data key encrypted under the specified KMS key, as usual. But instead of a plaintext copy of the private data key (`PrivateKeyPlaintext`), the response includes a copy of the private data key encrypted under the public key from the attestation document (`CiphertextForRecipient`). For information about the interaction between AWS KMS and AWS Nitro Enclaves or AWS NitroTPM, see Cryptographic attestation support in AWS KMS in the *AWS Key Management Service Developer Guide*.

You can use an optional encryption context to add additional security to the encryption operation. If you specify an `EncryptionContext`, you must specify the same encryption context (a case-sensitive exact match) when decrypting the encrypted data key. Otherwise, the request to decrypt fails with an `InvalidCiphertextException`. For more information, see Encryption Context in the *AWS Key Management Service Developer Guide*.

The KMS key that you use for this operation must be in a compatible key state. For details, see Key states of AWS KMS keys in the *AWS Key Management Service Developer Guide*.

Cross-account use: Yes. To perform this operation with a KMS key in a different AWS account, specify the key ARN or alias ARN in the value of the `KeyId` parameter.

Required permissions: kms:GenerateDataKeyPair (key policy)

Related operations:

- Decrypt
- Encrypt
- GenerateDataKey
- GenerateDataKeyPairWithoutPlaintext

- GenerateDataKeyWithoutPlaintext

Eventual consistency: The AWS KMS API follows an eventual consistency model. For more information, see AWS KMS eventual consistency.

Request Syntax

```
{
   "DryRun": boolean,
   "EncryptionContext": {
      "string" : "string"
   },
   "GrantTokens": [ "string" ],
   "KeyId": "string",
   "KeyPairSpec": "string",
   "Recipient": {
      "AttestationDocument": blob,
      "KeyEncryptionAlgorithm": "string"
   }
}
```

Request Parameters

For information about the parameters that are common to all actions, see Common Parameters.

The request accepts the following data in JSON format.

> **Note**
>
> In the following list, the required parameters are described first.

KeyId

Specifies the symmetric encryption KMS key that encrypts the private key in the data key pair. You cannot specify an asymmetric KMS key or a KMS key in a custom key store. To get the type and origin of your KMS key, use the DescribeKey operation.

To specify a KMS key, use its key ID, key ARN, alias name, or alias ARN. When using an alias name, prefix it with `"alias/"`. To specify a KMS key in a different AWS account, you must use the key ARN or alias ARN.

For example:

- Key ID: `1234abcd-12ab-34cd-56ef-1234567890ab`
- Key ARN: `arn:aws:kms:us-east-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab`
- Alias name: `alias/ExampleAlias`
- Alias ARN: `arn:aws:kms:us-east-2:111122223333:alias/ExampleAlias`

To get the key ID and key ARN for a KMS key, use ListKeys or DescribeKey. To get the alias name and alias ARN, use ListAliases.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 2048.

Required: Yes

KeyPairSpec

Determines the type of data key pair that is generated.

The AWS KMS rule that restricts the use of asymmetric RSA and SM2 KMS keys to encrypt and decrypt or to sign and verify (but not both), the rule that permits you to use ECC KMS keys only to sign and verify, and the rule that permits you to use ML-DSA key pairs to sign and verify only are not effective on data key pairs, which are used outside of AWS KMS. The SM2 key spec is only available in China Regions. The ML-KEM key specs are not available in China Regions.

Type: String

Valid Values: `RSA_2048 | RSA_3072 | RSA_4096 | ECC_NIST_P256 | ECC_NIST_P384 | ECC_NIST_P521 | ECC_SECG_P256K1 | SM2 | ECC_NIST_EDWARDS25519 | ML_KEM_512 | ML_KEM_768 | ML_KEM_1024`

Required: Yes

DryRun

Checks if your request will succeed. `DryRun` is an optional parameter.

To learn more about how to use this parameter, see Testing your permissions in the *AWS Key Management Service Developer Guide*.

Type: Boolean

Required: No

EncryptionContext

Specifies the encryption context that will be used when encrypting the private key in the data key pair.

> **Important**
>
> Do not include confidential or sensitive information in this field. This field may be displayed in plaintext in CloudTrail logs and other output.

An *encryption context* is a collection of non-secret key-value pairs that represent additional authenticated data. When you use an encryption context to encrypt data, you must specify the same (an exact case-sensitive match) encryption context to decrypt the data. An encryption context is supported only on operations with symmetric encryption KMS keys. On operations with symmetric encryption KMS keys, an encryption context is optional, but it is strongly recommended.

For more information, see Encryption context in the *AWS Key Management Service Developer Guide*.

Type: String to string map

Required: No

GrantTokens

A list of grant tokens.

Use a grant token when your permission to call this operation comes from a new grant that has not yet achieved *eventual consistency*. For more information, see Grant token and Using a grant token in the *AWS Key Management Service Developer Guide*.

Type: Array of strings

Array Members: Minimum number of 0 items. Maximum number of 10 items.

Length Constraints: Minimum length of 1. Maximum length of 8192.

Required: No

Recipient

A signed attestation document from an AWS Nitro enclave or NitroTPM, and the encryption algorithm to use with the public key in the attestation document. The only valid encryption algorithm is RSAES_OAEP_SHA_256.

This parameter only supports attestation documents for AWS Nitro Enclaves or AWS NitroTPM. To call GenerateDataKeyPair generate an attestation document use either AWS Nitro Enclaves SDK for an AWS Nitro Enclaves or AWS NitroTPM tools for AWS NitroTPM. Then use the Recipient parameter from any AWS SDK to provide the attestation document for the attested environment.

When you use this parameter, instead of returning a plaintext copy of the private data key, AWS KMS encrypts the plaintext private data key under the public key in the attestation document, and returns the resulting ciphertext in the `CiphertextForRecipient` field in the response. This ciphertext can be decrypted only with the private key in the attested environment. The `CiphertextBlob` field in the response contains a copy of the private data key encrypted under the KMS key specified by the `KeyId` parameter. The `PrivateKeyPlaintext` field in the response is null or empty.

For information about the interaction between AWS KMS and AWS Nitro Enclaves or AWS NitroTPM, see Cryptographic attestation support in AWS KMS in the *AWS Key Management Service Developer Guide*.

Type: RecipientInfo object

Required: No

Response Syntax

```
{
   "CiphertextForRecipient": blob,
   "KeyId": "string",
   "KeyMaterialId": "string",
   "KeyPairSpec": "string",
   "PrivateKeyCiphertextBlob": blob,
   "PrivateKeyPlaintext": blob,
   "PublicKey": blob
}
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

CiphertextForRecipient

The plaintext private data key encrypted with the public key from the attestation document. This ciphertext can be decrypted only by using a private key from the attested environment.

This field is included in the response only when the `Recipient` parameter in the request includes a valid attestation document from an AWS Nitro enclave or NitroTPM. For information about the interaction between AWS KMS and AWS Nitro Enclaves or AWS NitroTPM, see Cryptographic attestation support in AWS KMS in the *AWS Key Management Service Developer Guide*.

Type: Base64-encoded binary data object

Length Constraints: Minimum length of 1. Maximum length of 6144.

KeyId

The Amazon Resource Name (key ARN) of the KMS key that encrypted the private key.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 2048.

KeyMaterialId

The identifier of the key material used to encrypt the private key.

Type: String

Length Constraints: Fixed length of 64.

Pattern: `^[a-f0-9]+$`

KeyPairSpec

The type of data key pair that was generated.

Type: String

Valid Values: `RSA_2048 | RSA_3072 | RSA_4096 | ECC_NIST_P256 | ECC_NIST_P384 | ECC_NIST_P521 | ECC_SECG_P256K1 | SM2 | ECC_NIST_EDWARDS25519 | ML_KEM_512 | ML_KEM_768 | ML_KEM_1024`

PrivateKeyCiphertextBlob

The encrypted copy of the private key. When you use the HTTP API or the AWS CLI, the value is Base64-encoded. Otherwise, it is not Base64-encoded.

Type: Base64-encoded binary data object

Length Constraints: Minimum length of 1. Maximum length of 6144.

PrivateKeyPlaintext

The plaintext copy of the private key. When you use the HTTP API or the AWS CLI, the value is Base64-encoded. Otherwise, it is not Base64-encoded.

If the response includes the `CiphertextForRecipient` field, the `PrivateKeyPlaintext` field is null or empty.

Type: Base64-encoded binary data object

Length Constraints: Minimum length of 1. Maximum length of 4096.

PublicKey

The public key (in plaintext). When you use the HTTP API or the AWS CLI, the value is Base64-encoded. Otherwise, it is not Base64-encoded.

Type: Base64-encoded binary data object

Length Constraints: Minimum length of 1. Maximum length of 8192.

Errors

For information about the errors that are common to all actions, see Common Error Types.

DependencyTimeoutException

The system timed out while trying to fulfill the request. You can retry the request.

HTTP Status Code: 500

DisabledException

The request was rejected because the specified KMS key is not enabled.

HTTP Status Code: 400

DryRunOperationException

The request was rejected because the DryRun parameter was specified.

HTTP Status Code: 400

InvalidGrantTokenException

The request was rejected because the specified grant token is not valid.

HTTP Status Code: 400

InvalidKeyUsageException

The request was rejected for one of the following reasons:

- The `KeyUsage` value of the KMS key is incompatible with the API operation.
- The encryption algorithm or signing algorithm specified for the operation is incompatible with the type of key material in the KMS key (`KeySpec`).

For encrypting, decrypting, re-encrypting, and generating data keys, the `KeyUsage` must be `ENCRYPT_DECRYPT`. For signing and verifying messages, the `KeyUsage` must be `SIGN_VERIFY`. For generating and verifying message authentication codes (MACs), the `KeyUsage` must be `GENERATE_VERIFY_MAC`. For deriving key agreement secrets, the `KeyUsage` must be `KEY_AGREEMENT`. For encapsulating and decapsulating shared secrets, the `KeyUsage` must be `ENCAPSULATE_DECAPSULATE`. To find the `KeyUsage` of a KMS key, use the DescribeKey operation.

To find the encryption or signing algorithms supported for a particular KMS key, use the DescribeKey operation.

HTTP Status Code: 400

KeyUnavailableException

The request was rejected because the specified KMS key was not available. You can retry the request.

HTTP Status Code: 500

KMSInternalException

The request was rejected because an internal exception occurred. The request can be retried.

HTTP Status Code: 500

KMSInvalidStateException

The request was rejected because the state of the specified resource is not valid for this request.

This exceptions means one of the following:

- The key state of the KMS key is not compatible with the operation.

  To find the key state, use the DescribeKey operation. For more information about which key states are compatible with each AWS KMS operation, see Key states of AWS KMS keys in the AWS Key Management Service Developer Guide .
- For cryptographic operations on KMS keys in custom key stores, this exception represents a general failure with many possible causes. To identify the cause, see the error message that accompanies the exception.

HTTP Status Code: 400

NotFoundException

The request was rejected because the specified entity or resource could not be found.

HTTP Status Code: 400

UnsupportedOperationException

The request was rejected because a specified parameter is not supported or a specified resource is not valid for this operation.

HTTP Status Code: 400

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface V2
- AWS SDK for .NET V4

- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- AWS SDK for PHP V3
- AWS SDK for Python (Boto3)
- AWS SDK for Ruby V3

GenerateDataKeyPairWithoutPlaintext

Returns a unique asymmetric data key pair for use outside of AWS KMS. This operation returns a plaintext public key and a copy of the private key that is encrypted under the symmetric encryption KMS key you specify. Unlike GenerateDataKeyPair, this operation does not return a plaintext private key. The bytes in the keys are random; they are not related to the caller or to the KMS key that is used to encrypt the private key.

You can use the public key that `GenerateDataKeyPairWithoutPlaintext` returns to encrypt data or verify a signature outside of AWS KMS. Then, store the encrypted private key with the data. When you are ready to decrypt data or sign a message, you can use the Decrypt operation to decrypt the encrypted private key.

To generate a data key pair, you must specify a symmetric encryption KMS key to encrypt the private key in a data key pair. You cannot use an asymmetric KMS key or a KMS key in a custom key store. To get the type and origin of your KMS key, use the DescribeKey operation.

Use the `KeyPairSpec` parameter to choose an RSA or Elliptic Curve (ECC) data key pair. In China Regions, you can also choose an SM2 data key pair. AWS KMS recommends that you use ECC key pairs for signing, and use RSA and SM2 key pairs for either encryption or signing, but not both. However, AWS KMS cannot enforce any restrictions on the use of data key pairs outside of AWS KMS.

`GenerateDataKeyPairWithoutPlaintext` returns a unique data key pair for each request. The bytes in the key are not related to the caller or KMS key that is used to encrypt the private key. The public key is a DER-encoded X.509 SubjectPublicKeyInfo, as specified in RFC 5280.

> **Note**
>
> For ML-KEM data key pairs, the DER-encoded PKCS8 PrivateKeyInfo contains the private key in seed format. As a result, the length of the private key is the same for all ML-KEM key pair specs.

You can use an optional encryption context to add additional security to the encryption operation. If you specify an `EncryptionContext`, you must specify the same encryption context (a case-sensitive exact match) when decrypting the encrypted data key. Otherwise, the request to decrypt fails with an `InvalidCiphertextException`. For more information, see Encryption Context in the *AWS Key Management Service Developer Guide*.

The KMS key that you use for this operation must be in a compatible key state. For details, see Key states of AWS KMS keys in the *AWS Key Management Service Developer Guide*.

Cross-account use: Yes. To perform this operation with a KMS key in a different AWS account, specify the key ARN or alias ARN in the value of the `KeyId` parameter.

Required permissions: kms:GenerateDataKeyPairWithoutPlaintext (key policy)

Related operations:

- Decrypt
- Encrypt
- GenerateDataKey
- GenerateDataKeyPair
- GenerateDataKeyWithoutPlaintext

Eventual consistency: The AWS KMS API follows an eventual consistency model. For more information, see AWS KMS eventual consistency.

Request Syntax

```
{
   "DryRun": boolean,
   "EncryptionContext": {
      "string" : "string"
   },
   "GrantTokens": [ "string" ],
   "KeyId": "string",
   "KeyPairSpec": "string"
}
```

Request Parameters

For information about the parameters that are common to all actions, see Common Parameters.

The request accepts the following data in JSON format.

> **Note**
>
> In the following list, the required parameters are described first.

KeyId

Specifies the symmetric encryption KMS key that encrypts the private key in the data key pair. You cannot specify an asymmetric KMS key or a KMS key in a custom key store. To get the type and origin of your KMS key, use the DescribeKey operation.

To specify a KMS key, use its key ID, key ARN, alias name, or alias ARN. When using an alias name, prefix it with `"alias/"`. To specify a KMS key in a different AWS account, you must use the key ARN or alias ARN.

For example:

- Key ID: `1234abcd-12ab-34cd-56ef-1234567890ab`
- Key ARN: `arn:aws:kms:us-east-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab`
- Alias name: `alias/ExampleAlias`
- Alias ARN: `arn:aws:kms:us-east-2:111122223333:alias/ExampleAlias`

To get the key ID and key ARN for a KMS key, use ListKeys or DescribeKey. To get the alias name and alias ARN, use ListAliases.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 2048.

Required: Yes

KeyPairSpec

Determines the type of data key pair that is generated.

The AWS KMS rule that restricts the use of asymmetric RSA and SM2 KMS keys to encrypt and decrypt or to sign and verify (but not both), the rule that permits you to use ECC KMS keys only to sign and verify, and the rule that permits you to use ML-DSA key pairs to sign and verify only are not effective on data key pairs, which are used outside of AWS KMS. The SM2 key spec is only available in China Regions. The ML-KEM key specs are not available in China Regions.

Type: String

Valid Values: `RSA_2048 | RSA_3072 | RSA_4096 | ECC_NIST_P256 | ECC_NIST_P384 | ECC_NIST_P521 | ECC_SECG_P256K1 | SM2 | ECC_NIST_EDWARDS25519 | ML_KEM_512 | ML_KEM_768 | ML_KEM_1024`

Required: Yes

DryRun

Checks if your request will succeed. `DryRun` is an optional parameter.

To learn more about how to use this parameter, see Testing your permissions in the *AWS Key Management Service Developer Guide*.

Type: Boolean

Required: No

EncryptionContext

Specifies the encryption context that will be used when encrypting the private key in the data key pair.

> **Important**
>
> Do not include confidential or sensitive information in this field. This field may be displayed in plaintext in CloudTrail logs and other output.

An *encryption context* is a collection of non-secret key-value pairs that represent additional authenticated data. When you use an encryption context to encrypt data, you must specify the same (an exact case-sensitive match) encryption context to decrypt the data. An encryption context is supported only on operations with symmetric encryption KMS keys. On operations with symmetric encryption KMS keys, an encryption context is optional, but it is strongly recommended.

For more information, see Encryption context in the *AWS Key Management Service Developer Guide*.

Type: String to string map

Required: No

GrantTokens

A list of grant tokens.

Use a grant token when your permission to call this operation comes from a new grant that has not yet achieved *eventual consistency*. For more information, see Grant token and Using a grant token in the *AWS Key Management Service Developer Guide*.

Type: Array of strings

Array Members: Minimum number of 0 items. Maximum number of 10 items.

Length Constraints: Minimum length of 1. Maximum length of 8192.

Required: No

Response Syntax

```
{
   "KeyId": "string",
   "KeyMaterialId": "string",
   "KeyPairSpec": "string",
   "PrivateKeyCiphertextBlob": blob,
   "PublicKey": blob
}
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

KeyId

The Amazon Resource Name (key ARN) of the KMS key that encrypted the private key.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 2048.

KeyMaterialId

The identifier of the key material used to encrypt the private key.

Type: String

Length Constraints: Fixed length of 64.

Pattern: `^[a-f0-9]+$`

KeyPairSpec

The type of data key pair that was generated.

Type: String

Valid Values: `RSA_2048 | RSA_3072 | RSA_4096 | ECC_NIST_P256 | ECC_NIST_P384 | ECC_NIST_P521 | ECC_SECG_P256K1 | SM2 | ECC_NIST_EDWARDS25519 | ML_KEM_512 | ML_KEM_768 | ML_KEM_1024`

PrivateKeyCiphertextBlob

The encrypted copy of the private key. When you use the HTTP API or the AWS CLI, the value is Base64-encoded. Otherwise, it is not Base64-encoded.

Type: Base64-encoded binary data object

Length Constraints: Minimum length of 1. Maximum length of 6144.

PublicKey

The public key (in plaintext). When you use the HTTP API or the AWS CLI, the value is Base64-encoded. Otherwise, it is not Base64-encoded.

Type: Base64-encoded binary data object

Length Constraints: Minimum length of 1. Maximum length of 8192.

Errors

For information about the errors that are common to all actions, see Common Error Types.

DependencyTimeoutException

The system timed out while trying to fulfill the request. You can retry the request.

HTTP Status Code: 500

DisabledException

The request was rejected because the specified KMS key is not enabled.

HTTP Status Code: 400

DryRunOperationException

The request was rejected because the DryRun parameter was specified.

HTTP Status Code: 400

InvalidGrantTokenException

The request was rejected because the specified grant token is not valid.

HTTP Status Code: 400

InvalidKeyUsageException

The request was rejected for one of the following reasons:

- The `KeyUsage` value of the KMS key is incompatible with the API operation.
- The encryption algorithm or signing algorithm specified for the operation is incompatible with the type of key material in the KMS key (`KeySpec`).

For encrypting, decrypting, re-encrypting, and generating data keys, the `KeyUsage` must be `ENCRYPT_DECRYPT`. For signing and verifying messages, the `KeyUsage` must be `SIGN_VERIFY`. For generating and verifying message authentication codes (MACs), the `KeyUsage` must be `GENERATE_VERIFY_MAC`. For deriving key agreement secrets, the `KeyUsage` must be `KEY_AGREEMENT`. For encapsulating and decapsulating shared secrets, the `KeyUsage` must be `ENCAPSULATE_DECAPSULATE`. To find the `KeyUsage` of a KMS key, use the DescribeKey operation.

To find the encryption or signing algorithms supported for a particular KMS key, use the DescribeKey operation.

HTTP Status Code: 400

KeyUnavailableException

The request was rejected because the specified KMS key was not available. You can retry the request.

HTTP Status Code: 500

KMSInternalException

The request was rejected because an internal exception occurred. The request can be retried.

HTTP Status Code: 500

KMSInvalidStateException

The request was rejected because the state of the specified resource is not valid for this request.

This exceptions means one of the following:

- The key state of the KMS key is not compatible with the operation.

  To find the key state, use the DescribeKey operation. For more information about which key states are compatible with each AWS KMS operation, see Key states of AWS KMS keys in the AWS Key Management Service Developer Guide .
- For cryptographic operations on KMS keys in custom key stores, this exception represents a general failure with many possible causes. To identify the cause, see the error message that accompanies the exception.

HTTP Status Code: 400

NotFoundException

The request was rejected because the specified entity or resource could not be found.

HTTP Status Code: 400

UnsupportedOperationException

The request was rejected because a specified parameter is not supported or a specified resource is not valid for this operation.

HTTP Status Code: 400

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface V2
- AWS SDK for .NET V4

- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- AWS SDK for PHP V3
- AWS SDK for Python (Boto3)
- AWS SDK for Ruby V3

GenerateDataKeyWithoutPlaintext

Returns a unique symmetric data key for use outside of AWS KMS. This operation returns a data key that is encrypted under a symmetric encryption KMS key that you specify. The bytes in the key are random; they are not related to the caller or to the KMS key.

`GenerateDataKeyWithoutPlaintext` is identical to the GenerateDataKey operation except that it does not return a plaintext copy of the data key.

This operation is useful for systems that need to encrypt data at some point, but not immediately. When you need to encrypt the data, you call the Decrypt operation on the encrypted copy of the key.

It's also useful in distributed systems with different levels of trust. For example, you might store encrypted data in containers. One component of your system creates new containers and stores an encrypted data key with each container. Then, a different component puts the data into the containers. That component first decrypts the data key, uses the plaintext data key to encrypt data, puts the encrypted data into the container, and then destroys the plaintext data key. In this system, the component that creates the containers never sees the plaintext data key.

To request an asymmetric data key pair, use the GenerateDataKeyPair or GenerateDataKeyPairWithoutPlaintext operations.

To generate a data key, you must specify the symmetric encryption KMS key that is used to encrypt the data key. You cannot use an asymmetric KMS key or a key in a custom key store to generate a data key. To get the type of your KMS key, use the DescribeKey operation.

You must also specify the length of the data key. Use either the `KeySpec` or `NumberOfBytes` parameters (but not both). For 128-bit and 256-bit data keys, use the `KeySpec` parameter.

To generate an SM4 data key (China Regions only), specify a `KeySpec` value of `AES_128` or `NumberOfBytes` value of 16. The symmetric encryption key used in China Regions to encrypt your data key is an SM4 encryption key.

If the operation succeeds, you will find the encrypted copy of the data key in the `CiphertextBlob` field.

You can use an optional encryption context to add additional security to the encryption operation. If you specify an `EncryptionContext`, you must specify the same encryption context (a case-

sensitive exact match) when decrypting the encrypted data key. Otherwise, the request to decrypt fails with an `InvalidCiphertextException`. For more information, see Encryption Context in the *AWS Key Management Service Developer Guide*.

The KMS key that you use for this operation must be in a compatible key state. For details, see Key states of AWS KMS keys in the *AWS Key Management Service Developer Guide*.

Cross-account use: Yes. To perform this operation with a KMS key in a different AWS account, specify the key ARN or alias ARN in the value of the `KeyId` parameter.

Required permissions: kms:GenerateDataKeyWithoutPlaintext (key policy)

Related operations:

- Decrypt
- Encrypt
- GenerateDataKey
- GenerateDataKeyPair
- GenerateDataKeyPairWithoutPlaintext

Eventual consistency: The AWS KMS API follows an eventual consistency model. For more information, see AWS KMS eventual consistency.

Request Syntax

```
{
   "DryRun": boolean,
   "EncryptionContext": {
      "string" : "string"
   },
   "GrantTokens": [ "string" ],
   "KeyId": "string",
   "KeySpec": "string",
   "NumberOfBytes": number
}
```

Request Parameters

For information about the parameters that are common to all actions, see Common Parameters.

The request accepts the following data in JSON format.

> **Note**
>
> In the following list, the required parameters are described first.

KeyId

Specifies the symmetric encryption KMS key that encrypts the data key. You cannot specify an asymmetric KMS key or a KMS key in a custom key store. To get the type and origin of your KMS key, use the DescribeKey operation.

To specify a KMS key, use its key ID, key ARN, alias name, or alias ARN. When using an alias name, prefix it with `"alias/"`. To specify a KMS key in a different AWS account, you must use the key ARN or alias ARN.

For example:

- Key ID: `1234abcd-12ab-34cd-56ef-1234567890ab`
- Key ARN: `arn:aws:kms:us-east-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab`
- Alias name: `alias/ExampleAlias`
- Alias ARN: `arn:aws:kms:us-east-2:111122223333:alias/ExampleAlias`

To get the key ID and key ARN for a KMS key, use ListKeys or DescribeKey. To get the alias name and alias ARN, use ListAliases.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 2048.

Required: Yes

DryRun

Checks if your request will succeed. `DryRun` is an optional parameter.

To learn more about how to use this parameter, see Testing your permissions in the *AWS Key Management Service Developer Guide*.

Type: Boolean

Required: No

EncryptionContext

Specifies the encryption context that will be used when encrypting the data key.

> **Important**
>
> Do not include confidential or sensitive information in this field. This field may be displayed in plaintext in CloudTrail logs and other output.

An *encryption context* is a collection of non-secret key-value pairs that represent additional authenticated data. When you use an encryption context to encrypt data, you must specify the same (an exact case-sensitive match) encryption context to decrypt the data. An encryption context is supported only on operations with symmetric encryption KMS keys. On operations with symmetric encryption KMS keys, an encryption context is optional, but it is strongly recommended.

For more information, see Encryption context in the *AWS Key Management Service Developer Guide*.

Type: String to string map

Required: No

GrantTokens

A list of grant tokens.

Use a grant token when your permission to call this operation comes from a new grant that has not yet achieved *eventual consistency*. For more information, see Grant token and Using a grant token in the *AWS Key Management Service Developer Guide*.

Type: Array of strings

Array Members: Minimum number of 0 items. Maximum number of 10 items.

Length Constraints: Minimum length of 1. Maximum length of 8192.

Required: No

KeySpec

The length of the data key. Use AES_128 to generate a 128-bit symmetric key, or AES_256 to generate a 256-bit symmetric key.

Type: String

Valid Values: AES_256 | AES_128

Required: No

NumberOfBytes

The length of the data key in bytes. For example, use the value 64 to generate a 512-bit data key (64 bytes is 512 bits). For common key lengths (128-bit and 256-bit symmetric keys), we recommend that you use the KeySpec field instead of this one.

Type: Integer

Valid Range: Minimum value of 1. Maximum value of 1024.

Required: No

Response Syntax

```
{
   "CiphertextBlob": blob,
   "KeyId": "string",
   "KeyMaterialId": "string"
}
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

CiphertextBlob

The encrypted data key. When you use the HTTP API or the AWS CLI, the value is Base64-encoded. Otherwise, it is not Base64-encoded.

Type: Base64-encoded binary data object

Length Constraints: Minimum length of 1. Maximum length of 6144.

KeyId

The Amazon Resource Name (key ARN) of the KMS key that encrypted the data key.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 2048.

KeyMaterialId

The identifier of the key material used to encrypt the data key.

Type: String

Length Constraints: Fixed length of 64.

Pattern: `^[a-f0-9]+$`

Errors

For information about the errors that are common to all actions, see Common Error Types.

DependencyTimeoutException

The system timed out while trying to fulfill the request. You can retry the request.

HTTP Status Code: 500

DisabledException

The request was rejected because the specified KMS key is not enabled.

HTTP Status Code: 400

DryRunOperationException

The request was rejected because the DryRun parameter was specified.

HTTP Status Code: 400

InvalidGrantTokenException

The request was rejected because the specified grant token is not valid.

HTTP Status Code: 400

InvalidKeyUsageException

The request was rejected for one of the following reasons:

- The `KeyUsage` value of the KMS key is incompatible with the API operation.
- The encryption algorithm or signing algorithm specified for the operation is incompatible with the type of key material in the KMS key (`KeySpec`).

For encrypting, decrypting, re-encrypting, and generating data keys, the `KeyUsage` must be `ENCRYPT_DECRYPT`. For signing and verifying messages, the `KeyUsage` must be `SIGN_VERIFY`. For generating and verifying message authentication codes (MACs), the `KeyUsage` must be `GENERATE_VERIFY_MAC`. For deriving key agreement secrets, the `KeyUsage` must be `KEY_AGREEMENT`. For encapsulating and decapsulating shared secrets, the `KeyUsage` must be `ENCAPSULATE_DECAPSULATE`. To find the `KeyUsage` of a KMS key, use the DescribeKey operation.

To find the encryption or signing algorithms supported for a particular KMS key, use the DescribeKey operation.

HTTP Status Code: 400

KeyUnavailableException

The request was rejected because the specified KMS key was not available. You can retry the request.

HTTP Status Code: 500

KMSInternalException

The request was rejected because an internal exception occurred. The request can be retried.

HTTP Status Code: 500

KMSInvalidStateException

The request was rejected because the state of the specified resource is not valid for this request.

This exceptions means one of the following:

- The key state of the KMS key is not compatible with the operation.

  To find the key state, use the DescribeKey operation. For more information about which key states are compatible with each AWS KMS operation, see Key states of AWS KMS keys in the AWS Key Management Service Developer Guide .

- For cryptographic operations on KMS keys in custom key stores, this exception represents a general failure with many possible causes. To identify the cause, see the error message that accompanies the exception.

HTTP Status Code: 400

NotFoundException

The request was rejected because the specified entity or resource could not be found.

HTTP Status Code: 400

Examples

The following examples are formatted for legibility.

Example Request

This example illustrates one usage of GenerateDataKeyWithoutPlaintext.

```
POST / HTTP/1.1
Host: kms.us-east-2.amazonaws.com
Content-Length: 50
X-Amz-Target: TrentService.GenerateDataKeyWithoutPlaintext
X-Amz-Date: 20161112T001941Z
Content-Type: application/x-amz-json-1.1
Authorization: AWS4-HMAC-SHA256\
 Credential=AKIAI44QH8DHBEXAMPLE/20161112/us-east-2/kms/aws4_request,\
 SignedHeaders=content-type;host;x-amz-date;x-amz-target,\
 Signature=c86e7fc0218461e537c0d06ac29d865d94dba6fbfad00a844f61200e651df483

{
  "KeyId": "alias/ExampleAlias",
  "KeySpec": "AES_256"
}
```

Example Response

This example illustrates one usage of GenerateDataKeyWithoutPlaintext.

```
HTTP/1.1 200 OK
Server: Server
```

```
Date: Sat, 12 Nov 2016 00:19:41 GMT
Content-Type: application/x-amz-json-1.1
Content-Length: 331
Connection: keep-alive
x-amzn-RequestId: b4ca7ee7-a86d-11e6-8a4e-2f341b963ed6

{
  "CiphertextBlob":
 "AQEDAHjRYf5WytIc0C857tFSnBaPn2F8DgfmThbJlGfR8P3WlwAAAH4wfAYJKoZIhvcNAQcGoG8wbQIBADBoBgkqhkiG9
ntdQTL16wQIBEIA7BE/3LB7F1meU8z4e1vEKBGZgXPwMvkZXbKnf3wxCD9lB4hU29lii4euOqxp8pESb
+7oCN9f1R75ac3s=",
  "KeyId": "arn:aws:kms:us-
east-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
  "KeyMaterialId": "0b7fd7ddbac6eef27907413567cad8c810e2883dc8a7534067a82ee1142fc1e6",
}
```

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface V2
- AWS SDK for .NET V4
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- AWS SDK for PHP V3
- AWS SDK for Python (Boto3)
- AWS SDK for Ruby V3

GenerateMac

Generates a hash-based message authentication code (HMAC) for a message using an HMAC KMS key and a MAC algorithm that the key supports. HMAC KMS keys and the HMAC algorithms that AWS KMS uses conform to industry standards defined in RFC 2104.

You can use value that GenerateMac returns in the VerifyMac operation to demonstrate that the original message has not changed. Also, because a secret key is used to create the hash, you can verify that the party that generated the hash has the required secret key. You can also use the raw result to implement HMAC-based algorithms such as key derivation functions. This operation is part of AWS KMS support for HMAC KMS keys. For details, see HMAC keys in AWS KMS in the AWS Key Management Service Developer Guide .

> **Note**
>
> Best practices recommend that you limit the time during which any signing mechanism, including an HMAC, is effective. This deters an attack where the actor uses a signed message to establish validity repeatedly or long after the message is superseded. HMAC tags do not include a timestamp, but you can include a timestamp in the token or message to help you detect when its time to refresh the HMAC.

The KMS key that you use for this operation must be in a compatible key state. For details, see Key states of AWS KMS keys in the *AWS Key Management Service Developer Guide*.

Cross-account use: Yes. To perform this operation with a KMS key in a different AWS account, specify the key ARN or alias ARN in the value of the `KeyId` parameter.

Required permissions: kms:GenerateMac (key policy)

Related operations: VerifyMac

Eventual consistency: The AWS KMS API follows an eventual consistency model. For more information, see AWS KMS eventual consistency.

Request Syntax

```
{
   "DryRun": boolean,
   "GrantTokens": [ "string" ],
```

```
   "KeyId": "string",
   "MacAlgorithm": "string",
   "Message": blob
}
```

Request Parameters

For information about the parameters that are common to all actions, see Common Parameters.

The request accepts the following data in JSON format.

> **Note**
>
> In the following list, the required parameters are described first.

KeyId

The HMAC KMS key to use in the operation. The MAC algorithm computes the HMAC for the message and the key as described in RFC 2104.

To identify an HMAC KMS key, use the DescribeKey operation and see the `KeySpec` field in the response.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 2048.

Required: Yes

MacAlgorithm

The MAC algorithm used in the operation.

The algorithm must be compatible with the HMAC KMS key that you specify. To find the MAC algorithms that your HMAC KMS key supports, use the DescribeKey operation and see the `MacAlgorithms` field in the `DescribeKey` response.

Type: String

Valid Values: `HMAC_SHA_224 | HMAC_SHA_256 | HMAC_SHA_384 | HMAC_SHA_512`

Required: Yes

Message

The message to be hashed. Specify a message of up to 4,096 bytes.

`GenerateMac` and VerifyMac do not provide special handling for message digests. If you generate an HMAC for a hash digest of a message, you must verify the HMAC of the same hash digest.

Type: Base64-encoded binary data object

Length Constraints: Minimum length of 1. Maximum length of 4096.

Required: Yes

DryRun

Checks if your request will succeed. `DryRun` is an optional parameter.

To learn more about how to use this parameter, see Testing your permissions in the *AWS Key Management Service Developer Guide*.

Type: Boolean

Required: No

GrantTokens

A list of grant tokens.

Use a grant token when your permission to call this operation comes from a new grant that has not yet achieved *eventual consistency*. For more information, see Grant token and Using a grant token in the *AWS Key Management Service Developer Guide*.

Type: Array of strings

Array Members: Minimum number of 0 items. Maximum number of 10 items.

Length Constraints: Minimum length of 1. Maximum length of 8192.

Required: No

Response Syntax

```
{
```

```
    "KeyId": "string",
    "Mac": blob,
    "MacAlgorithm": "string"
}
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

KeyId

The HMAC KMS key used in the operation.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 2048.

Mac

The hash-based message authentication code (HMAC) that was generated for the specified message, HMAC KMS key, and MAC algorithm.

This is the standard, raw HMAC defined in RFC 2104.

Type: Base64-encoded binary data object

Length Constraints: Minimum length of 1. Maximum length of 6144.

MacAlgorithm

The MAC algorithm that was used to generate the HMAC.

Type: String

Valid Values: `HMAC_SHA_224 | HMAC_SHA_256 | HMAC_SHA_384 | HMAC_SHA_512`

Errors

For information about the errors that are common to all actions, see Common Error Types.

DisabledException

The request was rejected because the specified KMS key is not enabled.

HTTP Status Code: 400

DryRunOperationException

The request was rejected because the DryRun parameter was specified.

HTTP Status Code: 400

InvalidGrantTokenException

The request was rejected because the specified grant token is not valid.

HTTP Status Code: 400

InvalidKeyUsageException

The request was rejected for one of the following reasons:

- The `KeyUsage` value of the KMS key is incompatible with the API operation.
- The encryption algorithm or signing algorithm specified for the operation is incompatible with the type of key material in the KMS key (`KeySpec`).

For encrypting, decrypting, re-encrypting, and generating data keys, the `KeyUsage` must be `ENCRYPT_DECRYPT`. For signing and verifying messages, the `KeyUsage` must be `SIGN_VERIFY`. For generating and verifying message authentication codes (MACs), the `KeyUsage` must be `GENERATE_VERIFY_MAC`. For deriving key agreement secrets, the `KeyUsage` must be `KEY_AGREEMENT`. For encapsulating and decapsulating shared secrets, the `KeyUsage` must be `ENCAPSULATE_DECAPSULATE`. To find the `KeyUsage` of a KMS key, use the DescribeKey operation.

To find the encryption or signing algorithms supported for a particular KMS key, use the DescribeKey operation.

HTTP Status Code: 400

KeyUnavailableException

The request was rejected because the specified KMS key was not available. You can retry the request.

HTTP Status Code: 500

KMSInternalException

The request was rejected because an internal exception occurred. The request can be retried.

HTTP Status Code: 500

KMSInvalidStateException

The request was rejected because the state of the specified resource is not valid for this request.

This exceptions means one of the following:

- The key state of the KMS key is not compatible with the operation.

  To find the key state, use the DescribeKey operation. For more information about which key states are compatible with each AWS KMS operation, see Key states of AWS KMS keys in the AWS Key Management Service Developer Guide .
- For cryptographic operations on KMS keys in custom key stores, this exception represents a general failure with many possible causes. To identify the cause, see the error message that accompanies the exception.

HTTP Status Code: 400

NotFoundException

The request was rejected because the specified entity or resource could not be found.

HTTP Status Code: 400

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface V2
- AWS SDK for .NET V4
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- AWS SDK for PHP V3
- AWS SDK for Python (Boto3)

- AWS SDK for Ruby V3

GenerateRandom

Returns a random byte string that is cryptographically secure.

You must use the `NumberOfBytes` parameter to specify the length of the random byte string. There is no default value for string length.

By default, the random byte string is generated in AWS KMS. To generate the byte string in the AWS CloudHSM cluster associated with an AWS CloudHSM key store, use the `CustomKeyStoreId` parameter.

`GenerateRandom` also supports AWS Nitro Enclaves, which provide an isolated compute environment in Amazon EC2. To call `GenerateRandom` for a Nitro enclave or NitroTPM, use the AWS Nitro Enclaves SDK or any AWS SDK. Use the `Recipient` parameter to provide the attestation document for the attested environment. Instead of plaintext bytes, the response includes the plaintext bytes encrypted under the public key from the attestation document (`CiphertextForRecipient`). For information about the interaction between AWS KMS and AWS Nitro Enclaves or AWS NitroTPM, see Cryptographic attestation support in AWS KMS in the *AWS Key Management Service Developer Guide*.

For more information about entropy and random number generation, see Entropy and random number generation in the *AWS Key Management Service Developer Guide*.

Cross-account use: Not applicable. `GenerateRandom` does not use any account-specific resources, such as KMS keys.

Required permissions: kms:GenerateRandom (IAM policy)

Eventual consistency: The AWS KMS API follows an eventual consistency model. For more information, see AWS KMS eventual consistency.

Request Syntax

```
{
   "CustomKeyStoreId": "string",
   "NumberOfBytes": number,
   "Recipient": {
      "AttestationDocument": blob,
      "KeyEncryptionAlgorithm": "string"
   }
```

```
}
```

Request Parameters

For information about the parameters that are common to all actions, see Common Parameters.

The request accepts the following data in JSON format.

> **Note**
>
> In the following list, the required parameters are described first.

CustomKeyStoreId

Generates the random byte string in the AWS CloudHSM cluster that is associated with the specified AWS CloudHSM key store. To find the ID of a custom key store, use the DescribeCustomKeyStores operation.

External key store IDs are not valid for this parameter. If you specify the ID of an external key store, `GenerateRandom` throws an `UnsupportedOperationException`.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 64.

Required: No

NumberOfBytes

The length of the random byte string. This parameter is required.

Type: Integer

Valid Range: Minimum value of 1. Maximum value of 1024.

Required: No

Recipient

A signed attestation document from an AWS Nitro enclave or NitroTPM, and the encryption algorithm to use with the public key in the attestation document. The only valid encryption algorithm is `RSAES_OAEP_SHA_256`.

This parameter supports the AWS Nitro Enclaves SDK or any AWS SDK for AWS Nitro Enclaves. It supports any AWS SDK for AWS NitroTPM.

When you use this parameter, instead of returning plaintext bytes, AWS KMS encrypts the plaintext bytes under the public key in the attestation document, and returns the resulting ciphertext in the `CiphertextForRecipient` field in the response. This ciphertext can be decrypted only with the private key in the attested environment. The `Plaintext` field in the response is null or empty.

For information about the interaction between AWS KMS and AWS Nitro Enclaves or AWS NitroTPM, see Cryptographic attestation support in AWS KMS in the *AWS Key Management Service Developer Guide*.

Type: RecipientInfo object

Required: No

Response Syntax

```
{
   "CiphertextForRecipient": blob,
   "Plaintext": blob
}
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

CiphertextForRecipient

The plaintext random bytes encrypted with the public key from the attestation document. This ciphertext can be decrypted only by using a private key from the attested environment.

This field is included in the response only when the `Recipient` parameter in the request includes a valid attestation document from an AWS Nitro enclave or NitroTPM. For information about the interaction between AWS KMS and AWS Nitro Enclaves or AWS NitroTPM, see Cryptographic attestation support in AWS KMS in the *AWS Key Management Service Developer Guide*.

Type: Base64-encoded binary data object

Length Constraints: Minimum length of 1. Maximum length of 6144.

Plaintext

The random byte string. When you use the HTTP API or the AWS CLI, the value is Base64-encoded. Otherwise, it is not Base64-encoded.

If the response includes the `CiphertextForRecipient` field, the `Plaintext` field is null or empty.

Type: Base64-encoded binary data object

Length Constraints: Minimum length of 1. Maximum length of 4096.

Errors

For information about the errors that are common to all actions, see Common Error Types.

CustomKeyStoreInvalidStateException

The request was rejected because of the `ConnectionState` of the custom key store. To get the `ConnectionState` of a custom key store, use the DescribeCustomKeyStores operation.

This exception is thrown under the following conditions:

- You requested the ConnectCustomKeyStore operation on a custom key store with a `ConnectionState` of `DISCONNECTING` or `FAILED`. This operation is valid for all other `ConnectionState` values. To reconnect a custom key store in a `FAILED` state, disconnect it (DisconnectCustomKeyStore), then connect it (`ConnectCustomKeyStore`).
- You requested the CreateKey operation in a custom key store that is not connected. This operations is valid only when the custom key store `ConnectionState` is `CONNECTED`.
- You requested the DisconnectCustomKeyStore operation on a custom key store with a `ConnectionState` of `DISCONNECTING` or `DISCONNECTED`. This operation is valid for all other `ConnectionState` values.
- You requested the UpdateCustomKeyStore or DeleteCustomKeyStore operation on a custom key store that is not disconnected. `UpdateCustomKeyStore` can be called on a custom key store in the `CONNECTED` state only to update `NewCustomKeyStoreName`. For all other properties, the custom key store `ConnectionState` must be `DISCONNECTED`.

- You requested the GenerateRandom operation in an AWS CloudHSM key store that is not connected. This operation is valid only when the AWS CloudHSM key store `ConnectionState` is `CONNECTED`.

HTTP Status Code: 400

CustomKeyStoreNotFoundException

The request was rejected because AWS KMS cannot find a custom key store with the specified key store name or ID.

HTTP Status Code: 400

DependencyTimeoutException

The system timed out while trying to fulfill the request. You can retry the request.

HTTP Status Code: 500

KMSInternalException

The request was rejected because an internal exception occurred. The request can be retried.

HTTP Status Code: 500

UnsupportedOperationException

The request was rejected because a specified parameter is not supported or a specified resource is not valid for this operation.

HTTP Status Code: 400

Examples

Example Request

The following example is formatted for legibility.

```
POST / HTTP/1.1
Host: kms.us-east-2.amazonaws.com
Content-Length: 21
X-Amz-Target: TrentService.GenerateRandom
X-Amz-Date: 20161114T215101Z
Content-Type: application/x-amz-json-1.1
```

```
Authorization: AWS4-HMAC-SHA256\
 Credential=AKIAI44QH8DHBEXAMPLE/20161114/us-east-2/kms/aws4_request,\
 SignedHeaders=content-type;host;x-amz-date;x-amz-target,\
 Signature=e3a0cfdbfb71fae5c89e422ad8322b6a44aed85bf68e3d11f3f315bbaa82ad22

{"NumberOfBytes": 32}
```

Example Response

This example illustrates one usage of GenerateRandom.

```
HTTP/1.1 200 OK
Server: Server
Date: Mon, 14 Nov 2016 21:51:02 GMT
Content-Type: application/x-amz-json-1.1
Content-Length: 60
Connection: keep-alive
x-amzn-RequestId: 6f79b0ad-aab4-11e6-971f-0f7b7e5b6782

{"Plaintext":"+Q2hxK6OBuU6K6ZIIBucFMCW2NJkhiSWDySSQyWp9zA="}
```

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface V2
- AWS SDK for .NET V4
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- AWS SDK for PHP V3
- AWS SDK for Python (Boto3)
- AWS SDK for Ruby V3

GetKeyLastUsage

Returns usage information about the last successful cryptographic operation performed with a specified KMS key, including the operation type, timestamp, and associated AWS CloudTrail event ID.

The `TrackingStartDate` in the `GetKeyLastUsage` response indicates the date from which AWS KMS began recording cryptographic activity for a given key. Use this value together with `KeyCreationDate` to understand the key's usage history:

- If the `KeyLastUsage` response element is *present*, the key has been used for a successful cryptographic operation since the `TrackingStartDate`. The response includes the operation type, timestamp, and associated CloudTrail event ID.
- If the `KeyLastUsage` response element is *empty* and `KeyCreationDate` is on or after `TrackingStartDate`, the key has not been used for a successful cryptographic operation since it was created.
- If the `KeyLastUsage` response element is *empty* and `KeyCreationDate` is before `TrackingStartDate`, there is no record of the key being used for a successful cryptographic operation since the `TrackingStartDate`. However, the key may have been used before tracking began. To determine whether the key was used before the `TrackingStartDate`, examine your past AWS CloudTrail logs.

For multi-Region KMS keys, primary and replica keys track last usage independently. Each key in a multi-Region key set maintains its own usage information.

The `ReEncrypt` operation uses two keys: a source key for decryption and a destination key for encryption. Usage information is recorded for both keys independently, each with the AWS CloudTrail event ID from the respective key owner's account.

> **Note**
>
> Do not use `GetKeyLastUsage` as the sole indicator when scheduling a key for deletion. Instead, first disable the key and monitor AWS CloudTrail for `DisabledException` entries, as there could be infrequent workflows that are dependent on the key. By looking for this exception, you can identify potential dependencies and workload failures before they occur. AWS CloudTrail remains the authoritative source for all API calls made to your key.

Cross-account use: No. You cannot perform this operation on a KMS key in a different AWS account.

Required permissions: kms:GetKeyLastUsage (key policy)

Related operations:

- DescribeKey
- DisableKey
- ScheduleKeyDeletion

Eventual consistency: The AWS KMS API follows an eventual consistency model. For more information, see AWS KMS eventual consistency.

Request Syntax

```
{
   "KeyId": "string"
}
```

Request Parameters

For information about the parameters that are common to all actions, see Common Parameters.

The request accepts the following data in JSON format.

> **Note**
>
> In the following list, the required parameters are described first.

KeyId

Identifies the KMS key to get usage information for. To specify a KMS key, use its key ID or key ARN. Alias names are not supported.

Specify the key ID or key ARN of the KMS key.

For example:

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

To get the key ID and key ARN for a KMS key, use ListKeys or DescribeKey.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 2048.

Required: Yes

Response Syntax

```
{
   "KeyCreationDate": number,
   "KeyId": "string",
   "KeyLastUsage": {
      "CloudTrailEventId": "string",
      "KmsRequestId": "string",
      "Operation": "string",
      "Timestamp": number
   },
   "TrackingStartDate": number
}
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

KeyCreationDate

The date and time when the KMS key was created.

Type: Timestamp

KeyId

The globally unique identifier for the KMS key.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 2048.

KeyLastUsage

Contains usage information about the last time the KMS key was used for a successful cryptographic operation. If the key has not been used since tracking began, this response element is an empty object.

Type: KeyLastUsageData object

TrackingStartDate

The date from which AWS KMS began recording cryptographic activity for this key, or the date the KMS key was created, whichever is later.

Type: Timestamp

Errors

For information about the errors that are common to all actions, see Common Error Types.

DependencyTimeoutException

The system timed out while trying to fulfill the request. You can retry the request.

HTTP Status Code: 500

InvalidArnException

The request was rejected because a specified ARN, or an ARN in a key policy, is not valid.

HTTP Status Code: 400

KMSInternalException

The request was rejected because an internal exception occurred. The request can be retried.

HTTP Status Code: 500

NotFoundException

The request was rejected because the specified entity or resource could not be found.

HTTP Status Code: 400

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface V2
- AWS SDK for .NET V4
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- AWS SDK for PHP V3
- AWS SDK for Python (Boto3)
- AWS SDK for Ruby V3

GetKeyPolicy

Gets a key policy attached to the specified KMS key.

A KMS key can have two key policies. You manage the `default` key policy. AWS Resource Access Manager (AWS RAM) manages the `AWS_RAM` key policy for KMS keys that you share through AWS RAM. You can use `GetKeyPolicy` to view the `AWS_RAM` key policy, but you can't change it with PutKeyPolicy. To get the `AWS_RAM` key policy, set the `PolicyName` parameter to `AWS_RAM`. If the KMS key doesn't have an `AWS_RAM` key policy, `GetKeyPolicy` returns a `NotFoundException`. For information about sharing KMS keys, see Sharing KMS keys with AWS RAM in the *AWS Key Management Service Developer Guide*. For information about AWS RAM managed permissions, see Managing permissions in AWS RAM in the *AWS RAM User Guide*.

Cross-account use: No. You cannot perform this operation on a KMS key in a different AWS account.

Required permissions: kms:GetKeyPolicy (key policy)

You can use the kms:PolicyName condition key to control access to this operation based on the value of the `PolicyName` parameter in the request. If the request doesn't include `PolicyName`, the value of `kms:PolicyName` is `default`.

Related operations: PutKeyPolicy

Eventual consistency: The AWS KMS API follows an eventual consistency model. For more information, see AWS KMS eventual consistency.

Request Syntax

```
{
   "KeyId": "string",
   "PolicyName": "string"
}
```

Request Parameters

For information about the parameters that are common to all actions, see Common Parameters.

The request accepts the following data in JSON format.

Note

In the following list, the required parameters are described first.

KeyId

Identifies the KMS key whose key policy you want to get.

Specify the key ID or key ARN of the KMS key.

For example:

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

To get the key ID and key ARN for a KMS key, use ListKeys or DescribeKey.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 2048.

Required: Yes

PolicyName

Specifies the name of the key policy. If no policy name is specified, the default value is `default`. Valid values are `default` and `AWS_RAM`. Use `AWS_RAM` to get the key policy that AWS RAM manages for a KMS key that you share through AWS RAM. If the KMS key doesn't have an `AWS_RAM` key policy, the request fails with a `NotFoundException`. To get the names of key policies, use ListKeyPolicies.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 128.

Pattern: `[\w]+`

Required: No

Response Syntax

```
{
   "Policy": "string",
   "PolicyName": "string"
}
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

Policy

A key policy document in JSON format.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 131072.

Pattern: `[\u0009\u000A\u000D\u0020-\u00FF]+`

PolicyName

The name of the key policy. Valid values are:

default

The key policy that you manage.

AWS_RAM

The key policy that AWS RAM manages for KMS keys that you share through AWS RAM.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 128.

Pattern: `[\w]+`

Errors

For information about the errors that are common to all actions, see Common Error Types.

DependencyTimeoutException

The system timed out while trying to fulfill the request. You can retry the request.

HTTP Status Code: 500

InvalidArnException

The request was rejected because a specified ARN, or an ARN in a key policy, is not valid.

HTTP Status Code: 400

KMSInternalException

The request was rejected because an internal exception occurred. The request can be retried.

HTTP Status Code: 500

KMSInvalidStateException

The request was rejected because the state of the specified resource is not valid for this request.

This exceptions means one of the following:

- The key state of the KMS key is not compatible with the operation.

  To find the key state, use the DescribeKey operation. For more information about which key states are compatible with each AWS KMS operation, see Key states of AWS KMS keys in the AWS Key Management Service Developer Guide .
- For cryptographic operations on KMS keys in custom key stores, this exception represents a general failure with many possible causes. To identify the cause, see the error message that accompanies the exception.

HTTP Status Code: 400

NotFoundException

The request was rejected because the specified entity or resource could not be found.

HTTP Status Code: 400

Examples

The following examples are formatted for legibility.

Example Request

This example illustrates one usage of GetKeyPolicy.

```
POST / HTTP/1.1
Host: kms.us-east-2.amazonaws.com
Content-Length: 74
X-Amz-Target: TrentService.GetKeyPolicy
X-Amz-Date: 20161114T225546Z
Content-Type: application/x-amz-json-1.1
Authorization: AWS4-HMAC-SHA256\
 Credential=AKIAI44QH8DHBEXAMPLE/20161114/us-east-2/kms/aws4_request,\
 SignedHeaders=content-type;host;x-amz-date;x-amz-target,\
 Signature=a88e20eebfbea3bf62d1512d0d2987e2d233becc7631a442237d3661df623a40

{
  "PolicyName": "default",
  "KeyId": "1234abcd-12ab-34cd-56ef-1234567890ab"
}
```

Example Response

This example illustrates one usage of GetKeyPolicy.

```
HTTP/1.1 200 OK
Server: Server
Date: Mon, 14 Nov 2016 22:55:47 GMT
Content-Type: application/x-amz-json-1.1
Content-Length: 326
Connection: keep-alive
x-amzn-RequestId: 7b105e7b-aabd-11e6-8039-3123b558b719

{
    "Policy":"{\n
        \"Version\" : \"2012-10-17\",\n
        \"Id\" : \"key-default-1\",\n
        \"Statement\" : [ {\n
            \"Sid\" : \"Enable IAM User Permissions\",\n
            \"Effect\" : \"Allow\",\n
            \"Principal\" : {\n
                \"AWS\" : \"arn:aws:iam::111122223333:root\"\n
            },\n
            \"Action\" : \"kms:*\",\n
```

```
            \"Resource\" : \"*\"\n
        } ]\n
    }",
    "PolicyName": "default"
}
```

Example – Request for the AWS_RAM key policy

This example illustrates one usage of GetKeyPolicy.

```
POST / HTTP/1.1
Host: kms.us-east-2.amazonaws.com
Content-Length: 80
X-Amz-Target: TrentService.GetKeyPolicy
X-Amz-Date: 20261117T193012Z
Content-Type: application/x-amz-json-1.1
Authorization: AWS4-HMAC-SHA256\
 Credential=AKIAI44QH8DHBEXAMPLE/20261117/us-east-2/kms/aws4_request,\
 SignedHeaders=content-type;host;x-amz-date;x-amz-target,\
 Signature=EXAMPLE

{
  "PolicyName": "AWS_RAM",
  "KeyId": "1234abcd-12ab-34cd-56ef-1234567890ab"
}
```

Example – Response for the AWS_RAM key policy

This example illustrates one usage of GetKeyPolicy.

```
HTTP/1.1 200 OK
Server: Server
Date: Tue, 17 Nov 2026 19:30:12 GMT
Content-Type: application/x-amz-json-1.1
Content-Length: 510
Connection: keep-alive
x-amzn-RequestId: 5b2e9c1d-8f3a-4e6b-9d7c-2a1f0e3b4c5d

{
    "Policy":"{\n
        \"Version\" : \"2012-10-17\",\n
        \"Id\" : \"a1b2c3d4-5678-90ab-cdef-EXAMPLE11111\",\n
```

```
        \"Statement\" : [ {\n
            \"Sid\" : \"ExampleKmsKeyPermission-external-principals\",\n
            \"Effect\" : \"Allow\",\n
            \"Principal\" : {\n
                \"AWS\" : \"arn:aws:iam::444455556666:root\"\n
            },\n
            \"Action\" : [ \"kms:DescribeKey\", \"kms:Encrypt\", \"kms:Decrypt\" ],\n
            \"Resource\" : \"arn:aws:kms:us-
east-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab\"\n
        } ]\n
    }",
    "PolicyName": "AWS_RAM"
}
```

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface V2
- AWS SDK for .NET V4
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- AWS SDK for PHP V3
- AWS SDK for Python (Boto3)
- AWS SDK for Ruby V3

GetKeyRotationStatus

Provides detailed information about the rotation status for a KMS key, including whether automatic rotation of the key material is enabled for the specified KMS key, the rotation period, and the next scheduled rotation date.

Automatic key rotation is supported only on symmetric encryption KMS keys. You cannot enable automatic rotation of asymmetric KMS keys, HMAC KMS keys, KMS keys with imported key material, or KMS keys in a custom key store. To enable or disable automatic rotation of a set of related multi-Region keys, set the property on the primary key.

You can enable (EnableKeyRotation) and disable automatic rotation (DisableKeyRotation) of the key material in customer managed KMS keys. Key material rotation of AWS managed KMS keys is not configurable. AWS KMS always rotates the key material in AWS managed KMS keys every year. The key rotation status for AWS managed KMS keys is always `true`.

You can perform on-demand (RotateKeyOnDemand) rotation of the key material in customer managed KMS keys, regardless of whether or not automatic key rotation is enabled. You can use GetKeyRotationStatus to identify the date and time that an in progress on-demand rotation was initiated. You can use ListKeyRotations to view the details of completed rotations.

> **Note**
>
> In May 2022, AWS KMS changed the rotation schedule for AWS managed keys from every three years to every year. For details, see EnableKeyRotation.

The KMS key that you use for this operation must be in a compatible key state. For details, see Key states of AWS KMS keys in the *AWS Key Management Service Developer Guide*.

- Disabled: The key rotation status does not change when you disable a KMS key. However, while the KMS key is disabled, AWS KMS does not rotate the key material. When you re-enable the KMS key, rotation resumes. If the key material in the re-enabled KMS key hasn't been rotated in one year, AWS KMS rotates it immediately, and every year thereafter. If it's been less than a year since the key material in the re-enabled KMS key was rotated, the KMS key resumes its prior rotation schedule.
- Pending deletion: While a KMS key is pending deletion, its key rotation status is `false` and AWS KMS does not rotate the key material. If you cancel the deletion, the original key rotation status returns to `true`.

Cross-account use: Yes. To perform this operation on a KMS key in a different AWS account, specify the key ARN in the value of the `KeyId` parameter.

Required permissions: kms:GetKeyRotationStatus (key policy)

Related operations:

- DisableKeyRotation
- EnableKeyRotation
- ListKeyRotations
- RotateKeyOnDemand

Eventual consistency: The AWS KMS API follows an eventual consistency model. For more information, see AWS KMS eventual consistency.

Request Syntax

```
{
   "KeyId": "string"
}
```

Request Parameters

For information about the parameters that are common to all actions, see Common Parameters.

The request accepts the following data in JSON format.

> **Note**
>
> In the following list, the required parameters are described first.

KeyId

Gets the rotation status for the specified KMS key.

Specify the key ID or key ARN of the KMS key. To specify a KMS key in a different AWS account, you must use the key ARN.

For example:

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

To get the key ID and key ARN for a KMS key, use ListKeys or DescribeKey.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 2048.

Required: Yes

Response Syntax

```
{
   "KeyId": "string",
   "KeyRotationEnabled": boolean,
   "NextRotationDate": number,
   "OnDemandRotationStartDate": number,
   "RotationPeriodInDays": number
}
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

KeyId

Identifies the specified symmetric encryption KMS key.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 2048.

KeyRotationEnabled

A Boolean value that specifies whether key rotation is enabled.

Type: Boolean

NextRotationDate

The next date that AWS KMS will automatically rotate the key material.

Type: Timestamp

OnDemandRotationStartDate

Identifies the date and time that an in progress on-demand rotation was initiated.

AWS KMS uses a background process to perform rotations. As a result, there might be a slight delay between initiating on-demand key rotation and the rotation's completion. Once the on-demand rotation is complete, AWS KMS removes this field from the response. You can use ListKeyRotations to view the details of the completed on-demand rotation.

Type: Timestamp

RotationPeriodInDays

The number of days between each automatic rotation. The default value is 365 days.

Type: Integer

Valid Range: Minimum value of 90. Maximum value of 2560.

Errors

For information about the errors that are common to all actions, see Common Error Types.

DependencyTimeoutException

The system timed out while trying to fulfill the request. You can retry the request.

HTTP Status Code: 500

InvalidArnException

The request was rejected because a specified ARN, or an ARN in a key policy, is not valid.

HTTP Status Code: 400

KMSInternalException

The request was rejected because an internal exception occurred. The request can be retried.

HTTP Status Code: 500

KMSInvalidStateException

The request was rejected because the state of the specified resource is not valid for this request.

This exceptions means one of the following:

- The key state of the KMS key is not compatible with the operation.

  To find the key state, use the DescribeKey operation. For more information about which key states are compatible with each AWS KMS operation, see Key states of AWS KMS keys in the AWS Key Management Service Developer Guide .
- For cryptographic operations on KMS keys in custom key stores, this exception represents a general failure with many possible causes. To identify the cause, see the error message that accompanies the exception.

HTTP Status Code: 400

NotFoundException

The request was rejected because the specified entity or resource could not be found.

HTTP Status Code: 400

UnsupportedOperationException

The request was rejected because a specified parameter is not supported or a specified resource is not valid for this operation.

HTTP Status Code: 400

Examples

Example Request

The following example is formatted for legibility.

```
POST / HTTP/1.1
Host: kms.us-east-2.amazonaws.com
Content-Length: 49
X-Amz-Target: TrentService.GetKeyRotationStatus
X-Amz-Date: 20161115T005817Z
```

```
Content-Type: application/x-amz-json-1.1
Authorization: AWS4-HMAC-SHA256\
 Credential=AKIAI44QH8DHBEXAMPLE/20161115/us-east-2/kms/aws4_request,\
 SignedHeaders=content-type;host;x-amz-date;x-amz-target,\
 Signature=282cb3a4a5d10684ff6c363300c34569a0707c4d503b88778e78cc51ea52f9be

{"KeyId": "1234abcd-12ab-34cd-56ef-1234567890ab"}
```

Example Response

This example illustrates one usage of GetKeyRotationStatus.

```
HTTP/1.1 200 OK
Server: Server
Date: Tue, 15 Nov 2016 00:58:18 GMT
Content-Type: application/x-amz-json-1.1
Content-Length: 28
Connection: keep-alive
x-amzn-RequestId: 98b59330-aace-11e6-aff0-8333261e2fbd

{
    "KeyId": "1234abcd-12ab-34cd-56ef-1234567890ab",
    "KeyRotationEnabled": true,
    "NextRotationDate": "2024-02-14T18:14:33.587000+00:00",
    "RotationPeriodInDays": 365
}
```

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface V2
- AWS SDK for .NET V4
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin

- AWS SDK for PHP V3
- AWS SDK for Python (Boto3)
- AWS SDK for Ruby V3

GetParametersForImport

Returns the public key and an import token you need to import or reimport key material for a KMS key.

By default, KMS keys are created with key material that AWS KMS generates. This operation supports Importing key material, an advanced feature that lets you generate and import the cryptographic key material for a KMS key.

Before calling `GetParametersForImport`, use the CreateKey operation with an `Origin` value of `EXTERNAL` to create a KMS key with no key material. You can import key material for a symmetric encryption KMS key, HMAC KMS key, asymmetric encryption KMS key, or asymmetric signing KMS key. You can also import key material into a multi-Region key of any supported type. However, you can't import key material into a KMS key in a custom key store. You can also use `GetParametersForImport` to get a public key and import token to reimport the original key material into a KMS key whose key material expired or was deleted.

`GetParametersForImport` returns the items that you need to import your key material.

- The public key (or "wrapping key") of an RSA key pair that KMS generates.

  You will use this public key to encrypt ("wrap") your key material while it's in transit to AWS KMS.
- A import token that ensures that AWS KMS can decrypt your key material and associate it with the correct KMS key.

The public key and its import token are permanently linked and must be used together. Each public key and import token set is valid for 24 hours. The expiration date and time appear in the `ParametersValidTo` field in the `GetParametersForImport` response. You cannot use an expired public key or import token in an ImportKeyMaterial request. If your key and token expire, send another `GetParametersForImport` request.

`GetParametersForImport` requires the following information:

- The key ID of the KMS key for which you are importing the key material.
- The key spec of the public key ("wrapping key") that you will use to encrypt your key material during import.
- The wrapping algorithm that you will use with the public key to encrypt your key material.

You can use the same or a different public key spec and wrapping algorithm each time you import or reimport the same key material.

The KMS key that you use for this operation must be in a compatible key state. For details, see Key states of AWS KMS keys in the *AWS Key Management Service Developer Guide*.

Cross-account use: No. You cannot perform this operation on a KMS key in a different AWS account.

Required permissions: kms:GetParametersForImport (key policy)

Related operations:

- ImportKeyMaterial
- DeleteImportedKeyMaterial

Eventual consistency: The AWS KMS API follows an eventual consistency model. For more information, see AWS KMS eventual consistency.

Request Syntax

```
{
   "KeyId": "string",
   "WrappingAlgorithm": "string",
   "WrappingKeySpec": "string"
}
```

Request Parameters

For information about the parameters that are common to all actions, see Common Parameters.

The request accepts the following data in JSON format.

> **Note**
>
> In the following list, the required parameters are described first.

KeyId

The identifier of the KMS key that will be associated with the imported key material. The `Origin` of the KMS key must be `EXTERNAL`.

All KMS key types are supported, including multi-Region keys. However, you cannot import key material into a KMS key in a custom key store.

Specify the key ID or key ARN of the KMS key.

For example:

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

To get the key ID and key ARN for a KMS key, use ListKeys or DescribeKey.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 2048.

Required: Yes

WrappingAlgorithm

The algorithm you will use with the RSA public key (`PublicKey`) in the response to protect your key material during import. For more information, see Select a wrapping algorithm in the *AWS Key Management Service Developer Guide*.

For RSA_AES wrapping algorithms, you encrypt your key material with an AES key that you generate, then encrypt your AES key with the RSA public key from AWS KMS. For RSAES wrapping algorithms, you encrypt your key material directly with the RSA public key from AWS KMS.

The wrapping algorithms that you can use depend on the type of key material that you are importing. To import an RSA private key, you must use an RSA_AES wrapping algorithm.

- **RSA_AES_KEY_WRAP_SHA_256** — Supported for wrapping RSA and ECC key material.
- **RSA_AES_KEY_WRAP_SHA_1** — Supported for wrapping RSA and ECC key material.
- **RSAES_OAEP_SHA_256** — Supported for all types of key material, except RSA key material (private key).

You cannot use the RSAES_OAEP_SHA_256 wrapping algorithm with the RSA_2048 wrapping key spec to wrap ECC_NIST_P521 key material.

- **RSAES_OAEP_SHA_1** — Supported for all types of key material, except RSA key material (private key).

  You cannot use the RSAES_OAEP_SHA_1 wrapping algorithm with the RSA_2048 wrapping key spec to wrap ECC_NIST_P521 key material.
- **RSAES_PKCS1_V1_5** (Deprecated) — As of October 10, 2023, AWS KMS does not support the RSAES_PKCS1_V1_5 wrapping algorithm.

Type: String

Valid Values: `RSAES_PKCS1_V1_5 | RSAES_OAEP_SHA_1 | RSAES_OAEP_SHA_256 | RSA_AES_KEY_WRAP_SHA_1 | RSA_AES_KEY_WRAP_SHA_256 | SM2PKE`

Required: Yes

WrappingKeySpec

The type of RSA public key to return in the response. You will use this wrapping key with the specified wrapping algorithm to protect your key material during import.

Use the longest RSA wrapping key that is practical.

You cannot use an RSA_2048 public key to directly wrap an ECC_NIST_P521 private key. Instead, use an RSA_AES wrapping algorithm or choose a longer RSA public key.

Type: String

Valid Values: `RSA_2048 | RSA_3072 | RSA_4096 | SM2`

Required: Yes

Response Syntax

```
{
   "ImportToken": blob,
   "KeyId": "string",
   "ParametersValidTo": number,
   "PublicKey": blob
```

```
}
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

ImportToken

The import token to send in a subsequent ImportKeyMaterial request.

Type: Base64-encoded binary data object

Length Constraints: Minimum length of 1. Maximum length of 6144.

KeyId

The Amazon Resource Name (key ARN) of the KMS key to use in a subsequent ImportKeyMaterial request. This is the same KMS key specified in the `GetParametersForImport` request.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 2048.

ParametersValidTo

The time at which the import token and public key are no longer valid. After this time, you cannot use them to make an ImportKeyMaterial request and you must send another `GetParametersForImport` request to get new ones.

Type: Timestamp

PublicKey

The public key to use to encrypt the key material before importing it with ImportKeyMaterial.

Type: Base64-encoded binary data object

Length Constraints: Minimum length of 1. Maximum length of 4096.

Errors

For information about the errors that are common to all actions, see Common Error Types.

DependencyTimeoutException

The system timed out while trying to fulfill the request. You can retry the request.

HTTP Status Code: 500

InvalidArnException

The request was rejected because a specified ARN, or an ARN in a key policy, is not valid.

HTTP Status Code: 400

KMSInternalException

The request was rejected because an internal exception occurred. The request can be retried.

HTTP Status Code: 500

KMSInvalidStateException

The request was rejected because the state of the specified resource is not valid for this request.

This exceptions means one of the following:

- The key state of the KMS key is not compatible with the operation.

  To find the key state, use the DescribeKey operation. For more information about which key states are compatible with each AWS KMS operation, see Key states of AWS KMS keys in the AWS Key Management Service Developer Guide .
- For cryptographic operations on KMS keys in custom key stores, this exception represents a general failure with many possible causes. To identify the cause, see the error message that accompanies the exception.

HTTP Status Code: 400

NotFoundException

The request was rejected because the specified entity or resource could not be found.

HTTP Status Code: 400

UnsupportedOperationException

The request was rejected because a specified parameter is not supported or a specified resource is not valid for this operation.

HTTP Status Code: 400

Examples

The following examples are formatted for legibility.

Example Request

This example illustrates one usage of GetParametersForImport.

```
POST / HTTP/1.1
Host: kms.us-east-2.amazonaws.com
Content-Length: 121
X-Amz-Target: TrentService.GetParametersForImport
X-Amz-Date: 20231130T225216Z
Content-Type: application/x-amz-json-1.1
Authorization: AWS4-HMAC-SHA256\
 Credential=AKIAI44QH8DHBEXAMPLE/20161130/us-east-2/kms/aws4_request,\
 SignedHeaders=content-type;host;x-amz-date;x-amz-target,\
 Signature=5bcc8e7669b6de719091ad27ae0145daa319f881010958208e960329341421d5

{
  "WrappingAlgorithm": "RSAES_OAEP_SHA_256",
  "KeyId": "1234abcd-12ab-34cd-56ef-1234567890ab",
  "WrappingKeySpec": "RSA_4096"
}
```

Example Response

This example illustrates one usage of GetParametersForImport.

```
HTTP/1.1 200 OK
Server: Server
Date: Wed, 30 Jul 2023 22:52:17 GMT
Content-Type: application/x-amz-json-1.1
Content-Length: 2892
Connection: keep-alive
x-amzn-RequestId: a46d61e0-b74f-11e6-b0c0-3343f53dee45

{
  "ImportToken": "AQECAHgybIx2X9LNs5ADpvmFm5Sv//
daUB9ZeCKoiJxmiw09YQAABrQwggawBgkqhkiG9w0BBwagggahMIIGnQIBADCCBpYGCSqGSIb3DQEHATAeBglghkgBZQMEA
U4Wg2Vw+RMAgEQgIIGZ/wOYGszlrjopP6BW63jlYYn
+gd7jpdpx0dxPmPC5Ka6uuUomx1yMKVdgtMiX85jHr8or7RoLISwsyQH+CRD33V
```

```
+pQs+Rm0+XkinHj5Zl371ibHytqM1DwhCs5FdQJM+8kLau7EXTcar7XLQj86DWJRj/
dQW0nDdkQXgXvz7GFWkbYs3IELvTAc5lHOLHgkXeoXom3NtHMvbR2V34tYwaT86gdira9Qj0FDouNaTesEOJN/
QjBedXcnuWumwOzK+w/OL+MD4tR8/
B1jDjeafRv7YSMxiADr2FsfDL0ELhgXhFVC0Wz42oM0jYnoYjZuXx6fQxEmADjBMPjk6W
+SFs4sWOuHs0U8npsWBNOnLAZPqXskqSuPZzb3XMG59s+2ZUcbeARQjYv97861ohWgwzjxur2+wSlaGNYAb
+Xh7EV34n2KSLuJ1lSrZrEWlU1Pato6zzN1x0VHJgU3sMCJMQz1uch8ZGHbI7vvBvvvqTJT/
+087IA8thTTCRLAYTjr81sSEofug71twBrhct3pzKswaNQVmWMptBe54HWiWWZz1peNuIAIJtX9qtNzeuYEJyqfVBera0B5
+E4AQcSin0AWERUK9LY3BNM2svFrrl2tPWURtUPokMVI0i4NLw2fsHtLw1CXqwjGuzEGKvRfiaat3WGzAtMao5sSFQz/
XSCB9Ab5OsddOTArBr/ShuX1WYuPIL2+zQP+gadWjAfTgmx9Q4K2MxQUpS72bqUJmfzXqpVi63sKL43tOwJ
+2Bt8Z5JA9xaPkPwiYE5q7dWL4J57cr+Ty/GLXAhat9xIUsTjG5E3FIHLywKiBwlVjH/T5FXxk
+T0TXV/61UPGaxPX2HkFTirq/D2Uhz45pFwwH46nbhJe9NoRodjot+uAblfuAqxz0YElCRt/
gIMr87l4AF7X48JHfvqmZAYGdhJ1bUhSw8VfTOPkHpUV2k6Eq9DvcSRDsww1FI5+fVf0ZpDEf0W2itRz5Hq
+cRkQL9EZqLICNF0QrhEuEJNBXf3oSckvS1tqPnHaRIRmG71BONqwc7fSU7zmXa
+O95GV3gIgfvnQ3HJy5EHR2dgkjQdP
+hfdw7BcC9NT7ZyO9XefAI5GEr623hrzn6yom4JIiyUjjCQPK8mS75rIgazvyTp0WQKpSSKeJOZswYLNgip8Xv/
UBcehAKwRL0QhbOGhUbZvoRNS8c1FbrCUlcBc4W4aWzA4e7cepqy38/jfwRoh0UvN/
bbaDh8FC+jZyXhyXSTIPvM25HVVrxsDbsN8LkCabokXFlkhiawm3PqVm6QgWWKcpR2Td+ty
+Bdl2tRmGHDsPcHN0WaUEq2AjE7kzL0dv7Jd9OemBNTZSlEoQ8U5+sKbvmSrtFvPIj7zWDpDT9bkZFHcCvwlIE6AflbgBS8
phBgaiRlDQdDmJmGD1yl+dxnIcoPs14xlcIwBdpw/M
+lvUuX8K4tqLMKzi1MOE0heBhGL0uEebYSkSQSUXUTTCk9hEkqslw0VXgwpgnGBXAOnVtYdUaqFMx5RIVxW471bnU0CYW5M
H4KrdRPdvevc8kTG6I8fdK/ArYCvTk/yYL3L6YZbeqbActUTADX0iBijX/T5QYz/
Dd4H1eX4abHV70CnxftxCHuLMnwR8DpJVnkouQAqb4N7Ap6JIYkvNKFWb8HBlygq5kKcg5dTMAMiPRz80qsQm/
IwGG9JVbKeyhqlKtQOIerspm8J99lcn5s0aB180LKrtXAaFD1AyO3nDZxB3I71QKvOulr1BZ6K4meBKkEw3VqW4PpmxmBKn
+2ytZAdDox9zLT7YW457esjUQC6zibfBwb8G97leh704m37Stq6Z752u46frBNSPQlypGuSbqCw1peKeqf/
AVehk+j8RKBegOQSCvEja4KPmQrayXVzu3h1tDktA1/Wj21ercJaW20fcZ1KQG/
GPHuScFgBsWawQf1spqKwZyHAHPaWZCymD9Fo2yHBHi+/ARPwM02iuqDLi9Tqv/g0=",
  "KeyId": "arn:aws:kms:us-
east-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
  "ParametersValidTo": 1.480632737044E9,
  "PublicKey":
 "MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAvH3Yj0wbkLEpUl95Cv1cJVjsVNSjwGq3tCLnzXfhVwVvmzGN8
+iSK341kr2kFTpINN7T1ZaX9vfXBdGR+VtkRKMWoHQeWzHrPZ+3irvpXNCKxGUxmPNsJSjPUhuSXT5+0VrY/
LEYLQ5lUTrhU6z5/OK0kzaCc66DXc5ipSloS4Xyg
+QcYSMxe9xuqO5HtzFImUSKBm1W6eDT6lHnSbpi7vXzNbIX7pWxKw9nmQvQIDAQAB"
}
```


See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface V2
- AWS SDK for .NET V4

- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- AWS SDK for PHP V3
- AWS SDK for Python (Boto3)
- AWS SDK for Ruby V3

GetPublicKey

Returns the public key of an asymmetric KMS key. Unlike the private key of a asymmetric KMS key, which never leaves AWS KMS unencrypted, callers with `kms:GetPublicKey` permission can download the public key of an asymmetric KMS key. You can share the public key to allow others to encrypt messages and verify signatures outside of AWS KMS. For information about asymmetric KMS keys, see Asymmetric KMS keys in the *AWS Key Management Service Developer Guide*.

You do not need to download the public key. Instead, you can use the public key within AWS KMS by calling the Encrypt, ReEncrypt, or Verify operations with the identifier of an asymmetric KMS key. When you use the public key within AWS KMS, you benefit from the authentication, authorization, and logging that are part of every AWS KMS operation. You also reduce of risk of encrypting data that cannot be decrypted. These features are not effective outside of AWS KMS.

To help you use the public key safely outside of AWS KMS, `GetPublicKey` returns important information about the public key in the response, including:

- KeySpec: The type of key material in the public key, such as `RSA_4096` or `ECC_NIST_P521`.
- KeyUsage: Whether the key is used for encryption, signing, or deriving a shared secret.
- EncryptionAlgorithms, KeyAgreementAlgorithms, or SigningAlgorithms: A list of the encryption algorithms, key agreement algorithms, or signing algorithms for the key.

Although AWS KMS cannot enforce these restrictions on external operations, it is crucial that you use this information to prevent the public key from being used improperly. For example, you can prevent a public signing key from being used encrypt data, or prevent a public key from being used with an encryption algorithm that is not supported by AWS KMS. You can also avoid errors, such as using the wrong signing algorithm in a verification operation.

To verify a signature outside of AWS KMS with an SM2 public key (China Regions only), you must specify the distinguishing ID. By default, AWS KMS uses `1234567812345678` as the distinguishing ID. For more information, see Offline verification with SM2 key pairs.

The KMS key that you use for this operation must be in a compatible key state. For details, see Key states of AWS KMS keys in the *AWS Key Management Service Developer Guide*.

Cross-account use: Yes. To perform this operation with a KMS key in a different AWS account, specify the key ARN or alias ARN in the value of the `KeyId` parameter.

Required permissions: kms:GetPublicKey (key policy)

Related operations: CreateKey

Eventual consistency: The AWS KMS API follows an eventual consistency model. For more information, see AWS KMS eventual consistency.

Request Syntax

```
{
   "GrantTokens": [ "string" ],
   "KeyId": "string"
}
```

Request Parameters

For information about the parameters that are common to all actions, see Common Parameters.

The request accepts the following data in JSON format.

> **Note**
>
> In the following list, the required parameters are described first.

KeyId

Identifies the asymmetric KMS key that includes the public key.

To specify a KMS key, use its key ID, key ARN, alias name, or alias ARN. When using an alias name, prefix it with `"alias/"`. To specify a KMS key in a different AWS account, you must use the key ARN or alias ARN.

For example:

- Key ID: `1234abcd-12ab-34cd-56ef-1234567890ab`
- Key ARN: `arn:aws:kms:us-east-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab`
- Alias name: `alias/ExampleAlias`
- Alias ARN: `arn:aws:kms:us-east-2:111122223333:alias/ExampleAlias`

To get the key ID and key ARN for a KMS key, use ListKeys or DescribeKey. To get the alias name and alias ARN, use ListAliases.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 2048.

Required: Yes

GrantTokens

A list of grant tokens.

Use a grant token when your permission to call this operation comes from a new grant that has not yet achieved *eventual consistency*. For more information, see Grant token and Using a grant token in the *AWS Key Management Service Developer Guide*.

Type: Array of strings

Array Members: Minimum number of 0 items. Maximum number of 10 items.

Length Constraints: Minimum length of 1. Maximum length of 8192.

Required: No

Response Syntax

```
{
   "CustomerMasterKeySpec": "string",
   "EncapsulationAlgorithms": [ "string" ],
   "EncryptionAlgorithms": [ "string" ],
   "KeyAgreementAlgorithms": [ "string" ],
   "KeyId": "string",
   "KeySpec": "string",
   "KeyUsage": "string",
   "PublicKey": blob,
   "SigningAlgorithms": [ "string" ]
}
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

CustomerMasterKeySpec

This parameter has been deprecated.

Instead, use the `KeySpec` field in the `GetPublicKey` response.

The `KeySpec` and `CustomerMasterKeySpec` fields have the same value. We recommend that you use the `KeySpec` field in your code. However, to avoid breaking changes, AWS KMS supports both fields.

Type: String

Valid Values: `RSA_2048 | RSA_3072 | RSA_4096 | ECC_NIST_P256 | ECC_NIST_P384 | ECC_NIST_P521 | ECC_SECG_P256K1 | SYMMETRIC_DEFAULT | HMAC_224 | HMAC_256 | HMAC_384 | HMAC_512 | SM2`

EncapsulationAlgorithms

The key encapsulation algorithm used to establish a shared secret. This field is present only when the KMS key has a `KeyUsage` value of `ENCAPSULATE_DECAPSULATE`.

Type: Array of strings

Valid Values: `ML_KEM`

EncryptionAlgorithms

The encryption algorithms that AWS KMS supports for this key.

This information is critical. If a public key encrypts data outside of AWS KMS by using an unsupported encryption algorithm, the ciphertext cannot be decrypted.

This field appears in the response only when the `KeyUsage` of the public key is `ENCRYPT_DECRYPT`.

Type: Array of strings

Valid Values: `SYMMETRIC_DEFAULT | RSAES_OAEP_SHA_1 | RSAES_OAEP_SHA_256 | SM2PKE`

KeyAgreementAlgorithms

The key agreement algorithm used to derive a shared secret. This field is present only when the KMS key has a `KeyUsage` value of `KEY_AGREEMENT`.

Type: Array of strings

Valid Values: ECDH

KeyId

The Amazon Resource Name (key ARN) of the asymmetric KMS key from which the public key was downloaded.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 2048.

KeySpec

The type of the of the public key that was downloaded.

Type: String

Valid Values: RSA_2048 | RSA_3072 | RSA_4096 | ECC_NIST_P256 | ECC_NIST_P384 | ECC_NIST_P521 | ECC_SECG_P256K1 | SYMMETRIC_DEFAULT | HMAC_224 | HMAC_256 | HMAC_384 | HMAC_512 | SM2 | ML_DSA_44 | ML_DSA_65 | ML_DSA_87 | ECC_NIST_EDWARDS25519 | ML_KEM_512 | ML_KEM_768 | ML_KEM_1024

KeyUsage

The permitted use of the public key. Valid values for asymmetric key pairs are ENCRYPT_DECRYPT, SIGN_VERIFY, and KEY_AGREEMENT.

This information is critical. For example, if a public key with SIGN_VERIFY key usage encrypts data outside of AWS KMS, the ciphertext cannot be decrypted.

Type: String

Valid Values: SIGN_VERIFY | ENCRYPT_DECRYPT | GENERATE_VERIFY_MAC | KEY_AGREEMENT | ENCAPSULATE_DECAPSULATE

PublicKey

The exported public key.

The value is a DER-encoded X.509 public key, also known as SubjectPublicKeyInfo (SPKI), as defined in RFC 5280. When you use the HTTP API or the AWS CLI, the value is Base64-encoded. Otherwise, it is not Base64-encoded.

Type: Base64-encoded binary data object

Length Constraints: Minimum length of 1. Maximum length of 8192.

SigningAlgorithms

The signing algorithms that AWS KMS supports for this key.

This field appears in the response only when the `KeyUsage` of the public key is `SIGN_VERIFY`.

Type: Array of strings

Valid Values: `RSASSA_PSS_SHA_256 | RSASSA_PSS_SHA_384 | RSASSA_PSS_SHA_512 | RSASSA_PKCS1_V1_5_SHA_256 | RSASSA_PKCS1_V1_5_SHA_384 | RSASSA_PKCS1_V1_5_SHA_512 | ECDSA_SHA_256 | ECDSA_SHA_384 | ECDSA_SHA_512 | SM2DSA | ML_DSA_SHAKE_256 | ED25519_SHA_512 | ED25519_PH_SHA_512`

Errors

For information about the errors that are common to all actions, see Common Error Types.

DependencyTimeoutException

The system timed out while trying to fulfill the request. You can retry the request.

HTTP Status Code: 500

DisabledException

The request was rejected because the specified KMS key is not enabled.

HTTP Status Code: 400

InvalidArnException

The request was rejected because a specified ARN, or an ARN in a key policy, is not valid.

HTTP Status Code: 400

InvalidGrantTokenException

The request was rejected because the specified grant token is not valid.

HTTP Status Code: 400

InvalidKeyUsageException

The request was rejected for one of the following reasons:

- The `KeyUsage` value of the KMS key is incompatible with the API operation.
- The encryption algorithm or signing algorithm specified for the operation is incompatible with the type of key material in the KMS key (`KeySpec`).

For encrypting, decrypting, re-encrypting, and generating data keys, the `KeyUsage` must be `ENCRYPT_DECRYPT`. For signing and verifying messages, the `KeyUsage` must be `SIGN_VERIFY`. For generating and verifying message authentication codes (MACs), the `KeyUsage` must be `GENERATE_VERIFY_MAC`. For deriving key agreement secrets, the `KeyUsage` must be `KEY_AGREEMENT`. For encapsulating and decapsulating shared secrets, the `KeyUsage` must be `ENCAPSULATE_DECAPSULATE`. To find the `KeyUsage` of a KMS key, use the DescribeKey operation.

To find the encryption or signing algorithms supported for a particular KMS key, use the DescribeKey operation.

HTTP Status Code: 400

KeyUnavailableException

The request was rejected because the specified KMS key was not available. You can retry the request.

HTTP Status Code: 500

KMSInternalException

The request was rejected because an internal exception occurred. The request can be retried.

HTTP Status Code: 500

KMSInvalidStateException

The request was rejected because the state of the specified resource is not valid for this request.

This exceptions means one of the following:

- The key state of the KMS key is not compatible with the operation.

  To find the key state, use the DescribeKey operation. For more information about which key states are compatible with each AWS KMS operation, see Key states of AWS KMS keys in the AWS Key Management Service Developer Guide .

- For cryptographic operations on KMS keys in custom key stores, this exception represents a general failure with many possible causes. To identify the cause, see the error message that accompanies the exception.

HTTP Status Code: 400

NotFoundException

The request was rejected because the specified entity or resource could not be found.

HTTP Status Code: 400

UnsupportedOperationException

The request was rejected because a specified parameter is not supported or a specified resource is not valid for this operation.

HTTP Status Code: 400

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface V2
- AWS SDK for .NET V4
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- AWS SDK for PHP V3
- AWS SDK for Python (Boto3)
- AWS SDK for Ruby V3

ImportKeyMaterial

Imports or reimports key material into an existing KMS key that was created without key material. You can also use this operation to set or update the expiration model and expiration date of the imported key material.

By default, AWS KMS creates KMS keys with key material that it generates. You can also generate and import your own key material. For more information about importing key material, see Importing key material.

For asymmetric and HMAC keys, you cannot change the key material after the initial import. You can import multiple key materials into symmetric encryption keys and rotate the key material on demand using `RotateKeyOnDemand`.

You can import new key materials into multi-Region symmetric encryption keys. To do so, you must import the new key material into the primary Region key. Then you can import the same key materials into the replica Region keys. You cannot directly import new key material into the replica Region keys.

To import new key material for a multi-Region symmetric key, you'll need to complete the following:

1. Call `ImportKeyMaterial` on the primary Region key with the `ImportType`set to `NEW_KEY_MATERIAL`.
2. Call `ImportKeyMaterial` on the replica Region key with the `ImportType` set to `EXISTING_KEY_MATERIAL` using the same key material imported to the primary Region key. You must do this for every replica Region key before you can perform the RotateKeyOnDemand operation on the primary Region key.

After you import key material, you can reimport the same key material into that KMS key or, if the key supports on-demand rotation, import new key material. You can use the `ImportType` parameter to indicate whether you are importing new key material or re-importing previously imported key material. You might reimport key material to replace key material that expired or key material that you deleted. You might also reimport key material to change the expiration model or expiration date of the key material.

Each time you import key material into AWS KMS, you can determine whether (`ExpirationModel`) and when (`ValidTo`) the key material expires. To change the expiration of

your key material, you must import it again, either by calling `ImportKeyMaterial` or using the import features of the AWS KMS console.

Before you call `ImportKeyMaterial`, complete these steps:

- Create or identify a KMS key with `EXTERNAL` origin, which indicates that the KMS key is designed for imported key material.

  To create a new KMS key for imported key material, call the CreateKey operation with an `Origin` value of `EXTERNAL`. You can create a symmetric encryption KMS key, HMAC KMS key, asymmetric encryption KMS key, asymmetric key agreement key, or asymmetric signing KMS key. You can also import key material into a multi-Region key of any supported type. However, you can't import key material into a KMS key in a custom key store.
- Call the GetParametersForImport operation to get a public key and import token set for importing key material.
- Use the public key in the GetParametersForImport response to encrypt your key material.

Then, in an `ImportKeyMaterial` request, you submit your encrypted key material and import token. When calling this operation, you must specify the following values:

- The key ID or key ARN of the KMS key to associate with the imported key material. Its `Origin` must be `EXTERNAL` and its `KeyState` must be `PendingImport` or `Enabled`. You cannot perform this operation on a KMS key in a custom key store, or on a KMS key in a different AWS account. To get the `Origin` and `KeyState` of a KMS key, call DescribeKey.
- The encrypted key material.
- The import token that GetParametersForImport returned. You must use a public key and token from the same `GetParametersForImport` response.
- Whether the key material expires (`ExpirationModel`) and, if so, when (`ValidTo`). For help with this choice, see Setting an expiration time in the *AWS Key Management Service Developer Guide*.

  If you set an expiration date, AWS KMS deletes the key material from the KMS key on the specified date, making the KMS key unusable. To use the KMS key in cryptographic operations again, you must reimport the same key material. However, you can delete and reimport the key material at any time, including before the key material expires. Each time you reimport, you can eliminate or reset the expiration time.

When this operation is successful, the state of the KMS key changes to `Enabled`, and you can use the KMS key in cryptographic operations. For symmetric encryption keys, you will need to import all of the key materials associated with the KMS key to change its state to `Enabled`. Use the `ListKeyRotations` operation to list the ID and import state of each key material associated with a KMS key.

If this operation fails, use the exception to help determine the problem. If the error is related to the key material, the import token, or wrapping key, use GetParametersForImport to get a new public key and import token for the KMS key and repeat the import procedure. For help, see Create a KMS key with imported key material in the *AWS Key Management Service Developer Guide*.

The KMS key that you use for this operation must be in a compatible key state. For details, see Key states of AWS KMS keys in the *AWS Key Management Service Developer Guide*.

Cross-account use: No. You cannot perform this operation on a KMS key in a different AWS account.

Required permissions: kms:ImportKeyMaterial (key policy)

Related operations:

- DeleteImportedKeyMaterial
- GetParametersForImport
- ListKeyRotations
- RotateKeyOnDemand

Eventual consistency: The AWS KMS API follows an eventual consistency model. For more information, see AWS KMS eventual consistency.

Request Syntax

```
{
   "EncryptedKeyMaterial": blob,
   "ExpirationModel": "string",
   "ImportToken": blob,
   "ImportType": "string",
   "KeyId": "string",
   "KeyMaterialDescription": "string",
   "KeyMaterialId": "string",
   "ValidTo": number
```

```
}
```

Request Parameters

For information about the parameters that are common to all actions, see Common Parameters.

The request accepts the following data in JSON format.

> **Note**
>
> In the following list, the required parameters are described first.

EncryptedKeyMaterial

The encrypted key material to import. The key material must be encrypted under the public wrapping key that GetParametersForImport returned, using the wrapping algorithm that you specified in the same `GetParametersForImport` request.

Type: Base64-encoded binary data object

Length Constraints: Minimum length of 1. Maximum length of 6144.

Required: Yes

ImportToken

The import token that you received in the response to a previous GetParametersForImport request. It must be from the same response that contained the public key that you used to encrypt the key material.

Type: Base64-encoded binary data object

Length Constraints: Minimum length of 1. Maximum length of 6144.

Required: Yes

KeyId

The identifier of the KMS key that will be associated with the imported key material. This must be the same KMS key specified in the `KeyID` parameter of the corresponding GetParametersForImport request. The `Origin` of the KMS key must be `EXTERNAL` and its `KeyState` must be `PendingImport`.

The KMS key can be a symmetric encryption KMS key, HMAC KMS key, asymmetric encryption KMS key, or asymmetric signing KMS key, including a multi-Region key of any supported type. You cannot perform this operation on a KMS key in a custom key store, or on a KMS key in a different AWS account.

Specify the key ID or key ARN of the KMS key.

For example:

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

To get the key ID and key ARN for a KMS key, use ListKeys or DescribeKey.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 2048.

Required: Yes

ExpirationModel

Specifies whether the key material expires. The default is `KEY_MATERIAL_EXPIRES`. For help with this choice, see Setting an expiration time in the *AWS Key Management Service Developer Guide*.

When the value of `ExpirationModel` is `KEY_MATERIAL_EXPIRES`, you must specify a value for the `ValidTo` parameter. When value is `KEY_MATERIAL_DOES_NOT_EXPIRE`, you must omit the `ValidTo` parameter.

You cannot change the `ExpirationModel` or `ValidTo` values for the current import after the request completes. To change either value, you must reimport the key material.

Type: String

Valid Values: `KEY_MATERIAL_EXPIRES | KEY_MATERIAL_DOES_NOT_EXPIRE`

Required: No

ImportType

Indicates whether the key material being imported is previously associated with this KMS key or not. This parameter is optional and only usable with symmetric encryption keys. If no

key material has ever been imported into the KMS key, and this parameter is omitted, the parameter defaults to `NEW_KEY_MATERIAL`. After the first key material is imported, if this parameter is omitted then the parameter defaults to `EXISTING_KEY_MATERIAL`.

For multi-Region keys, you must first import new key material into the primary Region key. You should use the `NEW_KEY_MATERIAL` import type when importing key material into the primary Region key. Then, you can import the same key material into the replica Region key. The import type for the replica Region key should be `EXISTING_KEY_MATERIAL`.

Type: String

Valid Values: `NEW_KEY_MATERIAL | EXISTING_KEY_MATERIAL`

Required: No

KeyMaterialDescription

Description for the key material being imported. This parameter is optional and only usable with symmetric encryption keys. If you do not specify a key material description, AWS KMS retains the value you specified when you last imported the same key material into this KMS key.

Type: String

Length Constraints: Minimum length of 0. Maximum length of 256.

Pattern: `^[a-zA-Z0-9:/_\s.-]+$`

Required: No

KeyMaterialId

Identifies the key material being imported. This parameter is optional and only usable with symmetric encryption keys. You cannot specify a key material ID with `ImportType` set to `NEW_KEY_MATERIAL`. Whenever you import key material into a symmetric encryption key, AWS KMS assigns a unique identifier to the key material based on the KMS key ID and the imported key material. When you re-import key material with a specified key material ID, AWS KMS:

- Computes the identifier for the key material
- Matches the computed identifier against the specified key material ID
- Verifies that the key material ID is already associated with the KMS key

To get the list of key material IDs associated with a KMS key, use ListKeyRotations.

Type: String

Length Constraints: Fixed length of 64.

Pattern: `^[a-f0-9]+$`

Required: No

ValidTo

The date and time when the imported key material expires. This parameter is required when the value of the `ExpirationModel` parameter is `KEY_MATERIAL_EXPIRES`. Otherwise it is not valid.

The value of this parameter must be a future date and time. The maximum value is 365 days from the request date.

When the key material expires, AWS KMS deletes the key material from the KMS key. Without its key material, the KMS key is unusable. To use the KMS key in cryptographic operations, you must reimport the same key material.

You cannot change the `ExpirationModel` or `ValidTo` values for the current import after the request completes. To change either value, you must delete (DeleteImportedKeyMaterial) and reimport the key material.

Type: Timestamp

Required: No

Response Syntax

```
{
   "KeyId": "string",
   "KeyMaterialId": "string"
}
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

KeyId

The Amazon Resource Name (key ARN) of the KMS key into which key material was imported.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 2048.

KeyMaterialId

Identifies the imported key material.

Type: String

Length Constraints: Fixed length of 64.

Pattern: `^[a-f0-9]+$`

Errors

For information about the errors that are common to all actions, see Common Error Types.

DependencyTimeoutException

The system timed out while trying to fulfill the request. You can retry the request.

HTTP Status Code: 500

ExpiredImportTokenException

The request was rejected because the specified import token is expired. Use GetParametersForImport to get a new import token and public key, use the new public key to encrypt the key material, and then try the request again.

HTTP Status Code: 400

IncorrectKeyMaterialException

The request was rejected because the key material in the request is, expired, invalid, or does not meet expectations. For example, it is not the same key material that was previously imported or AWS KMS expected new key material but the key material being imported is already associated with the KMS key.

HTTP Status Code: 400

InvalidArnException

The request was rejected because a specified ARN, or an ARN in a key policy, is not valid.

HTTP Status Code: 400

InvalidCiphertextException

From the Decrypt or ReEncrypt operation, the request was rejected because the specified ciphertext, or additional authenticated data incorporated into the ciphertext, such as the encryption context, is corrupted, missing, or otherwise invalid.

From the ImportKeyMaterial operation, the request was rejected because AWS KMS could not decrypt the encrypted (wrapped) key material.

From the Decapsulate operation, the request was rejected because the length of the `SharedSecretCiphertextBlob` is inconsistent with the specified ML-KEM key spec.

HTTP Status Code: 400

InvalidImportTokenException

The request was rejected because the provided import token is invalid or is associated with a different KMS key.

HTTP Status Code: 400

KMSInternalException

The request was rejected because an internal exception occurred. The request can be retried.

HTTP Status Code: 500

KMSInvalidStateException

The request was rejected because the state of the specified resource is not valid for this request.

This exceptions means one of the following:

- The key state of the KMS key is not compatible with the operation.

  To find the key state, use the DescribeKey operation. For more information about which key states are compatible with each AWS KMS operation, see Key states of AWS KMS keys in the AWS Key Management Service Developer Guide .

- For cryptographic operations on KMS keys in custom key stores, this exception represents a general failure with many possible causes. To identify the cause, see the error message that accompanies the exception.

HTTP Status Code: 400

NotFoundException

The request was rejected because the specified entity or resource could not be found.

HTTP Status Code: 400

UnsupportedOperationException

The request was rejected because a specified parameter is not supported or a specified resource is not valid for this operation.

HTTP Status Code: 400

Examples

Example Request

The following example is formatted for legibility.

```
POST / HTTP/1.1
Host: kms.us-east-2.amazonaws.com
Content-Length: 2835
X-Amz-Target: TrentService.ImportKeyMaterial
X-Amz-Date: 20161201T212609Z
Content-Type: application/x-amz-json-1.1
Authorization: AWS4-HMAC-SHA256\
 Credential=AKIAI44QH8DHBEXAMPLE/20161201/us-east-2/kms/aws4_request,\
 SignedHeaders=content-type;host;x-amz-date;x-amz-target,\
 Signature=dda4e269d4fd93decf1401aeb651e49c206c412c609141f6c743f146e1afb4e3

{
  "ExpirationModel": "KEY_MATERIAL_DOES_NOT_EXPIRE",
  "KeyId": "1234abcd-12ab-34cd-56ef-1234567890ab",
  "ImportToken": "AQECAHgybIx2X9LNs5ADpvmFm5Sv//
daUB9ZeCKoiJxmiw09YQAABrQwggawBgkqhkiG9w0BBwagggahMIIGnQIBADCCBpYGCSqGSIb3DQEHATAeBglghkgBZQMEA
U4Wg2Vw+RMAgEQgIIGZ/wOYGszlrjopP6BW63jlYYn
+gd7jpdpx0dxPmPC5Ka6uuUomx1yMKVdgtMiX85jHr8or7RoLISwsyQH+CRD33V
```

```
+pQs+Rm0+XkinHj5Zl371ibHytqM1DwhCs5FdQJM+8kLau7EXTcar7XLQj86DWJRj/
dQW0nDdkQXgXvz7GFWkbYs3IELvTAc5lHOLHgkXeoXom3NtHMvbR2V34tYwaT86gdira9Qj0FDouNaTesEOJN/
QjBedXcnuWumwOzK+w/OL+MD4tR8/
B1jDjeafRv7YSMxiADr2FsfDL0ELhgXhFVC0Wz42oM0jYnoYjZuXx6fQxEmADjBMPjk6W
+SFs4sWOuHs0U8npsWBNOnLAZPqXskqSuPZzb3XMG59s+2ZUcbeARQjYv97861ohWgwzjxur2+wSlaGNYAb
+Xh7EV34n2KSLuJ1lSrZrEWlU1Pato6zzN1x0VHJgU3sMCJMQz1uch8ZGHbI7vvBvvvqTJT/
+087IA8thTTCRLAYTjr81sSEofug71twBrhct3pzKswaNQVmWMptBe54HWiWWZz1peNuIAIJtX9qtNzeuYEJyqfVBera0B5
+E4AQcSin0AWERUK9LY3BNM2svFrrl2tPWURtUPokMVI0i4NLw2fsHtLw1CXqwjGuzEGKvRfiaat3WGzAtMao5sSFQz/
XSCB9Ab5OsddOTArBr/ShuX1WYuPIL2+zQP+gadWjAfTgmx9Q4K2MxQUpS72bqUJmfzXqpVi63sKL43tOwJ
+2Bt8Z5JA9xaPkPwiYE5q7dWL4J57cr+Ty/GLXAhat9xIUsTjG5E3FIHLywKiBwlVjH/T5FXxk
+T0TXV/61UPGaxPX2HkFTirq/D2Uhz45pFwwH46nbhJe9NoRodjot+uAblfuAqxz0YElCRt/
gIMr87l4AF7X48JHfvqmZAYGdhJ1bUhSw8VfTOPkHpUV2k6Eq9DvcSRDsww1FI5+fVf0ZpDEf0W2itRz5Hq
+cRkQL9EZqLICNF0QrhEuEJNBXf3oSckvS1tqPnHaRIRmG71BONqwc7fSU7zmXa
+O95GV3gIgfvnQ3HJy5EHR2dgkjQdP
+hfdw7BcC9NT7ZyO9XefAI5GEr623hrzn6yom4JIiyUjjCQPK8mS75rIgazvyTp0WQKpSSKeJOZswYLNgip8Xv/
UBcehAKwRL0QhbOGhUbZvoRNS8c1FbrCUlcBc4W4aWzA4e7cepqy38/jfwRoh0UvN/
bbaDh8FC+jZyXhyXSTIPvM25HVVrxsDbsN8LkCabokXFlkhiawm3PqVm6QgWWKcpR2Td+ty
+Bdl2tRmGHDsPcHN0WaUEq2AjE7kzL0dv7Jd9OemBNTZSlEoQ8U5+sKbvmSrtFvPIj7zWDpDT9bkZFHcCvwlIE6AflbgBS8
phBgaiRlDQdDmJmGD1yl+dxnIcoPs14xlcIwBdpw/M
+lvUuX8K4tqLMKzi1MOE0heBhGL0uEebYSkSQSUXUTTCk9hEkqslw0VXgwpgnGBXAOnVtYdUaqFMx5RIVxW471bnU0CYW5M
H4KrdRPdvevc8kTG6I8fdK/ArYCvTk/yYL3L6YZbeqbActUTADX0iBijX/T5QYz/
Dd4H1eX4abHV70CnxftxCHuLMnwR8DpJVnkouQAqb4N7Ap6JIYkvNKFWb8HBlygq5kKcg5dTMAMiPRz80qsQm/
IwGG9JVbKeyhqlKtQOIerspm8J99lcn5s0aB180LKrtXAaFD1AyO3nDZxB3I71QKvOulr1BZ6K4meBKkEw3VqW4PpmxmBKn
+2ytZAdDox9zLT7YW457esjUQC6zibfBwb8G97leh704m37Stq6Z752u46frBNSPQlypGuSbqCw1peKeqf/
AVehk+j8RKBegOQSCvEja4KPmQrayXVzu3h1tDktA1/Wj21ercJaW20fcZ1KQG/
GPHuScFgBsWawQf1spqKwZyHAHPaWZCymD9Fo2yHBHi+/ARPwM02iuqDLi9Tqv/g0=",
  "EncryptedKeyMaterial": "CubeyZ4cm/xMEA0UG5jP1iBzh/0E+uUg407JDcXhIC+iuMm
+wPgITaEby+Y3nM/e6gjUls5vy9TdBRFv4+JtksvB5hW4Znb2lUQhTUv+SSAZpaI14kAgTq/
jC2GTLkaC6Vf5zJx2xaLrOKGV2Xu4YgONIGslubHNffTC3aL/YBJ/FXTXaVu7rS2phOFCrZ
+ATittS03w4DiCVoNwo2v0QE0+dVoUNjXNQC1veWxhPlC7FezfK7AIsBSSXotJfANxRkybg8KcmkSoYdzr3N0L0v7oMorgb
PzphK6RWJGJig4tk+lxUT8hV7xiLkFskGjIHFmp6Xbon8w=="
}
```

Example Response

This example illustrates one usage of ImportKeyMaterial.

```
HTTP/1.1 200 OK
Server: Server
Date: Thu, 01 Dec 2016 21:26:10 GMT
Content-Type: application/x-amz-json-1.1
Content-Length: 2
Connection: keep-alive
```

```
x-amzn-RequestId: c72fb6ff-b80c-11e6-ae07-61b14fe11739

{
  "KeyId": "1234abcd-12ab-34cd-56ef-1234567890ab",
  "KeyMaterialId": "0b7fd7ddbac6eef27907413567cad8c810e2883dc8a7534067a82ee1142fc1e6",
}
```

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface V2
- AWS SDK for .NET V4
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- AWS SDK for PHP V3
- AWS SDK for Python (Boto3)
- AWS SDK for Ruby V3

ListAliases

Gets a list of aliases in the caller's AWS account and region. For more information about aliases, see CreateAlias.

By default, the `ListAliases` operation returns all aliases in the account and region. To get only the aliases associated with a particular KMS key, use the `KeyId` parameter.

The `ListAliases` response can include aliases that you created and associated with your customer managed keys, and aliases that AWS created and associated with AWS managed keys in your account. You can recognize AWS aliases because their names have the format `aws/<service-name>`, such as `aws/dynamodb`.

The response might also include aliases that have no `TargetKeyId` field. These are predefined aliases that AWS has created but has not yet associated with a KMS key. Aliases that AWS creates in your account, including predefined aliases, do not count against your AWS KMS aliases quota.

Cross-account use: No. `ListAliases` does not return aliases in other AWS accounts.

Required permissions: kms:ListAliases (IAM policy)

For details, see Controlling access to aliases in the *AWS Key Management Service Developer Guide*.

Related operations:

- CreateAlias
- DeleteAlias
- UpdateAlias

Eventual consistency: The AWS KMS API follows an eventual consistency model. For more information, see AWS KMS eventual consistency.

Request Syntax

```
{
   "KeyId": "string",
   "Limit": number,
   "Marker": "string"
}
```

Request Parameters

For information about the parameters that are common to all actions, see Common Parameters.

The request accepts the following data in JSON format.

> **Note**
>
> In the following list, the required parameters are described first.

KeyId

Lists only aliases that are associated with the specified KMS key. Enter a KMS key in your AWS account.

This parameter is optional. If you omit it, `ListAliases` returns all aliases in the account and Region.

Specify the key ID or key ARN of the KMS key.

For example:

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

To get the key ID and key ARN for a KMS key, use ListKeys or DescribeKey.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 2048.

Required: No

Limit

Use this parameter to specify the maximum number of items to return. When this value is present, AWS KMS does not return more than the specified number of items, but it might return fewer.

This value is optional. If you include a value, it must be between 1 and 100, inclusive. If you do not include a value, it defaults to 50.

Type: Integer

Valid Range: Minimum value of 1. Maximum value of 100.

Required: No

Marker

Use this parameter in a subsequent request after you receive a response with truncated results. Set it to the value of `NextMarker` from the truncated response you just received.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 1024.

Pattern: `[\u0020-\u00FF]*`

Required: No

Response Syntax

```
{
   "Aliases": [
      {
         "AliasArn": "string",
         "AliasName": "string",
         "CreationDate": number,
         "LastUpdatedDate": number,
         "TargetKeyId": "string"
      }
   ],
   "NextMarker": "string",
   "Truncated": boolean
}
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

Aliases

A list of aliases.

Type: Array of AliasListEntry objects

NextMarker

When `Truncated` is true, this element is present and contains the value to use for the `Marker` parameter in a subsequent request.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 1024.

Pattern: `[\u0020-\u00FF]*`

Truncated

A flag that indicates whether there are more items in the list. When this value is true, the list in this response is truncated. To get more items, pass the value of the `NextMarker` element in this response to the `Marker` parameter in a subsequent request.

Type: Boolean

Errors

For information about the errors that are common to all actions, see Common Error Types.

DependencyTimeoutException

The system timed out while trying to fulfill the request. You can retry the request.

HTTP Status Code: 500

InvalidArnException

The request was rejected because a specified ARN, or an ARN in a key policy, is not valid.

HTTP Status Code: 400

InvalidMarkerException

The request was rejected because the marker that specifies where pagination should next begin is not valid.

HTTP Status Code: 400

KMSInternalException

The request was rejected because an internal exception occurred. The request can be retried.

HTTP Status Code: 500

NotFoundException

The request was rejected because the specified entity or resource could not be found.

HTTP Status Code: 400

Examples

The following examples are formatted for legibility.

Example Request

This example illustrates one usage of ListAliases.

```
POST / HTTP/1.1
Host: kms.us-east-2.amazonaws.com
Content-Length: 2
X-Amz-Target: TrentService.ListAliases
X-Amz-Date: 20161203T011453Z
Content-Type: application/x-amz-json-1.1
Authorization: AWS4-HMAC-SHA256\
 Credential=AKIAI44QH8DHBEXAMPLE/20161203/us-east-2/kms/aws4_request,\
 SignedHeaders=content-type;host;x-amz-date;x-amz-target,\
 Signature=c2867e5f45167bf713e8f2c9998772ad72a20958db2cc0ef46bfba1632ca4d62

{}
```

Example Response

This example illustrates one usage of ListAliases.

```
HTTP/1.1 200 OK
Server: Server
Date: Sat, 03 Dec 2016 01:14:55 GMT
Content-Type: application/x-amz-json-1.1
Content-Length: 2874
Connection: keep-alive
x-amzn-RequestId: e6196175-b8f5-11e6-b404-15dcd0a7add5

{
  "Aliases": [
```

```
    {
      "AliasArn": "arn:aws:kms:us-east-2:111122223333:alias/aws/acm",
      "AliasName": "alias/aws/acm",
      "TargetKeyId": "da03f6f7-d279-427a-9cae-de48d07e5b66",
      "CreationDate": 1566518783.394,
      "LastUpdatedDate": 1566518783.394
    },
    {
      "AliasArn": "arn:aws:kms:us-east-2:111122223333:alias/aws/ebs",
      "AliasName": "alias/aws/ebs",
      "TargetKeyId": "25a217e7-7170-4b8c-8bf6-045ea5f70e5b",
      "CreationDate": 1493622000.704,
      "LastUpdatedDate": 1493622000.704
    },
    {
      "AliasArn": "arn:aws:kms:us-east-2:111122223333:alias/aws/elasticfilesystem",

      "AliasName": "alias/aws/elasticfilesystem",
      "TargetKeyId": "",
      "CreationDate": 1493622000.704,
      "LastUpdatedDate":
    }
    {
      "AliasArn": "arn:aws:kms:us-east-2:111122223333:alias/example1",
      "AliasName": "alias/example1",
      "TargetKeyId": "4da1e216-62d0-46c5-a7c0-5f3a3d2f8046",
      "CreationDate": 1593622000.191,
      "LastUpdatedDate": 1604158407.202
    },
    {
      "AliasArn": "arn:aws:kms:us-east-2:111122223333:alias/example2",
      "AliasName": "alias/example2",
      "TargetKeyId": "f32fef59-2cc2-445b-8573-2d73328acbee",
      "CreationDate": 1516435200.399,
      "LastUpdatedDate": 1516435200.399
    },
    {
      "AliasArn": "arn:aws:kms:us-east-2:111122223333:alias/example3",
      "AliasName": "alias/example3",
      "TargetKeyId": "1374ef38-d34e-4d5f-b2c9-4e0daee38855",
      "CreationDate": 1589526000.454,
      "LastUpdatedDate": 1589612400.106
    }
  ],
```

```
  "Truncated": false
}
```

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface V2
- AWS SDK for .NET V4
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- AWS SDK for PHP V3
- AWS SDK for Python (Boto3)
- AWS SDK for Ruby V3

ListGrants

Gets a list of all grants for the specified KMS key.

You must specify the KMS key in all requests. You can filter the grant list by grant ID, grantee principal, or grantee service principal.

For detailed information about grants, including grant terminology, see Grants in AWS KMS in the AWS Key Management Service Developer Guide . For examples of creating grants in several programming languages, see Use CreateGrant with an AWS SDK or CLI.

> **Note**
>
> When a grant is created with the `GranteePrincipal` field, the `ListGrants` response usually contains the user or role designated as the grantee principal in the grant. However, if the grantee principal is an AWS service, the `GranteePrincipal` field contains an AWS service principal, which might correspond to several different grantee principals, such as an IAM user, IAM role, or AWS account.
> When a grant is created with the `GranteeServicePrincipal` field, the `ListGrants` response always includes a `GranteeServicePrincipal` that indicates the grantee is actually an AWS service principal.

Cross-account use: Yes. To perform this operation on a KMS key in a different AWS account, specify the key ARN in the value of the `KeyId` parameter.

Required permissions: kms:ListGrants (key policy)

Related operations:

- CreateGrant
- ListRetirableGrants
- RetireGrant
- RevokeGrant

Eventual consistency: The AWS KMS API follows an eventual consistency model. For more information, see AWS KMS eventual consistency.

Request Syntax

```
{
   "GranteePrincipal": "string",
   "GranteeServicePrincipal": "string",
   "GrantId": "string",
   "KeyId": "string",
   "Limit": number,
   "Marker": "string"
}
```

Request Parameters

For information about the parameters that are common to all actions, see Common Parameters.

The request accepts the following data in JSON format.

> **Note**
>
> In the following list, the required parameters are described first.

KeyId

Returns only grants for the specified KMS key. This parameter is required.

Specify the key ID or key ARN of the KMS key. To specify a KMS key in a different AWS account, you must use the key ARN.

For example:

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

To get the key ID and key ARN for a KMS key, use ListKeys or DescribeKey.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 2048.

Required: Yes

GranteePrincipal

Returns only grants where the specified principal is the grantee principal for the grant.

You can specify either `GranteePrincipal` or `GranteeServicePrincipal`, but not both.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 256.

Pattern: `^[\w+=,.@:/-]+$`

Required: No

GranteeServicePrincipal

Returns only grants where the specified AWS service principal is the grantee service principal for the grant. This filter is only usable by callers in a service principal.

You can specify either `GranteePrincipal` or `GranteeServicePrincipal`, but not both.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 128.

Pattern: `^([A-Za-z0-9\-]+)\.([A-Za-z0-9\-]+)(\.[A-Za-z0-9\-]+)+$`

Required: No

GrantId

Returns only the grant with the specified grant ID. The grant ID uniquely identifies the grant.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 128.

Required: No

Limit

Use this parameter to specify the maximum number of items to return. When this value is present, AWS KMS does not return more than the specified number of items, but it might return fewer.

This value is optional. If you include a value, it must be between 1 and 100, inclusive. If you do not include a value, it defaults to 50.

Type: Integer

Valid Range: Minimum value of 1. Maximum value of 100.

Required: No

Marker

Use this parameter in a subsequent request after you receive a response with truncated results. Set it to the value of `NextMarker` from the truncated response you just received.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 1024.

Pattern: `[\u0020-\u00FF]*`

Required: No

Response Syntax

```
{
   "Grants": [
      {
         "Constraints": {
            "EncryptionContextEquals": {
               "string" : "string"
            },
            "EncryptionContextSubset": {
               "string" : "string"
            },
            "SourceArn": "string"
         },
         "CreationDate": number,
         "GranteePrincipal": "string",
         "GranteeServicePrincipal": "string",
         "GrantId": "string",
         "IssuingAccount": "string",
         "KeyId": "string",
         "Name": "string",
         "Operations": [ "string" ],
         "RetiringPrincipal": "string",
         "RetiringServicePrincipal": "string"
```

```
      }
   ],
   "NextMarker": "string",
   "Truncated": boolean
}
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

Grants

A list of grants.

Type: Array of GrantListEntry objects

NextMarker

When `Truncated` is true, this element is present and contains the value to use for the `Marker` parameter in a subsequent request.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 1024.

Pattern: `[\u0020-\u00FF]*`

Truncated

A flag that indicates whether there are more items in the list. When this value is true, the list in this response is truncated. To get more items, pass the value of the `NextMarker` element in this response to the `Marker` parameter in a subsequent request.

Type: Boolean

Errors

For information about the errors that are common to all actions, see Common Error Types.

DependencyTimeoutException

The system timed out while trying to fulfill the request. You can retry the request.

HTTP Status Code: 500

InvalidArnException

The request was rejected because a specified ARN, or an ARN in a key policy, is not valid.

HTTP Status Code: 400

InvalidGrantIdException

The request was rejected because the specified `GrantId` is not valid.

HTTP Status Code: 400

InvalidMarkerException

The request was rejected because the marker that specifies where pagination should next begin is not valid.

HTTP Status Code: 400

KMSInternalException

The request was rejected because an internal exception occurred. The request can be retried.

HTTP Status Code: 500

KMSInvalidStateException

The request was rejected because the state of the specified resource is not valid for this request.

This exceptions means one of the following:

- The key state of the KMS key is not compatible with the operation.

  To find the key state, use the DescribeKey operation. For more information about which key states are compatible with each AWS KMS operation, see Key states of AWS KMS keys in the AWS Key Management Service Developer Guide .
- For cryptographic operations on KMS keys in custom key stores, this exception represents a general failure with many possible causes. To identify the cause, see the error message that accompanies the exception.

HTTP Status Code: 400

NotFoundException

The request was rejected because the specified entity or resource could not be found.

HTTP Status Code: 400

Examples

The following examples are formatted for legibility.

Example Request

This example illustrates one usage of ListGrants.

```
POST / HTTP/1.1
Host: kms.us-east-2.amazonaws.com
Content-Length: 49
X-Amz-Target: TrentService.ListGrants
X-Amz-Date: 20161206T231134Z
Content-Type: application/x-amz-json-1.1
Authorization: AWS4-HMAC-SHA256\
 Credential=AKIAI44QH8DHBEXAMPLE/20161206/us-east-2/kms/aws4_request,\
 SignedHeaders=content-type;host;x-amz-date;x-amz-target,\
 Signature=157e1dd2ef1992e70e403e96c9f7122c5eb18bf82e4e5a71a83d63dcbc1c681b

{"KeyId": "1234abcd-12ab-34cd-56ef-1234567890ab"}
```

Example Response

This example illustrates one usage of ListGrants.

```
HTTP/1.1 200 OK
Server: Server
Date: Tue, 06 Dec 2016 23:11:34 GMT
Content-Type: application/x-amz-json-1.1
Content-Length: 1652
Connection: keep-alive
x-amzn-RequestId: 54ee4e2f-bc09-11e6-8073-89d6c33fcd1f

{
  "Grants": [
    {
      "CreationDate": 1.477431461E9,
      "GrantId": "91ad875e49b04a9d1f3bdeb84d821f9db6ea95e1098813f6d47f0c65fbe2a172",
      "GranteePrincipal": "acm.us-east-2.amazonaws.com",
      "IssuingAccount": "arn:aws:iam::111122223333:root",
```

```
      "KeyId": "arn:aws:kms:us-
east-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
      "Name": "",
      "Operations": [
        "Encrypt",
        "ReEncryptFrom",
        "ReEncryptTo"
      ],
      "RetiringPrincipal": "acm.us-east-2.amazonaws.com"
    },
    {
      "CreationDate": 1.477431461E9,
      "GrantId": "a5d67d3e207a8fc1f4928749ee3e52eb0440493a8b9cf05bbfad91655b056200",
      "GranteePrincipal": "acm.us-east-2.amazonaws.com",
      "IssuingAccount": "arn:aws:iam::111122223333:root",
      "KeyId": "arn:aws:kms:us-
east-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
      "Name": "",
      "Operations": [
        "ReEncryptFrom",
        "ReEncryptTo"
      ],
      "RetiringPrincipal": "acm.us-east-2.amazonaws.com"
    },
    {
      "CreationDate": 1.477431461E9,
      "GrantId": "c541aaf05d90cb78846a73b346fc43e65be28b7163129488c738e0c9e0628f4f",
      "GranteePrincipal": "acm.us-east-2.amazonaws.com",
      "IssuingAccount": "arn:aws:iam::111122223333:root",
      "KeyId": "arn:aws:kms:us-
east-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
      "Name": "",
      "Operations": [
        "Encrypt",
        "ReEncryptFrom",
        "ReEncryptTo"
      ],
      "RetiringPrincipal": "acm.us-east-2.amazonaws.com"
    },
    {
      "CreationDate": 1.477431461E9,
      "GrantId": "dd2052c67b4c76ee45caf1dc6a1e2d24e8dc744a51b36ae2f067dc540ce0105c",
      "GranteePrincipal": "acm.us-east-2.amazonaws.com",
      "IssuingAccount": "arn:aws:iam::111122223333:root",
```

```
      "KeyId": "arn:aws:kms:us-
east-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
      "Name": "",
      "Operations": [
        "Encrypt",
        "ReEncryptFrom",
        "ReEncryptTo"
      ],
      "RetiringPrincipal": "acm.us-east-2.amazonaws.com"
    }
  ],
  "Truncated": false
}
```

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface V2
- AWS SDK for .NET V4
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- AWS SDK for PHP V3
- AWS SDK for Python (Boto3)
- AWS SDK for Ruby V3

ListKeyPolicies

Gets the names of the key policies that are attached to a KMS key. This operation is designed to get policy names that you can use in a GetKeyPolicy operation. The valid policy names are `default` and `AWS_RAM`.

For information about the `AWS_RAM` key policy, see Sharing KMS keys with AWS RAM in the *AWS Key Management Service Developer Guide*.

Cross-account use: No. You cannot perform this operation on a KMS key in a different AWS account.

Required permissions: kms:ListKeyPolicies (key policy)

Related operations:

- GetKeyPolicy
- PutKeyPolicy

Eventual consistency: The AWS KMS API follows an eventual consistency model. For more information, see AWS KMS eventual consistency.

Request Syntax

```
{
   "KeyId": "string",
   "Limit": number,
   "Marker": "string"
}
```

Request Parameters

For information about the parameters that are common to all actions, see Common Parameters.

The request accepts the following data in JSON format.

> **Note**
>
> In the following list, the required parameters are described first.

KeyId

Gets the names of key policies for the specified KMS key.

Specify the key ID or key ARN of the KMS key.

For example:

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

To get the key ID and key ARN for a KMS key, use ListKeys or DescribeKey.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 2048.

Required: Yes

Limit

Use this parameter to specify the maximum number of items to return. When this value is present, AWS KMS does not return more than the specified number of items, but it might return fewer.

This value is optional. If you include a value, it must be between 1 and 1000, inclusive. If you do not include a value, it defaults to 100.

A KMS key can have at most two key policies, `default` and `AWS_RAM`.

Type: Integer

Valid Range: Minimum value of 1. Maximum value of 1000.

Required: No

Marker

Use this parameter in a subsequent request after you receive a response with truncated results. Set it to the value of `NextMarker` from the truncated response you just received.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 1024.

Pattern: `[\u0020-\u00FF]*`

Required: No

Response Syntax

```
{
   "NextMarker": "string",
   "PolicyNames": [ "string" ],
   "Truncated": boolean
}
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

NextMarker

When `Truncated` is true, this element is present and contains the value to use for the `Marker` parameter in a subsequent request.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 1024.

Pattern: `[\u0020-\u00FF]*`

PolicyNames

A list of key policy names. Valid values are:

default

The key policy that you manage.

AWS_RAM

The key policy that AWS RAM manages for KMS keys that you share through AWS RAM.

Type: Array of strings

Length Constraints: Minimum length of 1. Maximum length of 128.

Pattern: `[\w]+`

Truncated

A flag that indicates whether there are more items in the list. When this value is true, the list in this response is truncated. To get more items, pass the value of the `NextMarker` element in this response to the `Marker` parameter in a subsequent request.

Type: Boolean

Errors

For information about the errors that are common to all actions, see Common Error Types.

DependencyTimeoutException

The system timed out while trying to fulfill the request. You can retry the request.

HTTP Status Code: 500

InvalidArnException

The request was rejected because a specified ARN, or an ARN in a key policy, is not valid.

HTTP Status Code: 400

KMSInternalException

The request was rejected because an internal exception occurred. The request can be retried.

HTTP Status Code: 500

KMSInvalidStateException

The request was rejected because the state of the specified resource is not valid for this request.

This exceptions means one of the following:

- The key state of the KMS key is not compatible with the operation.

  To find the key state, use the DescribeKey operation. For more information about which key states are compatible with each AWS KMS operation, see Key states of AWS KMS keys in the AWS Key Management Service Developer Guide .

- For cryptographic operations on KMS keys in custom key stores, this exception represents a general failure with many possible causes. To identify the cause, see the error message that accompanies the exception.

HTTP Status Code: 400

NotFoundException

The request was rejected because the specified entity or resource could not be found.

HTTP Status Code: 400

Examples

The following examples are formatted for legibility.

Example Request

This example illustrates one usage of ListKeyPolicies.

```
POST / HTTP/1.1
Host: kms.us-east-2.amazonaws.com
Content-Length: 49
X-Amz-Target: TrentService.ListKeyPolicies
X-Amz-Date: 20161206T235923Z
Content-Type: application/x-amz-json-1.1
Authorization: AWS4-HMAC-SHA256\
 Credential=AKIAI44QH8DHBEXAMPLE/20161206/us-east-2/kms/aws4_request,\
 SignedHeaders=content-type;host;x-amz-date;x-amz-target,\
 Signature=82fe067c53d0dfff36793b8b6ef2d82d8adf0f1c05016bf4b4d6c50563ec7033

{"KeyId": "1234abcd-12ab-34cd-56ef-1234567890ab"}
```

Example Response

This example illustrates one usage of ListKeyPolicies.

```
HTTP/1.1 200 OK
Server: Server
Date: Tue, 06 Dec 2016 23:59:24 GMT
Content-Type: application/x-amz-json-1.1
```

```
Content-Length: 45
Connection: keep-alive
x-amzn-RequestId: 036f8e4b-bc10-11e6-b60b-ffb5eb2d1d15

{
  "PolicyNames": ["default"],
  "Truncated": false
}
```

Example – Request for a KMS key with an AWS_RAM key policy

This example illustrates one usage of ListKeyPolicies.

```
POST / HTTP/1.1
Host: kms.us-east-2.amazonaws.com
Content-Length: 49
X-Amz-Target: TrentService.ListKeyPolicies
X-Amz-Date: 20261117T193524Z
Content-Type: application/x-amz-json-1.1
Authorization: AWS4-HMAC-SHA256\
 Credential=AKIAI44QH8DHBEXAMPLE/20261117/us-east-2/kms/aws4_request,\
 SignedHeaders=content-type;host;x-amz-date;x-amz-target,\
 Signature=EXAMPLE

{"KeyId": "1234abcd-12ab-34cd-56ef-1234567890ab"}
```

Example – Response for a KMS key with an AWS_RAM key policy

This example illustrates one usage of ListKeyPolicies.

```
HTTP/1.1 200 OK
Server: Server
Date: Tue, 17 Nov 2026 19:35:25 GMT
Content-Type: application/x-amz-json-1.1
Content-Length: 65
Connection: keep-alive
x-amzn-RequestId: 56351da0-b3ac-43d7-b453-f827a64ab8d3

{
  "PolicyNames": ["AWS_RAM", "default"],
  "Truncated": false
}
```

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface V2
- AWS SDK for .NET V4
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- AWS SDK for PHP V3
- AWS SDK for Python (Boto3)
- AWS SDK for Ruby V3

ListKeyRotations

Returns information about the key materials associated with the specified KMS key. You can use the optional `IncludeKeyMaterial` parameter to control which key materials are included in the response.

You must specify the KMS key in all requests. You can refine the key rotations list by limiting the number of rotations returned.

For detailed information about automatic and on-demand key rotations, see Rotate AWS KMS keys in the *AWS Key Management Service Developer Guide*.

Cross-account use: No. You cannot perform this operation on a KMS key in a different AWS account.

Required permissions: kms:ListKeyRotations (key policy)

Related operations:

- EnableKeyRotation
- DeleteImportedKeyMaterial
- DisableKeyRotation
- GetKeyRotationStatus
- ImportKeyMaterial
- RotateKeyOnDemand

Eventual consistency: The AWS KMS API follows an eventual consistency model. For more information, see AWS KMS eventual consistency.

Request Syntax

```
{
   "IncludeKeyMaterial": "string",
   "KeyId": "string",
   "Limit": number,
   "Marker": "string"
}
```

Request Parameters

For information about the parameters that are common to all actions, see Common Parameters.

The request accepts the following data in JSON format.

> **Note**
>
> In the following list, the required parameters are described first.

KeyId

Gets the key rotations for the specified KMS key.

Specify the key ID or key ARN of the KMS key.

For example:

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

To get the key ID and key ARN for a KMS key, use ListKeys or DescribeKey.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 2048.

Required: Yes

IncludeKeyMaterial

Use this optional parameter to control which key materials associated with this key are listed in the response. The default value of this parameter is `ROTATIONS_ONLY`. If you omit this parameter, AWS KMS returns information on the key materials created by automatic or on-demand key rotation. When you specify a value of `ALL_KEY_MATERIAL`, AWS KMS adds the first key material and any imported key material pending rotation to the response. This parameter can only be used with KMS keys that support automatic or on-demand key rotation.

Type: String

Valid Values: `ALL_KEY_MATERIAL | ROTATIONS_ONLY`

Required: No

Limit

Use this parameter to specify the maximum number of items to return. When this value is present, AWS KMS does not return more than the specified number of items, but it might return fewer.

This value is optional. If you include a value, it must be between 1 and 1000, inclusive. If you do not include a value, it defaults to 100.

Type: Integer

Valid Range: Minimum value of 1. Maximum value of 1000.

Required: No

Marker

Use this parameter in a subsequent request after you receive a response with truncated results. Set it to the value of `NextMarker` from the truncated response you just received.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 1024.

Pattern: `[\u0020-\u00FF]*`

Required: No

Response Syntax

```
{
   "NextMarker": "string",
   "Rotations": [
      {
         "ExpirationModel": "string",
         "ImportState": "string",
         "KeyId": "string",
         "KeyMaterialDescription": "string",
         "KeyMaterialId": "string",
         "KeyMaterialState": "string",
         "RotationDate": number,
```

```
            "RotationType": "string",
            "ValidTo": number
        }
    ],
    "Truncated": boolean
}
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

NextMarker

When `Truncated` is true, this element is present and contains the value to use for the `Marker` parameter in a subsequent request.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 1024.

Pattern: `[\u0020-\u00FF]*`

Rotations

A list of completed key material rotations. When the optional input parameter `IncludeKeyMaterial` is specified with a value of `ALL_KEY_MATERIAL`, this list includes the first key material and any imported key material pending rotation.

Type: Array of RotationsListEntry objects

Truncated

A flag that indicates whether there are more items in the list. When this value is true, the list in this response is truncated. To get more items, pass the value of the `NextMarker` element in this response to the `Marker` parameter in a subsequent request.

Type: Boolean

Errors

For information about the errors that are common to all actions, see Common Error Types.

InvalidArnException

The request was rejected because a specified ARN, or an ARN in a key policy, is not valid.

HTTP Status Code: 400

InvalidMarkerException

The request was rejected because the marker that specifies where pagination should next begin is not valid.

HTTP Status Code: 400

KMSInternalException

The request was rejected because an internal exception occurred. The request can be retried.

HTTP Status Code: 500

KMSInvalidStateException

The request was rejected because the state of the specified resource is not valid for this request.

This exceptions means one of the following:

- The key state of the KMS key is not compatible with the operation.

  To find the key state, use the DescribeKey operation. For more information about which key states are compatible with each AWS KMS operation, see Key states of AWS KMS keys in the AWS Key Management Service Developer Guide .
- For cryptographic operations on KMS keys in custom key stores, this exception represents a general failure with many possible causes. To identify the cause, see the error message that accompanies the exception.

HTTP Status Code: 400

NotFoundException

The request was rejected because the specified entity or resource could not be found.

HTTP Status Code: 400

UnsupportedOperationException

The request was rejected because a specified parameter is not supported or a specified resource is not valid for this operation.

HTTP Status Code: 400

Examples

Example Request

The following example is formatted for legibility.

```
POST / HTTP/1.1
Host: kms.us-east-2.amazonaws.com
Content-Length: 48
X-Amz-Target: TrentService.ListKeyRotations
X-Amz-Date: 20240405T151426Z
Content-Type: application/x-amz-json-1.1
Authorization: AWS4-HMAC-SHA256\
 Credential=AKIAI44QH8DHBEXAMPLE/20161107/us-east-2/kms/aws4_request,\
 SignedHeaders=content-type;host;x-amz-date;x-amz-target,\
 Signature=4783e177036ca78627fe0cda9dcfdaf4ad7c8312d0e7c3d71d814b0c4cff1c0b

{"KeyId": "1234abcd-12ab-34cd-56ef-1234567890ab"}
```

Example Response

This example illustrates one usage of ListKeyRotations.

```
HTTP/1.1 200 OK
Server: Server
Date: Fri, 05 Apr 2024 15:14:26 GMT
Content-Type: application/x-amz-json-1.1
Content-Length: 0
Connection: keep-alive
x-amzn-RequestId: 2077c3bf-a538-11e6-b6fb-794e83344f84

{
    "Rotations": [
        {
          "KeyId": "1234abcd-12ab-34cd-56ef-1234567890ab",
          "KeyMaterialId":
 "0b7fd7ddbac6eef27907413567cad8c810e2883dc8a7534067a82ee1142fc1e6",
          "KeyMaterialType": "NON_CURRENT",
          "RotationDate": "2024-03-02T10:11:36.564000+00:00",
          "RotationType": "AUTOMATIC"
```

```
        },
        {
          "KeyId": "1234abcd-12ab-34cd-56ef-1234567890ab",
          "KeyMaterialId":
 "2067e8b4da7b0d8ffa80ff7afb5ea4637e103c98c3c110bdb84cc0a9de75e733",
          "KeyMaterialType": "CURRENT",
          "RotationDate":  "2024-04-05T15:14:47.757000+00:00",
          "RotationType": "ON_DEMAND"
        }
    ],
  "Truncated": false
}
```

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface V2
- AWS SDK for .NET V4
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- AWS SDK for PHP V3
- AWS SDK for Python (Boto3)
- AWS SDK for Ruby V3

ListKeys

Gets a list of all KMS keys in the caller's AWS account and Region.

You can also use `ListKeys` to list KMS keys in the Region that other AWS accounts shared with the caller's AWS account through AWS Resource Access Manager (AWS RAM). To list them, use the `ResourceOwners` parameter. If you omit `ResourceOwners`, `ListKeys` lists only the KMS keys in the caller's AWS account.

When `ResourceOwners` includes `OTHER_ACCOUNTS_VIA_RAM`, `ListKeys` returns the KMS keys that other AWS accounts currently share with the caller's AWS account through AWS RAM. For information about how and when a resource that another account shares becomes available to your AWS account, see Access AWS resources shared with you in the *AWS RAM User Guide*.

Cross-account use: No. This operation doesn't take a KMS key parameter, so you can't use it to list the KMS keys of a specific AWS account. The only KMS keys in other AWS accounts that `ListKeys` returns are the ones that other accounts shared with the caller's AWS account through AWS RAM. To list them, set the `ResourceOwners` parameter to include `OTHER_ACCOUNTS_VIA_RAM`.

Required permissions: kms:ListKeys (IAM policy)

The same `kms:ListKeys` permission lets you list the KMS keys that other accounts shared with you through AWS RAM. AWS KMS doesn't evaluate the key policies of shared KMS keys when it lists them. To use a shared KMS key, you need permission for each operation on that KMS key.

Related operations:

- CreateKey
- DescribeKey
- ListAliases
- ListResourceTags

Eventual consistency: The AWS KMS API follows an eventual consistency model. For more information, see AWS KMS eventual consistency.

KMS keys that other AWS accounts share with you can take some time to appear in or disappear from `ListKeys` results after changes to resource shares, such as accepting an invitation or deleting a resource share.

Request Syntax

```
{
   "Limit": number,
   "Marker": "string",
   "ResourceOwners": [ "string" ]
}
```

Request Parameters

For information about the parameters that are common to all actions, see Common Parameters.

The request accepts the following data in JSON format.

> **Note**
>
> In the following list, the required parameters are described first.

Limit

Use this parameter to specify the maximum number of items to return. When this value is present, AWS KMS does not return more than the specified number of items, but it might return fewer.

This value is optional. If you include a value, it must be between 1 and 1000, inclusive. If you do not include a value, it defaults to 100.

Type: Integer

Valid Range: Minimum value of 1. Maximum value of 1000.

Required: No

Marker

Use this parameter in a subsequent request after you receive a response with truncated results. Set it to the value of `NextMarker` from the truncated response you just received.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 1024.

Pattern: `[\u0020-\u00FF]*`

Required: No

ResourceOwners

Specifies whose KMS keys to list. Use this parameter to list KMS keys that other AWS accounts shared with your AWS account through AWS Resource Access Manager (AWS RAM).

Specify one or two values. Valid values are:

SELF

Lists the KMS keys in your AWS account and Region.

OTHER_ACCOUNTS_VIA_RAM

Lists the KMS keys in the Region that other AWS accounts currently share with your AWS account through AWS RAM. This list doesn't include KMS keys that your AWS account owns. For each of these KMS keys, the key ARN in the response contains the account ID of the AWS account that owns the KMS key.

If you omit this parameter, `ListKeys` lists only the KMS keys in your AWS account. If you specify both values, in either order, `ListKeys` returns the KMS keys in your AWS account first, and then the KMS keys that other AWS accounts shared with you. A single response doesn't contain both kinds of KMS keys.

When `ResourceOwners` includes `OTHER_ACCOUNTS_VIA_RAM`, a response can contain fewer KMS keys than the `Limit` value, or none, even when `Truncated` is true. To get all of the KMS keys, continue calling `ListKeys` with the `NextMarker` value from each response until `Truncated` is false. Use a `NextMarker` value only in a request that has the same `ResourceOwners` value as the request that returned it. Otherwise, the request can fail with an `InvalidMarkerException`.

KMS keys that other AWS accounts share with you can take some time to appear in or disappear from the results after changes to resource shares.

Type: Array of strings

Array Members: Minimum number of 1 item. Maximum number of 2 items.

Valid Values: `SELF | OTHER_ACCOUNTS_VIA_RAM`

Required: No

Response Syntax

```
{
   "Keys": [
      {
         "KeyArn": "string",
         "KeyId": "string"
      }
   ],
   "NextMarker": "string",
   "Truncated": boolean
}
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

Keys

A list of KMS keys. For a KMS key that another AWS account shared with you through AWS RAM, the key ARN contains the account ID of the AWS account that owns the KMS key.

Type: Array of KeyListEntry objects

NextMarker

When `Truncated` is true, this element is present and contains the value to use for the `Marker` parameter in a subsequent request.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 1024.

Pattern: `[\u0020-\u00FF]*`

Truncated

A flag that indicates whether there are more items in the list. When this value is true, the list in this response is truncated. To get more items, pass the value of the `NextMarker` element in this response to the `Marker` parameter in a subsequent request.

Type: Boolean

Errors

For information about the errors that are common to all actions, see Common Error Types.

DependencyTimeoutException

The system timed out while trying to fulfill the request. You can retry the request.

HTTP Status Code: 500

InvalidMarkerException

The request was rejected because the marker that specifies where pagination should next begin is not valid.

HTTP Status Code: 400

KMSInternalException

The request was rejected because an internal exception occurred. The request can be retried.

HTTP Status Code: 500

Examples

The following examples are formatted for legibility.

Example Request

This example illustrates one usage of ListKeys.

```
POST / HTTP/1.1
Host: kms.us-east-2.amazonaws.com
Content-Length: 2
X-Amz-Target: TrentService.ListKeys
X-Amz-Date: 20161207T003550Z
Content-Type: application/x-amz-json-1.1
Authorization: AWS4-HMAC-SHA256\
 Credential=AKIAI44QH8DHBEXAMPLE/20161207/us-east-2/kms/aws4_request,\
 SignedHeaders=content-type;host;x-amz-date;x-amz-target,\
```

```
 Signature=2196a20c1a139ae8f6fe070881f41954616c775bb5a484814c35f8ee35cfa448

{}
```

Example Response

This example illustrates one usage of ListKeys.

```
HTTP/1.1 200 OK
Server: Server
Date: Wed, 07 Dec 2016 00:35:50 GMT
Content-Type: application/x-amz-json-1.1
Content-Length: 980
Connection: keep-alive
x-amzn-RequestId: 1a5f0a53-bc15-11e6-82b3-e9e4af764a06

{
  "Keys": [
    {
      "KeyArn": "arn:aws:kms:us-east-2:111122223333:key/0d990263-018e-4e65-a703-
eff731de951e",
      "KeyId": "0d990263-018e-4e65-a703-eff731de951e"
    },
    {
      "KeyArn": "arn:aws:kms:us-
east-2:111122223333:key/144be297-0ae1-44ac-9c8f-93cd8c82f841",
      "KeyId": "144be297-0ae1-44ac-9c8f-93cd8c82f841"
    },
    {
      "KeyArn": "arn:aws:kms:us-east-2:111122223333:key/21184251-b765-428e-
b852-2c7353e72571",
      "KeyId": "21184251-b765-428e-b852-2c7353e72571"
    },
    {
      "KeyArn": "arn:aws:kms:us-east-2:111122223333:key/214fe92f-5b03-4ae1-b350-
db2a45dbe10c",
      "KeyId": "214fe92f-5b03-4ae1-b350-db2a45dbe10c"
    },
    {
      "KeyArn": "arn:aws:kms:us-east-2:111122223333:key/339963f2-e523-49d3-af24-
a0fe752aa458",
      "KeyId": "339963f2-e523-49d3-af24-a0fe752aa458"
    },
```

```
    {
      "KeyArn": "arn:aws:kms:us-east-2:111122223333:key/b776a44b-df37-4438-9be4-
a27494e4271a",
      "KeyId": "b776a44b-df37-4438-9be4-a27494e4271a"
    },
    {
      "KeyArn": "arn:aws:kms:us-east-2:111122223333:key/deaf6c9e-cf2c-46a6-
bf6d-0b6d487cffbb",
      "KeyId": "deaf6c9e-cf2c-46a6-bf6d-0b6d487cffbb"
    }
  ],
  "Truncated": false
}
```

Example – Request for KMS keys shared with you

This example illustrates one usage of ListKeys.

```
POST / HTTP/1.1
Host: kms.us-east-2.amazonaws.com
Content-Length: 50
X-Amz-Target: TrentService.ListKeys
X-Amz-Date: 20261117T184227Z
Content-Type: application/x-amz-json-1.1
Authorization: AWS4-HMAC-SHA256\
 Credential=AKIAI44QH8DHBEXAMPLE/20261117/us-east-2/kms/aws4_request,\
 SignedHeaders=content-type;host;x-amz-date;x-amz-target,\
 Signature=EXAMPLE

{
  "ResourceOwners": ["OTHER_ACCOUNTS_VIA_RAM"]
}
```

Example – Response for KMS keys shared with you

This example illustrates one usage of ListKeys.

```
HTTP/1.1 200 OK
Server: Server
Date: Tue, 17 Nov 2026 18:42:27 GMT
Content-Type: application/x-amz-json-1.1
Content-Length: 364
```

```
Connection: keep-alive
x-amzn-RequestId: 3e8c6f2a-7d41-4b9e-a5c0-9f1b2d3e4a5c

{
  "Keys": [
    {
      "KeyArn": "arn:aws:kms:us-
east-2:444455556666:key/1234abcd-12ab-34cd-56ef-1234567890ab",
      "KeyId": "1234abcd-12ab-34cd-56ef-1234567890ab"
    },
    {
      "KeyArn": "arn:aws:kms:us-east-2:444455556666:key/0987dcba-09fe-87dc-65ba-
ab0987654321",
      "KeyId": "0987dcba-09fe-87dc-65ba-ab0987654321"
    }
  ],
  "Truncated": false
}
```

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface V2
- AWS SDK for .NET V4
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- AWS SDK for PHP V3
- AWS SDK for Python (Boto3)
- AWS SDK for Ruby V3

ListResourceTags

Returns all tags on the specified KMS key.

For general information about tags, including the format and syntax, see Tagging AWS resources in the *Amazon Web Services General Reference*. For information about using tags in AWS KMS, see Tags in AWS KMS.

Cross-account use: No. You cannot perform this operation on a KMS key in a different AWS account.

Required permissions: kms:ListResourceTags (key policy)

Related operations:

- CreateKey
- ReplicateKey
- TagResource
- UntagResource

Eventual consistency: The AWS KMS API follows an eventual consistency model. For more information, see AWS KMS eventual consistency.

Request Syntax

```
{
   "KeyId": "string",
   "Limit": number,
   "Marker": "string"
}
```

Request Parameters

For information about the parameters that are common to all actions, see Common Parameters.

The request accepts the following data in JSON format.

> **Note**
>
> In the following list, the required parameters are described first.

KeyId

Gets tags on the specified KMS key.

Specify the key ID or key ARN of the KMS key.

For example:

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

To get the key ID and key ARN for a KMS key, use ListKeys or DescribeKey.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 2048.

Required: Yes

Limit

Use this parameter to specify the maximum number of items to return. When this value is present, AWS KMS does not return more than the specified number of items, but it might return fewer.

This value is optional. If you include a value, it must be between 1 and 50, inclusive. If you do not include a value, it defaults to 50.

Type: Integer

Valid Range: Minimum value of 1. Maximum value of 1000.

Required: No

Marker

Use this parameter in a subsequent request after you receive a response with truncated results. Set it to the value of `NextMarker` from the truncated response you just received.

Do not attempt to construct this value. Use only the value of `NextMarker` from the truncated response you just received.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 1024.

Pattern: `[\u0020-\u00FF]*`

Required: No

Response Syntax

```
{
   "NextMarker": "string",
   "Tags": [
      {
         "TagKey": "string",
         "TagValue": "string"
      }
   ],
   "Truncated": boolean
}
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

NextMarker

When `Truncated` is true, this element is present and contains the value to use for the `Marker` parameter in a subsequent request.

Do not assume or infer any information from this value.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 1024.

Pattern: `[\u0020-\u00FF]*`

Tags

A list of tags. Each tag consists of a tag key and a tag value.

> **Note**
>
> Tagging or untagging a KMS key can allow or deny permission to the KMS key. For details, see ABAC for AWS KMS in the *AWS Key Management Service Developer Guide*.

Type: Array of Tag objects

Truncated

A flag that indicates whether there are more items in the list. When this value is true, the list in this response is truncated. To get more items, pass the value of the `NextMarker` element in this response to the `Marker` parameter in a subsequent request.

Type: Boolean

Errors

For information about the errors that are common to all actions, see Common Error Types.

InvalidArnException

The request was rejected because a specified ARN, or an ARN in a key policy, is not valid.

HTTP Status Code: 400

InvalidMarkerException

The request was rejected because the marker that specifies where pagination should next begin is not valid.

HTTP Status Code: 400

KMSInternalException

The request was rejected because an internal exception occurred. The request can be retried.

HTTP Status Code: 500

NotFoundException

The request was rejected because the specified entity or resource could not be found.

HTTP Status Code: 400

Examples

The following examples are formatted for legibility.

Example Request

This example illustrates one usage of ListResourceTags.

```
POST / HTTP/1.1
Host: kms.us-east-2.amazonaws.com
Content-Length: 49
X-Amz-Target: TrentService.ListResourceTags
X-Amz-Date: 20170109T200421Z
Content-Type: application/x-amz-json-1.1
Authorization: AWS4-HMAC-SHA256\
 Credential=AKIAI44QH8DHBEXAMPLE/20170109/us-east-2/kms/aws4_request,\
 SignedHeaders=content-type;host;x-amz-date;x-amz-target,\
 Signature=17706fce40fda00c6768b3297355c353490c1dfdf3b3a9591193612961cd2cb4

{"KeyId": "1234abcd-12ab-34cd-56ef-1234567890ab"}
```

Example Response

This example illustrates one usage of ListResourceTags.

```
HTTP/1.1 200 OK
Server: Server
Date: Mon, 09 Jan 2017 20:04:22 GMT
Content-Type: application/x-amz-json-1.1
Content-Length: 158
Connection: keep-alive
x-amzn-RequestId: cfb46544-d6a6-11e6-a164-b5365990e84e

{
  "Tags": [{
    "TagKey": "CostCenter",
```

```
    "TagValue": "87654"
  }, {
    "TagKey": "CreatedBy",
    "TagValue": "ExampleUser"
  }, {
    "TagKey": "Purpose",
    "TagValue": "Test"
  }],
  "Truncated": false
}
```

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface V2
- AWS SDK for .NET V4
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- AWS SDK for PHP V3
- AWS SDK for Python (Boto3)
- AWS SDK for Ruby V3

ListRetirableGrants

Returns information about all grants in the AWS account and Region that have the specified retiring principal or retiring service principal.

You can specify any principal in your AWS account. The grants that are returned include grants for KMS keys in your AWS account and other AWS accounts. You might use this operation to determine which grants you may retire. To retire a grant, use the RetireGrant operation.

For detailed information about grants, including grant terminology, see Grants in AWS KMS in the AWS Key Management Service Developer Guide . For examples of creating grants in several programming languages, see Use CreateGrant with an AWS SDK or CLI.

Cross-account use: You must specify a principal in your AWS account. This operation returns a list of grants where the retiring principal specified in the `ListRetirableGrants` request is the same retiring principal on the grant. This can include grants on KMS keys owned by other AWS accounts, but you do not need `kms:ListRetirableGrants` permission (or any other additional permission) in any AWS account other than your own.

Required permissions: kms:ListRetirableGrants (IAM policy) in your AWS account.

> **Note**
>
> When listing retirable grants by `RetiringPrincipal`, AWS KMS authorizes `ListRetirableGrants` requests by evaluating the caller account's kms:ListRetirableGrants permissions. The authorized resource in `ListRetirableGrants` calls is the retiring principal specified in the request. AWS KMS does not evaluate the caller's permissions to verify their access to any KMS keys or grants that might be returned by the `ListRetirableGrants` call.
> The `RetiringServicePrincipal` filter is only usable by callers in a service principal.

Related operations:

- CreateGrant
- ListGrants
- RetireGrant
- RevokeGrant

Eventual consistency: The AWS KMS API follows an eventual consistency model. For more information, see AWS KMS eventual consistency.

Request Syntax

```
{
   "Limit": number,
   "Marker": "string",
   "RetiringPrincipal": "string",
   "RetiringServicePrincipal": "string"
}
```

Request Parameters

For information about the parameters that are common to all actions, see Common Parameters.

The request accepts the following data in JSON format.

> **Note**
>
> In the following list, the required parameters are described first.

Limit

Use this parameter to specify the maximum number of items to return. When this value is present, AWS KMS does not return more than the specified number of items, but it might return fewer.

This value is optional. If you include a value, it must be between 1 and 100, inclusive. If you do not include a value, it defaults to 50.

Type: Integer

Valid Range: Minimum value of 1. Maximum value of 100.

Required: No

Marker

Use this parameter in a subsequent request after you receive a response with truncated results. Set it to the value of `NextMarker` from the truncated response you just received.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 1024.

Pattern: `[\u0020-\u00FF]*`

Required: No

RetiringPrincipal

The retiring principal for which to list grants. Enter a principal in your AWS account.

To specify the retiring principal, use the Amazon Resource Name (ARN) of an AWS principal. Valid principals include AWS accounts, IAM users, IAM roles, federated users, and assumed role users. For help with the ARN syntax for a principal, see IAM ARNs in the AWS Identity and Access Management User Guide .

You must specify either `RetiringPrincipal` or `RetiringServicePrincipal`, but not both.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 256.

Pattern: `^[\w+=,.@:/-]+$`

Required: No

RetiringServicePrincipal

The retiring service principal for which to list grants. This filter is only usable by callers in a service principal.

You must specify either `RetiringPrincipal` or `RetiringServicePrincipal`, but not both.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 128.

Pattern: `^([A-Za-z0-9\-]+)\.([A-Za-z0-9\-]+)(\.[A-Za-z0-9\-]+)+$`

Required: No

Response Syntax

```
{
```

```
   "Grants": [
      {
         "Constraints": {
            "EncryptionContextEquals": {
               "string" : "string"
            },
            "EncryptionContextSubset": {
               "string" : "string"
            },
            "SourceArn": "string"
         },
         "CreationDate": number,
         "GranteePrincipal": "string",
         "GranteeServicePrincipal": "string",
         "GrantId": "string",
         "IssuingAccount": "string",
         "KeyId": "string",
         "Name": "string",
         "Operations": [ "string" ],
         "RetiringPrincipal": "string",
         "RetiringServicePrincipal": "string"
      }
   ],
   "NextMarker": "string",
   "Truncated": boolean
}
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

Grants

A list of grants.

Type: Array of GrantListEntry objects

NextMarker

When `Truncated` is true, this element is present and contains the value to use for the `Marker` parameter in a subsequent request.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 1024.

Pattern: `[\u0020-\u00FF]*`

Truncated

A flag that indicates whether there are more items in the list. When this value is true, the list in this response is truncated. To get more items, pass the value of the `NextMarker` element in this response to the `Marker` parameter in a subsequent request.

Type: Boolean

Errors

For information about the errors that are common to all actions, see Common Error Types.

DependencyTimeoutException

The system timed out while trying to fulfill the request. You can retry the request.

HTTP Status Code: 500

InvalidArnException

The request was rejected because a specified ARN, or an ARN in a key policy, is not valid.

HTTP Status Code: 400

InvalidMarkerException

The request was rejected because the marker that specifies where pagination should next begin is not valid.

HTTP Status Code: 400

KMSInternalException

The request was rejected because an internal exception occurred. The request can be retried.

HTTP Status Code: 500

NotFoundException

The request was rejected because the specified entity or resource could not be found.

HTTP Status Code: 400

Examples

The following examples are formatted for legibility.

Example Request

This example illustrates one usage of ListRetirableGrants.

```
POST / HTTP/1.1
Host: kms.us-east-2.amazonaws.com
Content-Length: 61
X-Amz-Target: TrentService.ListRetirableGrants
X-Amz-Date: 20161207T191040Z
Content-Type: application/x-amz-json-1.1
Authorization: AWS4-HMAC-SHA256\
 Credential=AKIAI44QH8DHBEXAMPLE/20161207/us-east-2/kms/aws4_request,\
 SignedHeaders=content-type;host;x-amz-date;x-amz-target,\
 Signature=d5e43f0cfd75a3251f40bc27e76f83b3110b33e3d972142ae118b2b3c0f67b39

{"RetiringPrincipal": "arn:aws:iam::111122223333:role/ExampleRole"}
```

Example Response

This example illustrates one usage of ListRetirableGrants.

```
HTTP/1.1 200 OK
Server: Server
Date: Wed, 07 Dec 2016 19:10:41 GMT
Content-Type: application/x-amz-json-1.1
Content-Length: 436
Connection: keep-alive
x-amzn-RequestId: d86125dc-bcb0-11e6-82b3-e9e4af764a06

{
  "Grants": [
    {
      "CreationDate": 1.481137775E9,
      "GrantId": "0c237476b39f8bc44e45212e08498fbe3151305030726c0590dd8d3e9f3d6a60",
      "GranteePrincipal": "arn:aws:iam::111122223333:role/ExampleRole",
      "IssuingAccount": "arn:aws:iam::444455556666:root",
```

```
      "KeyId": "arn:aws:kms:us-
east-2:444455556666:key/1234abcd-12ab-34cd-56ef-1234567890ab",
      "Name": "",
      "Operations": [
        "Decrypt",
        "Encrypt"
      ],
      "RetiringPrincipal": "arn:aws:iam::111122223333:role/ExampleRole"
    }
  ],
  "Truncated": false
}
```

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface V2
- AWS SDK for .NET V4
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- AWS SDK for PHP V3
- AWS SDK for Python (Boto3)
- AWS SDK for Ruby V3

PutKeyPolicy

Attaches a key policy to the specified KMS key. You can use this operation to set only the `default` key policy.

For more information about key policies, see Key Policies in the *AWS Key Management Service Developer Guide*. For help writing and formatting a JSON policy document, see the IAM JSON Policy Reference in the AWS Identity and Access Management User Guide . For examples of adding a key policy in multiple programming languages, see Use PutKeyPolicy with an AWS SDK or CLI in the *AWS Key Management Service Developer Guide*.

A KMS key can also have an `AWS_RAM` key policy, which AWS Resource Access Manager (AWS RAM) manages for KMS keys that you share through AWS RAM. You can't use `PutKeyPolicy` to create, change, or delete the `AWS_RAM` key policy. If you set `PolicyName` to `AWS_RAM`, the request fails with a `ValidationException` and the message `The AWS_RAM policy can only be managed by the RAM service.` To view the `AWS_RAM` key policy, use GetKeyPolicy. For information about sharing KMS keys, see Sharing KMS keys with AWS RAM in the *AWS Key Management Service Developer Guide*.

Cross-account use: No. You cannot perform this operation on a KMS key in a different AWS account.

Required permissions: kms:PutKeyPolicy (key policy)

You can use the kms:PolicyName condition key to control access to this operation based on the value of the `PolicyName` parameter in the request. If the request doesn't include `PolicyName`, the value of `kms:PolicyName` is `default`.

Related operations: GetKeyPolicy

Eventual consistency: The AWS KMS API follows an eventual consistency model. For more information, see AWS KMS eventual consistency.

Request Syntax

```
{
   "BypassPolicyLockoutSafetyCheck": boolean,
   "KeyId": "string",
   "Policy": "string",
   "PolicyName": "string"
```

```
}
```

Request Parameters

For information about the parameters that are common to all actions, see Common Parameters.

The request accepts the following data in JSON format.

> **Note**
>
> In the following list, the required parameters are described first.

KeyId

Sets the key policy on the specified KMS key.

Specify the key ID or key ARN of the KMS key.

For example:

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

To get the key ID and key ARN for a KMS key, use ListKeys or DescribeKey.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 2048.

Required: Yes

Policy

The key policy to attach to the KMS key.

The key policy must meet the following criteria:

- The key policy must allow the calling principal to make a subsequent `PutKeyPolicy` request on the KMS key. This reduces the risk that the KMS key becomes unmanageable. For more information, see Default key policy in the *AWS Key Management Service Developer Guide*. (To omit this condition, set `BypassPolicyLockoutSafetyCheck` to true.)

- Each statement in the key policy must contain one or more principals. The principals in the key policy must exist and be visible to AWS KMS. When you create a new AWS principal, you might need to enforce a delay before including the new principal in a key policy because the new principal might not be immediately visible to AWS KMS. For more information, see Changes that I make are not always immediately visible in the *AWS Identity and Access Management User Guide*.

> **Note**
>
> If either of the required `Resource` or `Action` elements are missing from a key policy statement, the policy statement has no effect. When a key policy statement is missing one of these elements, the AWS KMS console correctly reports an error, but the `PutKeyPolicy` API request succeeds, even though the policy statement is ineffective. For more information on required key policy elements, see Elements in a key policy in the *AWS Key Management Service Developer Guide*.

A key policy document can include only the following characters:

- Printable ASCII characters from the space character (`\u0020`) through the end of the ASCII character range.
- Printable characters in the Basic Latin and Latin-1 Supplement character set (through `\u00FF`).
- The tab (`\u0009`), line feed (`\u000A`), and carriage return (`\u000D`) special characters

> **Note**
>
> If the key policy exceeds the length constraint, AWS KMS returns a `LimitExceededException`.

For information about key policies, see Key policies in AWS KMS in the *AWS Key Management Service Developer Guide*.For help writing and formatting a JSON policy document, see the IAM JSON Policy Reference in the AWS Identity and Access Management User Guide .

Type: String

Length Constraints: Minimum length of 1. Maximum length of 32768.

Pattern: `[\u0009\u000A\u000D\u0020-\u00FF]+`

Required: Yes

BypassPolicyLockoutSafetyCheck

Skips ("bypasses") the key policy lockout safety check. The default value is false.

> **Important**
>
> Setting this value to true increases the risk that the KMS key becomes unmanageable. Do not set this value to true indiscriminately.
> For more information, see Default key policy in the *AWS Key Management Service Developer Guide*.

Use this parameter only when you intend to prevent the principal that is making the request from making a subsequent PutKeyPolicy request on the KMS key.

The key policy lockout safety check and the `BypassPolicyLockoutSafetyCheck` parameter apply only to the `default` key policy. The check evaluates only the new `default` key policy, with `kms:PolicyName` set to `default`. It doesn't consider permissions in the `AWS_RAM` key policy.

Type: Boolean

Required: No

PolicyName

The name of the key policy. If no policy name is specified, the default value is `default`. Valid values are `default` and `AWS_RAM`, but you can set only the `default` key policy. AWS RAM manages the `AWS_RAM` key policy for KMS keys that you share through AWS RAM. If you specify `AWS_RAM`, the request fails with a `ValidationException`, or with an `AccessDeniedException` if you don't have permission to call `PutKeyPolicy` with that policy name.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 128.

Pattern: `[\w]+`

Required: No

Response Elements

If the action is successful, the service sends back an HTTP 200 response with an empty HTTP body.

Errors

For information about the errors that are common to all actions, see Common Error Types.

DependencyTimeoutException

The system timed out while trying to fulfill the request. You can retry the request.

HTTP Status Code: 500

InvalidArnException

The request was rejected because a specified ARN, or an ARN in a key policy, is not valid.

HTTP Status Code: 400

KMSInternalException

The request was rejected because an internal exception occurred. The request can be retried.

HTTP Status Code: 500

KMSInvalidStateException

The request was rejected because the state of the specified resource is not valid for this request.

This exceptions means one of the following:

- The key state of the KMS key is not compatible with the operation.

  To find the key state, use the DescribeKey operation. For more information about which key states are compatible with each AWS KMS operation, see Key states of AWS KMS keys in the AWS Key Management Service Developer Guide .
- For cryptographic operations on KMS keys in custom key stores, this exception represents a general failure with many possible causes. To identify the cause, see the error message that accompanies the exception.

HTTP Status Code: 400

LimitExceededException

The request was rejected because a length constraint or quota was exceeded. For more information, see Quotas in the *AWS Key Management Service Developer Guide*.

HTTP Status Code: 400

MalformedPolicyDocumentException

The request was rejected because the specified policy is not syntactically or semantically correct.

HTTP Status Code: 400

NotFoundException

The request was rejected because the specified entity or resource could not be found.

HTTP Status Code: 400

UnsupportedOperationException

The request was rejected because a specified parameter is not supported or a specified resource is not valid for this operation.

HTTP Status Code: 400

Examples

Example Request

The following example is formatted for legibility.

```
POST / HTTP/1.1
Host: kms.us-east-2.amazonaws.com
Content-Length: 2396
X-Amz-Target: TrentService.PutKeyPolicy
X-Amz-Date: 20161207T203023Z
Content-Type: application/x-amz-json-1.1
Authorization: AWS4-HMAC-SHA256\
 Credential=AKIAI44QH8DHBEXAMPLE/20161207/us-east-2/kms/aws4_request,\
 SignedHeaders=content-type;host;x-amz-date;x-amz-target,\
 Signature=e58ea91db06afc1bc7a1f204769cf6bc4d003ee090095a13caef361c69739ede

{
```

```
    "Policy": "{
        \"Version\": \"2012-10-17\",
        \"Id\": \"custom-policy-2016-12-07\",
        \"Statement\": [
            {
                \"Sid\": \"Enable IAM User Permissions\",
                \"Effect\": \"Allow\",
                \"Principal\": {
                    \"AWS\": \"arn:aws:iam::111122223333:root\"
                },
                \"Action\": \"kms:*\",
                \"Resource\": \"*\"
            },
            {
                \"Sid\": \"Allow access for Key Administrators\",
                \"Effect\": \"Allow\",
                \"Principal\": {
                    \"AWS\": [
                        \"arn:aws:iam::111122223333:user/ExampleAdminUser\",
                        \"arn:aws:iam::111122223333:role/ExampleAdminRole\"
                    ]
                },
                \"Action\": [
                    \"kms:Create*\",
                    \"kms:Describe*\",
                    \"kms:Enable*\",
                    \"kms:List*\",
                    \"kms:Put*\",
                    \"kms:Update*\",
                    \"kms:Revoke*\",
                    \"kms:Disable*\",
                    \"kms:Get*\",
                    \"kms:Delete*\",
                    \"kms:ScheduleKeyDeletion\",
                    \"kms:CancelKeyDeletion\"
                ],
                \"Resource\": \"*\"
            },
            {
                \"Sid\": \"Allow use of the key\",
                \"Effect\": \"Allow\",
                \"Principal\": {
                    \"AWS\": \"arn:aws:iam::111122223333:role/ExamplePowerUserRole\"
                },
```

```
                \"Action\": [
                    \"kms:Encrypt\",
                    \"kms:Decrypt\",
                    \"kms:ReEncrypt*\",
                    \"kms:GenerateDataKey*\",
                    \"kms:DescribeKey\"
                ],
                \"Resource\": \"*\"
            },
            {
                \"Sid\": \"Allow attachment of persistent resources\",
                \"Effect\": \"Allow\",
                \"Principal\": {
                    \"AWS\": \"arn:aws:iam::111122223333:role/ExamplePowerUserRole\"
                },
                \"Action\": [
                    \"kms:CreateGrant\",
                    \"kms:ListGrants\",
                    \"kms:RevokeGrant\"
                ],
                \"Resource\": \"*\",
                \"Condition\": {
                    \"Bool\": {
                        \"kms:GrantIsForAWSResource\": \"true\"
                    }
                }
            }
        ]
    }",
    "PolicyName": "default",
    "KeyId": "1234abcd-12ab-34cd-56ef-1234567890ab"
}
```

Example Response

This example illustrates one usage of PutKeyPolicy.

```
HTTP/1.1 200 OK
Server: Server
Date: Wed, 07 Dec 2016 20:30:23 GMT
Content-Type: application/x-amz-json-1.1
Content-Length: 0
Connection: keep-alive
```

```
x-amzn-RequestId: fb114d4c-bcbb-11e6-82b3-e9e4af764a06
```

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface V2
- AWS SDK for .NET V4
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- AWS SDK for PHP V3
- AWS SDK for Python (Boto3)
- AWS SDK for Ruby V3

ReEncrypt

Decrypts ciphertext and then reencrypts it entirely within AWS KMS. You can use this operation to change the KMS key under which data is encrypted, such as when you manually rotate a KMS key or change the KMS key that protects a ciphertext. You can also use it to reencrypt ciphertext under the same KMS key, such as to change the encryption context of a ciphertext.

The `ReEncrypt` operation can decrypt ciphertext that was encrypted by using a KMS key in an AWS KMS operation, such as Encrypt or GenerateDataKey. It can also decrypt ciphertext that was encrypted by using the public key of an asymmetric KMS key outside of AWS KMS. However, it cannot decrypt ciphertext produced by other libraries, such as the AWS Encryption SDK or Amazon S3 client-side encryption. These libraries return a ciphertext format that is incompatible with AWS KMS.

When you use the `ReEncrypt` operation, you need to provide information for the decrypt operation and the subsequent encrypt operation.

- If your ciphertext was encrypted under an asymmetric KMS key, you must use the `SourceKeyId` parameter to identify the KMS key that encrypted the ciphertext. You must also supply the encryption algorithm that was used. This information is required to decrypt the data.
- If your ciphertext was encrypted under a symmetric encryption KMS key, the `SourceKeyId` parameter is optional. AWS KMS can get this information from metadata that it adds to the symmetric ciphertext blob. This feature adds durability to your implementation by ensuring that authorized users can decrypt ciphertext decades after it was encrypted, even if they've lost track of the key ID. However, specifying the source KMS key is always recommended as a best practice. When you use the `SourceKeyId` parameter to specify a KMS key, AWS KMS uses only the KMS key you specify. If the ciphertext was encrypted under a different KMS key, the `ReEncrypt` operation fails. This practice ensures that you use the KMS key that you intend.
- To reencrypt the data, you must use the `DestinationKeyId` parameter to specify the KMS key that re-encrypts the data after it is decrypted. If the destination KMS key is an asymmetric KMS key, you must also provide the encryption algorithm. The algorithm that you choose must be compatible with the KMS key.

> **Important**
>
> When you use an asymmetric KMS key to encrypt or reencrypt data, be sure to record the KMS key and encryption algorithm that you choose. You will be required to provide the

same KMS key and encryption algorithm when you decrypt the data. If the KMS key and algorithm do not match the values used to encrypt the data, the decrypt operation fails. You are not required to supply the key ID and encryption algorithm when you decrypt with symmetric encryption KMS keys because AWS KMS stores this information in the ciphertext blob. AWS KMS cannot store metadata in ciphertext generated with asymmetric keys. The standard format for asymmetric key ciphertext does not include configurable fields.

The KMS key that you use for this operation must be in a compatible key state. For details, see Key states of AWS KMS keys in the *AWS Key Management Service Developer Guide*.

Note

When using grants with `SourceArn` constraints for `ReEncrypt` operations, the grants on both the source KMS key (for `ReEncryptFrom`) and the destination KMS key (for `ReEncryptTo`) must specify the same `SourceArn` value.

Cross-account use: Yes. The source KMS key and destination KMS key can be in different AWS accounts. Either or both KMS keys can be in a different account than the caller. To specify a KMS key in a different account, use the key ARN or alias ARN. A short key ID is also acceptable for the source key when decrypting symmetric ciphertexts, though using a full key ARN is recommended to be more explicit about the intended KMS key.

Required permissions:

- kms:ReEncryptFrom permission on the source KMS key (key policy)
- kms:ReEncryptTo permission on the destination KMS key (key policy)

To permit reencryption from or to a KMS key, include the `"kms:ReEncrypt*"` permission in your key policy. This permission is automatically included in the key policy when you use the console to create a KMS key. But you must include it manually when you create a KMS key programmatically or when you use the PutKeyPolicy operation to set a key policy.

Related operations:

- Decrypt

- Encrypt
- GenerateDataKey
- GenerateDataKeyPair

Eventual consistency: The AWS KMS API follows an eventual consistency model. For more information, see AWS KMS eventual consistency.

Request Syntax

```
{
   "CiphertextBlob": blob,
   "DestinationEncryptionAlgorithm": "string",
   "DestinationEncryptionContext": {
      "string" : "string"
   },
   "DestinationKeyId": "string",
   "DryRun": boolean,
   "DryRunModifiers": [ "string" ],
   "GrantTokens": [ "string" ],
   "SourceEncryptionAlgorithm": "string",
   "SourceEncryptionContext": {
      "string" : "string"
   },
   "SourceKeyId": "string"
}
```

Request Parameters

For information about the parameters that are common to all actions, see Common Parameters.

The request accepts the following data in JSON format.

> **Note**
>
> In the following list, the required parameters are described first.

DestinationKeyId

A unique identifier for the KMS key that is used to reencrypt the data. Specify a symmetric encryption KMS key or an asymmetric KMS key with a `KeyUsage` value of `ENCRYPT_DECRYPT`. To find the `KeyUsage` value of a KMS key, use the DescribeKey operation.

To specify a KMS key, use its key ID, key ARN, alias name, or alias ARN. When using an alias name, prefix it with `"alias/"`. To specify a KMS key in a different AWS account, you must use the key ARN or alias ARN.

For example:

- Key ID: `1234abcd-12ab-34cd-56ef-1234567890ab`
- Key ARN: `arn:aws:kms:us-east-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab`
- Alias name: `alias/ExampleAlias`
- Alias ARN: `arn:aws:kms:us-east-2:111122223333:alias/ExampleAlias`

To get the key ID and key ARN for a KMS key, use ListKeys or DescribeKey. To get the alias name and alias ARN, use ListAliases.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 2048.

Required: Yes

CiphertextBlob

Ciphertext of the data to reencrypt.

This parameter is required in all cases except when `DryRun` is `true` and `DryRunModifiers` is set to `IGNORE_CIPHERTEXT`.

Type: Base64-encoded binary data object

Length Constraints: Minimum length of 1. Maximum length of 6144.

Required: No

DestinationEncryptionAlgorithm

Specifies the encryption algorithm that AWS KMS will use to reecrypt the data after it has decrypted it. The default value, `SYMMETRIC_DEFAULT`, represents the encryption algorithm used for symmetric encryption KMS keys.

This parameter is required only when the destination KMS key is an asymmetric KMS key.

Type: String

Valid Values: `SYMMETRIC_DEFAULT | RSAES_OAEP_SHA_1 | RSAES_OAEP_SHA_256 | SM2PKE`

Required: No

DestinationEncryptionContext

Specifies that encryption context to use when the reencrypting the data.

> **Important**
>
> Do not include confidential or sensitive information in this field. This field may be displayed in plaintext in CloudTrail logs and other output.

A destination encryption context is valid only when the destination KMS key is a symmetric encryption KMS key. The standard ciphertext format for asymmetric KMS keys does not include fields for metadata.

An *encryption context* is a collection of non-secret key-value pairs that represent additional authenticated data. When you use an encryption context to encrypt data, you must specify the same (an exact case-sensitive match) encryption context to decrypt the data. An encryption context is supported only on operations with symmetric encryption KMS keys. On operations with symmetric encryption KMS keys, an encryption context is optional, but it is strongly recommended.

For more information, see Encryption context in the *AWS Key Management Service Developer Guide*.

Type: String to string map

Required: No

DryRun

Checks if your request will succeed. `DryRun` is an optional parameter.

To learn more about how to use this parameter, see Testing your permissions in the *AWS Key Management Service Developer Guide*.

Type: Boolean

Required: No

DryRunModifiers

Specifies the modifiers to apply to the dry run operation. `DryRunModifiers` is an optional parameter that only applies when `DryRun` is set to `true`.

When set to `IGNORE_CIPHERTEXT`, AWS KMS performs only authorization validation without ciphertext validation. This allows you to test permissions without requiring a valid ciphertext blob.

To learn more about how to use this parameter, see Testing your permissions in the *AWS Key Management Service Developer Guide*.

Type: Array of strings

Valid Values: `IGNORE_CIPHERTEXT`

Required: No

GrantTokens

A list of grant tokens.

Use a grant token when your permission to call this operation comes from a new grant that has not yet achieved *eventual consistency*. For more information, see Grant token and Using a grant token in the *AWS Key Management Service Developer Guide*.

Type: Array of strings

Array Members: Minimum number of 0 items. Maximum number of 10 items.

Length Constraints: Minimum length of 1. Maximum length of 8192.

Required: No

SourceEncryptionAlgorithm

Specifies the encryption algorithm that AWS KMS will use to decrypt the ciphertext before it is reencrypted. The default value, `SYMMETRIC_DEFAULT`, represents the algorithm used for symmetric encryption KMS keys.

Specify the same algorithm that was used to encrypt the ciphertext. If you specify a different algorithm, the decrypt attempt fails.

This parameter is required only when the ciphertext was encrypted under an asymmetric KMS key.

Type: String

Valid Values: `SYMMETRIC_DEFAULT | RSAES_OAEP_SHA_1 | RSAES_OAEP_SHA_256 | SM2PKE`

Required: No

SourceEncryptionContext

Specifies the encryption context to use to decrypt the ciphertext. Enter the same encryption context that was used to encrypt the ciphertext.

An *encryption context* is a collection of non-secret key-value pairs that represent additional authenticated data. When you use an encryption context to encrypt data, you must specify the same (an exact case-sensitive match) encryption context to decrypt the data. An encryption context is supported only on operations with symmetric encryption KMS keys. On operations with symmetric encryption KMS keys, an encryption context is optional, but it is strongly recommended.

For more information, see Encryption context in the *AWS Key Management Service Developer Guide*.

Type: String to string map

Required: No

SourceKeyId

Specifies the KMS key that AWS KMS will use to decrypt the ciphertext before it is re-encrypted.

Enter a key ID of the KMS key that was used to encrypt the ciphertext. If you identify a different KMS key, the `ReEncrypt` operation throws an `IncorrectKeyException`.

This parameter is required only when the ciphertext was encrypted under an asymmetric KMS key or when `DryRun` is `true` and `DryRunModifiers` is set to `IGNORE_CIPHERTEXT`. If you used a symmetric encryption KMS key, AWS KMS can get the KMS key from metadata that it adds to the symmetric ciphertext blob. However, it is always recommended as a best practice. This practice ensures that you use the KMS key that you intend.

To specify a KMS key, use its key ID, key ARN, alias name, or alias ARN. When using an alias name, prefix it with `"alias/"`. To specify a KMS key in a different AWS account, you should use the key ARN or alias ARN.

For example:

- Key ID: `1234abcd-12ab-34cd-56ef-1234567890ab`
- Key ARN: `arn:aws:kms:us-east-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab`
- Alias name: `alias/ExampleAlias`
- Alias ARN: `arn:aws:kms:us-east-2:111122223333:alias/ExampleAlias`

To get the key ID and key ARN for a KMS key, use ListKeys or DescribeKey. To get the alias name and alias ARN, use ListAliases.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 2048.

Required: No

Response Syntax

```
{
   "CiphertextBlob": blob,
   "DestinationEncryptionAlgorithm": "string",
   "DestinationKeyMaterialId": "string",
   "KeyId": "string",
   "SourceEncryptionAlgorithm": "string",
   "SourceKeyId": "string",
   "SourceKeyMaterialId": "string"
```

```
}
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

CiphertextBlob

The reencrypted data. When you use the HTTP API or the AWS CLI, the value is Base64-encoded. Otherwise, it is not Base64-encoded.

Type: Base64-encoded binary data object

Length Constraints: Minimum length of 1. Maximum length of 6144.

DestinationEncryptionAlgorithm

The encryption algorithm that was used to reencrypt the data.

Type: String

Valid Values: `SYMMETRIC_DEFAULT | RSAES_OAEP_SHA_1 | RSAES_OAEP_SHA_256 | SM2PKE`

DestinationKeyMaterialId

The identifier of the key material used to reencrypt the data. This field is present only when data is reencrypted using a symmetric encryption KMS key.

Type: String

Length Constraints: Fixed length of 64.

Pattern: `^[a-f0-9]+$`

KeyId

The Amazon Resource Name (key ARN) of the KMS key that was used to reencrypt the data.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 2048.

SourceEncryptionAlgorithm

The encryption algorithm that was used to decrypt the ciphertext before it was reencrypted.

Type: String

Valid Values: `SYMMETRIC_DEFAULT | RSAES_OAEP_SHA_1 | RSAES_OAEP_SHA_256 | SM2PKE`

SourceKeyId

Unique identifier of the KMS key used to originally encrypt the data.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 2048.

SourceKeyMaterialId

The identifier of the key material used to originally encrypt the data. This field is present only when the original encryption used a symmetric encryption KMS key.

Type: String

Length Constraints: Fixed length of 64.

Pattern: `^[a-f0-9]+$`

Errors

For information about the errors that are common to all actions, see Common Error Types.

DependencyTimeoutException

The system timed out while trying to fulfill the request. You can retry the request.

HTTP Status Code: 500

DisabledException

The request was rejected because the specified KMS key is not enabled.

HTTP Status Code: 400

DryRunOperationException

The request was rejected because the DryRun parameter was specified.

HTTP Status Code: 400

IncorrectKeyException

The request was rejected because the specified KMS key cannot decrypt the data. The `KeyId` in a Decrypt request and the `SourceKeyId` in a ReEncrypt request must identify the same KMS key that was used to encrypt the ciphertext.

HTTP Status Code: 400

InvalidCiphertextException

From the Decrypt or ReEncrypt operation, the request was rejected because the specified ciphertext, or additional authenticated data incorporated into the ciphertext, such as the encryption context, is corrupted, missing, or otherwise invalid.

From the ImportKeyMaterial operation, the request was rejected because AWS KMS could not decrypt the encrypted (wrapped) key material.

From the Decapsulate operation, the request was rejected because the length of the `SharedSecretCiphertextBlob` is inconsistent with the specified ML-KEM key spec.

HTTP Status Code: 400

InvalidGrantTokenException

The request was rejected because the specified grant token is not valid.

HTTP Status Code: 400

InvalidKeyUsageException

The request was rejected for one of the following reasons:

- The `KeyUsage` value of the KMS key is incompatible with the API operation.
- The encryption algorithm or signing algorithm specified for the operation is incompatible with the type of key material in the KMS key (`KeySpec`).

For encrypting, decrypting, re-encrypting, and generating data keys, the `KeyUsage` must be `ENCRYPT_DECRYPT`. For signing and verifying messages, the `KeyUsage` must be `SIGN_VERIFY`.

For generating and verifying message authentication codes (MACs), the KeyUsage must be GENERATE_VERIFY_MAC. For deriving key agreement secrets, the KeyUsage must be KEY_AGREEMENT. For encapsulating and decapsulating shared secrets, the KeyUsage must be ENCAPSULATE_DECAPSULATE. To find the KeyUsage of a KMS key, use the DescribeKey operation.

To find the encryption or signing algorithms supported for a particular KMS key, use the DescribeKey operation.

HTTP Status Code: 400

KeyUnavailableException

The request was rejected because the specified KMS key was not available. You can retry the request.

HTTP Status Code: 500

KMSInternalException

The request was rejected because an internal exception occurred. The request can be retried.

HTTP Status Code: 500

KMSInvalidStateException

The request was rejected because the state of the specified resource is not valid for this request.

This exceptions means one of the following:

- The key state of the KMS key is not compatible with the operation.

  To find the key state, use the DescribeKey operation. For more information about which key states are compatible with each AWS KMS operation, see Key states of AWS KMS keys in the AWS Key Management Service Developer Guide .
- For cryptographic operations on KMS keys in custom key stores, this exception represents a general failure with many possible causes. To identify the cause, see the error message that accompanies the exception.

HTTP Status Code: 400

NotFoundException

The request was rejected because the specified entity or resource could not be found.

HTTP Status Code: 400

Examples

The following examples are formatted for legibility.

Example Request

This example illustrates one usage of ReEncrypt.

```
POST / HTTP/1.1
Host: kms.us-east-2.amazonaws.com
Content-Length: 306
X-Amz-Target: TrentService.ReEncrypt
X-Amz-Date: 20161207T225816Z
Content-Type: application/x-amz-json-1.1* ReEncrypt
Authorization: AWS4-HMAC-SHA256\
 Credential=AKIAI44QH8DHBEXAMPLE/20161207/us-east-2/kms/aws4_request,\
 SignedHeaders=content-type;host;x-amz-date;x-amz-target,\
 Signature=7afd339e2a680e0726592ddf687aabe48e1d8a7933a60ebbdc0154b8e2936ef2

{
  "SourceKeyId": "arn:aws:kms:us-
east-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
  "DestinationKeyId": "0987dcba-09fe-87dc-65ba-ab0987654321",
  "CiphertextBlob": "AQECAHj/M9MyvNsMT8kW
+K5DVkMfunTHr0w6V6crnuAGw80uRwAAAH0wewYJKoZIhvcNAQcGoG4wbAIBADBnBgkqhkiG9w0BBwEwHgYJYIZIAWUDBAE
+FSkUmNmmE0H0aHHRYRD6XqUnaCNnzAuhhq4VTGBfii6oWtjVU83pGmradvUawxE/tbCg=="
}
```

Example Response

This example illustrates one usage of ReEncrypt.

```
HTTP/1.1 200 OK
Server: Server
Date: Wed, 07 Dec 2016 22:58:17 GMT
Content-Type: application/x-amz-json-1.1
Content-Length: 423
Connection: keep-alive
x-amzn-RequestId: a434eca2-bcd0-11e6-b60b-ffb5eb2d1d15
```

```
{
  "CiphertextBlob":
 "AQECAHjRYf5WytIc0C857tFSnBaPn2F8DgfmThbJlGfR8P3WlwAAAH0wewYJKoZIhvcNAQcGoG4wbAIBADBnBgkqhkiG9
vwjXjPBhQIBEIA6wjfzufQPhuU
+nVqa3Kj4nqSTdhDw1PTkImKCUEuvQDui6qsooyB4Qxe8OOBqciRNC7ENQN8lKaEijg==",
  "KeyId": "arn:aws:kms:us-east-2:111122223333:key/0987dcba-09fe-87dc-65ba-
ab0987654321",
  "SourceKeyId": "arn:aws:kms:us-
east-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
  "SourceEncryptionAlgorithm": "SYMMETRIC_DEFAULT",
  "SourceKeyMaterialId":
 "0b7fd7ddbac6eef27907413567cad8c810e2883dc8a7534067a82ee1142fc1e6",
  "DestinationEncryptionAlgorithm": "SYMMETRIC_DEFAULT",
  "DestinationKeyMaterialId":
 "2067e8b4da7b0d8ffa80ff7afb5ea4637e103c98c3c110bdb84cc0a9de75e733"
}
```


See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface V2
- AWS SDK for .NET V4
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- AWS SDK for PHP V3
- AWS SDK for Python (Boto3)
- AWS SDK for Ruby V3

ReplicateKey

Replicates a multi-Region key into the specified Region. This operation creates a multi-Region replica key based on a multi-Region primary key in a different Region of the same AWS partition. You can create multiple replicas of a primary key, but each must be in a different Region. To create a multi-Region primary key, use the CreateKey operation.

This operation supports *multi-Region keys*, an AWS KMS feature that lets you create multiple interoperable KMS keys in different AWS Regions. Because these KMS keys have the same key ID, key material, and other metadata, you can use them interchangeably to encrypt data in one AWS Region and decrypt it in a different AWS Region without re-encrypting the data or making a cross-Region call. For more information about multi-Region keys, see Multi-Region keys in AWS KMS in the *AWS Key Management Service Developer Guide*.

A *replica key* is a fully-functional KMS key that can be used independently of its primary and peer replica keys. A primary key and its replica keys share properties that make them interoperable. They have the same key ID and key material. They also have the same key spec, key usage, key material origin, and automatic key rotation status. AWS KMS automatically synchronizes these shared properties among related multi-Region keys. All other properties of a replica key can differ, including its key policy, tags, aliases, and key state. AWS KMS pricing and quotas for KMS keys apply to each primary key and replica key.

When this operation completes, the new replica key has a transient key state of `Creating`. This key state changes to `Enabled` (or `PendingImport`) after a few seconds when the process of creating the new replica key is complete. While the key state is `Creating`, you can manage key, but you cannot yet use it in cryptographic operations. If you are creating and using the replica key programmatically, retry on `KMSInvalidStateException` or call `DescribeKey` to check its `KeyState` value before using it. For details about the `Creating` key state, see Key states of AWS KMS keys in the *AWS Key Management Service Developer Guide*.

You cannot create more than one replica of a primary key in any Region. If the Region already includes a replica of the key you're trying to replicate, `ReplicateKey` returns an `AlreadyExistsException` error. If the key state of the existing replica is `PendingDeletion`, you can cancel the scheduled key deletion (CancelKeyDeletion) or wait for the key to be deleted. The new replica key you create will have the same shared properties as the original replica key.

The AWS CloudTrail log of a `ReplicateKey` operation records a `ReplicateKey` operation in the primary key's Region and a CreateKey operation in the replica key's Region.

If you replicate a multi-Region primary key with imported key material, the replica key is created with no key material. You must import the same key material that you imported into the primary key.

To convert a replica key to a primary key, use the UpdatePrimaryRegion operation.

> **Note**
>
> `ReplicateKey` uses different default values for the `KeyPolicy` and `Tags` parameters than those used in the AWS KMS console. For details, see the parameter descriptions.

Cross-account use: No. You cannot use this operation to create a replica key in a different AWS account.

Required permissions:

- `kms:ReplicateKey` on the primary key (in the primary key's Region). Include this permission in the primary key's key policy.
- `kms:CreateKey` in an IAM policy in the replica Region.
- To use the `Tags` parameter, `kms:TagResource` in an IAM policy in the replica Region.

Related operations

- CreateKey
- UpdatePrimaryRegion

Eventual consistency: The AWS KMS API follows an eventual consistency model. For more information, see AWS KMS eventual consistency.

Request Syntax

```
{
   "BypassPolicyLockoutSafetyCheck": boolean,
   "Description": "string",
   "KeyId": "string",
   "Policy": "string",
   "ReplicaRegion": "string",
   "Tags": [
```

```
      {
         "TagKey": "string",
         "TagValue": "string"
      }
   ]
}
```

Request Parameters

For information about the parameters that are common to all actions, see Common Parameters.

The request accepts the following data in JSON format.

> **Note**
>
> In the following list, the required parameters are described first.

KeyId

Identifies the multi-Region primary key that is being replicated. To determine whether a KMS key is a multi-Region primary key, use the DescribeKey operation to check the value of the `MultiRegionKeyType` property.

Specify the key ID or key ARN of a multi-Region primary key.

For example:

- Key ID: `mrk-1234abcd12ab34cd56ef1234567890ab`
- Key ARN: `arn:aws:kms:us-east-2:111122223333:key/mrk-1234abcd12ab34cd56ef1234567890ab`

To get the key ID and key ARN for a KMS key, use ListKeys or DescribeKey.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 2048.

Required: Yes

ReplicaRegion

The Region ID of the AWS Region for this replica key.

Enter the Region ID, such as `us-east-1` or `ap-southeast-2`. For a list of AWS Regions in which AWS KMS is supported, see AWS KMS service endpoints in the *Amazon Web Services General Reference*.

The replica must be in a different AWS Region than its primary key and other replicas of that primary key, but in the same AWS partition. AWS KMS must be available in the replica Region. If the Region is not enabled by default, the AWS account must be enabled in the Region. For information about AWS partitions, see Amazon Resource Names (ARNs) in the *Amazon Web Services General Reference*. For information about enabling and disabling Regions, see Enabling a Region and Disabling a Region in the *Amazon Web Services General Reference*.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 32.

Pattern: `^([a-z]+-){2,3}\d+$`

Required: Yes

BypassPolicyLockoutSafetyCheck

Skips ("bypasses") the key policy lockout safety check. The default value is false.

> **Important**
>
> Setting this value to true increases the risk that the KMS key becomes unmanageable. Do not set this value to true indiscriminately.
> For more information, see Default key policy in the *AWS Key Management Service Developer Guide*.

Use this parameter only when you intend to prevent the principal that is making the request from making a subsequent PutKeyPolicy request on the KMS key.

Type: Boolean

Required: No

Description

A description of the KMS key. The default value is an empty string (no description).

> **Important**
>
> Do not include confidential or sensitive information in this field. This field may be displayed in plaintext in CloudTrail logs and other output.

The description is not a shared property of multi-Region keys. You can specify the same description or a different description for each key in a set of related multi-Region keys. AWS KMS does not synchronize this property.

Type: String

Length Constraints: Minimum length of 0. Maximum length of 8192.

Required: No

Policy

The key policy to attach to the KMS key. This parameter is optional. If you do not provide a key policy, AWS KMS attaches the default key policy to the KMS key.

The key policy is not a shared property of multi-Region keys. You can specify the same key policy or a different key policy for each key in a set of related multi-Region keys. AWS KMS does not synchronize this property.

If you provide a key policy, it must meet the following criteria:

- The key policy must allow the calling principal to make a subsequent `PutKeyPolicy` request on the KMS key. This reduces the risk that the KMS key becomes unmanageable. For more information, see Default key policy in the *AWS Key Management Service Developer Guide*. (To omit this condition, set `BypassPolicyLockoutSafetyCheck` to true.)
- Each statement in the key policy must contain one or more principals. The principals in the key policy must exist and be visible to AWS KMS. When you create a new AWS principal, you might need to enforce a delay before including the new principal in a key policy because the new principal might not be immediately visible to AWS KMS. For more information, see Changes that I make are not always immediately visible in the *AWS Identity and Access Management User Guide*.

A key policy document can include only the following characters:

- Printable ASCII characters from the space character (`\u0020`) through the end of the ASCII character range.

- Printable characters in the Basic Latin and Latin-1 Supplement character set (through \u00FF).
- The tab (\u0009), line feed (\u000A), and carriage return (\u000D) special characters

For information about key policies, see Key policies in AWS KMS in the *AWS Key Management Service Developer Guide*. For help writing and formatting a JSON policy document, see the IAM JSON Policy Reference in the AWS Identity and Access Management User Guide .

Type: String

Length Constraints: Minimum length of 1. Maximum length of 32768.

Pattern: [\u0009\u000A\u000D\u0020-\u00FF]+

Required: No

Tags

Assigns one or more tags to the replica key. Use this parameter to tag the KMS key when it is created. To tag an existing KMS key, use the TagResource operation.

> **Important**
>
> Do not include confidential or sensitive information in this field. This field may be displayed in plaintext in CloudTrail logs and other output.

> **Note**
>
> Tagging or untagging a KMS key can allow or deny permission to the KMS key. For details, see ABAC for AWS KMS in the *AWS Key Management Service Developer Guide*.

To use this parameter, you must have kms:TagResource permission in an IAM policy.

Tags are not a shared property of multi-Region keys. You can specify the same tags or different tags for each key in a set of related multi-Region keys. AWS KMS does not synchronize this property.

Each tag consists of a tag key and a tag value. Both the tag key and the tag value are required, but the tag value can be an empty (null) string. You cannot have more than one tag on a KMS

key with the same tag key. If you specify an existing tag key with a different tag value, AWS KMS replaces the current tag value with the specified one.

When you add tags to an AWS resource, AWS generates a cost allocation report with usage and costs aggregated by tags. Tags can also be used to control access to a KMS key. For details, see Tags in AWS KMS.

Type: Array of Tag objects

Required: No

Response Syntax

```
{
   "ReplicaKeyMetadata": {
      "Arn": "string",
      "AWSAccountId": "string",
      "CloudHsmClusterId": "string",
      "CreationDate": number,
      "CurrentKeyMaterialId": "string",
      "CustomerMasterKeySpec": "string",
      "CustomKeyStoreId": "string",
      "DeletionDate": number,
      "Description": "string",
      "Enabled": boolean,
      "EncapsulationAlgorithms": [ "string" ],
      "EncryptionAlgorithms": [ "string" ],
      "ExpirationModel": "string",
      "KeyAgreementAlgorithms": [ "string" ],
      "KeyId": "string",
      "KeyManager": "string",
      "KeySpec": "string",
      "KeyState": "string",
      "KeyUsage": "string",
      "MacAlgorithms": [ "string" ],
      "MultiRegion": boolean,
      "MultiRegionConfiguration": {
         "MultiRegionKeyType": "string",
         "PrimaryKey": {
            "Arn": "string",
            "Region": "string"
         },
```

```
         "ReplicaKeys": [
            {
               "Arn": "string",
               "Region": "string"
            }
         ]
      },
      "Origin": "string",
      "PendingDeletionWindowInDays": number,
      "SigningAlgorithms": [ "string" ],
      "ValidTo": number,
      "XksKeyConfiguration": {
         "Id": "string"
      }
   },
   "ReplicaPolicy": "string",
   "ReplicaTags": [
      {
         "TagKey": "string",
         "TagValue": "string"
      }
   ]
}
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

ReplicaKeyMetadata

Displays details about the new replica key, including its Amazon Resource Name (key ARN) and Key states of AWS KMS keys. It also includes the ARN and AWS Region of its primary key and other replica keys.

Type: KeyMetadata object

ReplicaPolicy

The key policy of the new replica key. The value is a key policy document in JSON format.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 131072.

Pattern: `[\u0009\u000A\u000D\u0020-\u00FF]+`

ReplicaTags

The tags on the new replica key. The value is a list of tag key and tag value pairs.

Type: Array of Tag objects

Errors

For information about the errors that are common to all actions, see Common Error Types.

AlreadyExistsException

The request was rejected because it attempted to create a resource that already exists.

HTTP Status Code: 400

DisabledException

The request was rejected because the specified KMS key is not enabled.

HTTP Status Code: 400

InvalidArnException

The request was rejected because a specified ARN, or an ARN in a key policy, is not valid.

HTTP Status Code: 400

KMSInternalException

The request was rejected because an internal exception occurred. The request can be retried.

HTTP Status Code: 500

KMSInvalidStateException

The request was rejected because the state of the specified resource is not valid for this request.

This exceptions means one of the following:

- The key state of the KMS key is not compatible with the operation.

  To find the key state, use the DescribeKey operation. For more information about which key states are compatible with each AWS KMS operation, see Key states of AWS KMS keys in the AWS Key Management Service Developer Guide .
- For cryptographic operations on KMS keys in custom key stores, this exception represents a general failure with many possible causes. To identify the cause, see the error message that accompanies the exception.

HTTP Status Code: 400

LimitExceededException

The request was rejected because a length constraint or quota was exceeded. For more information, see Quotas in the *AWS Key Management Service Developer Guide*.

HTTP Status Code: 400

MalformedPolicyDocumentException

The request was rejected because the specified policy is not syntactically or semantically correct.

HTTP Status Code: 400

NotFoundException

The request was rejected because the specified entity or resource could not be found.

HTTP Status Code: 400

TagException

The request was rejected because one or more tags are not valid.

HTTP Status Code: 400

UnsupportedOperationException

The request was rejected because a specified parameter is not supported or a specified resource is not valid for this operation.

HTTP Status Code: 400

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface V2
- AWS SDK for .NET V4
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- AWS SDK for PHP V3
- AWS SDK for Python (Boto3)
- AWS SDK for Ruby V3

RetireGrant

Deletes a grant. Typically, you retire a grant when you no longer need its permissions. To identify the grant to retire, use a grant token, or both the grant ID and a key identifier (key ID or key ARN) of the KMS key. The CreateGrant operation returns both values.

This operation can be called by the *retiring principal* for a grant, by the *grantee principal* if the grant allows the `RetireGrant` operation, and by the AWS account in which the grant is created. It can also be called by principals to whom permission for retiring a grant is delegated.

For detailed information about grants, including grant terminology, see Grants in AWS KMS in the AWS Key Management Service Developer Guide . For examples of creating grants in several programming languages, see Use CreateGrant with an AWS SDK or CLI.

Cross-account use: Yes. You can retire a grant on a KMS key in a different AWS account.

Required permissions: Permission to retire a grant is determined primarily by the grant. For details, see Retiring and revoking grants in the *AWS Key Management Service Developer Guide*.

Related operations:

- CreateGrant
- ListGrants
- ListRetirableGrants
- RevokeGrant

Eventual consistency: The AWS KMS API follows an eventual consistency model. For more information, see AWS KMS eventual consistency.

Request Syntax

```
{
   "DryRun": boolean,
   "GrantId": "string",
   "GrantToken": "string",
   "KeyId": "string"
}
```

Request Parameters

For information about the parameters that are common to all actions, see Common Parameters.

The request accepts the following data in JSON format.

> **Note**
>
> In the following list, the required parameters are described first.

DryRun

Checks if your request will succeed. `DryRun` is an optional parameter.

To learn more about how to use this parameter, see Testing your permissions in the *AWS Key Management Service Developer Guide*.

Type: Boolean

Required: No

GrantId

Identifies the grant to retire. To get the grant ID, use CreateGrant, ListGrants, or ListRetirableGrants.

- Grant ID Example - 0123456789012345678901234567890123456789012345678901234567890123

Type: String

Length Constraints: Minimum length of 1. Maximum length of 128.

Required: No

GrantToken

Identifies the grant to be retired. You can use a grant token to identify a new grant even before it has achieved eventual consistency.

Only the CreateGrant operation returns a grant token. For details, see Grant token and Eventual consistency in the *AWS Key Management Service Developer Guide*.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 8192.

Required: No

KeyId

The key ARN KMS key associated with the grant. To find the key ARN, use the ListKeys operation.

For example: `arn:aws:kms:us-east-2:444455556666:key/1234abcd-12ab-34cd-56ef-1234567890ab`

Type: String

Length Constraints: Minimum length of 1. Maximum length of 2048.

Required: No

Response Elements

If the action is successful, the service sends back an HTTP 200 response with an empty HTTP body.

Errors

For information about the errors that are common to all actions, see Common Error Types.

DependencyTimeoutException

The system timed out while trying to fulfill the request. You can retry the request.

HTTP Status Code: 500

DryRunOperationException

The request was rejected because the DryRun parameter was specified.

HTTP Status Code: 400

InvalidArnException

The request was rejected because a specified ARN, or an ARN in a key policy, is not valid.

HTTP Status Code: 400

InvalidGrantIdException

The request was rejected because the specified `GrantId` is not valid.

HTTP Status Code: 400

InvalidGrantTokenException

The request was rejected because the specified grant token is not valid.

HTTP Status Code: 400

KMSInternalException

The request was rejected because an internal exception occurred. The request can be retried.

HTTP Status Code: 500

KMSInvalidStateException

The request was rejected because the state of the specified resource is not valid for this request.

This exceptions means one of the following:

- The key state of the KMS key is not compatible with the operation.

  To find the key state, use the DescribeKey operation. For more information about which key states are compatible with each AWS KMS operation, see Key states of AWS KMS keys in the AWS Key Management Service Developer Guide .
- For cryptographic operations on KMS keys in custom key stores, this exception represents a general failure with many possible causes. To identify the cause, see the error message that accompanies the exception.

HTTP Status Code: 400

NotFoundException

The request was rejected because the specified entity or resource could not be found.

HTTP Status Code: 400

Examples

Example Request

The following example is formatted for legibility.

```
POST / HTTP/1.1
Host: kms.us-east-2.amazonaws.com
Content-Length: 167
X-Amz-Target: TrentService.RetireGrant
X-Amz-Date: 20161208T233237Z
Content-Type: application/x-amz-json-1.1
Authorization: AWS4-HMAC-SHA256\
 Credential=AKIAI44QH8DHBEXAMPLE/20161208/us-east-2/kms/aws4_request,\
 SignedHeaders=content-type;host;x-amz-date;x-amz-target,\
 Signature=e463f010eb7d997b4f89ae836288a67f362b0afd762fcf242a3f76ba282448dc

{
  "KeyId": "arn:aws:kms:us-
east-2:444455556666:key/1234abcd-12ab-34cd-56ef-1234567890ab",
  "GrantId": "1ea8e6c7d4d49ecf7e4461c792f6a27651d7ff0ee13a724c19e730337faa26b1"
}
```

Example Response

This example illustrates one usage of RetireGrant.

```
HTTP/1.1 200 OK
Server: Server
Date: Thu, 08 Dec 2016 23:32:38 GMT
Content-Type: application/x-amz-json-1.1
Content-Length: 0
Connection: keep-alive
x-amzn-RequestId: 9ad2b038-bd9e-11e6-ace2-6fb96f685e31
```

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface V2
- AWS SDK for .NET V4
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2

- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- AWS SDK for PHP V3
- AWS SDK for Python (Boto3)
- AWS SDK for Ruby V3

RevokeGrant

Deletes the specified grant. You revoke a grant to terminate the permissions that the grant allows. For more information, see Retiring and revoking grants in the AWS Key Management Service Developer Guide .

When you create, retire, or revoke a grant, there might be a brief delay, usually less than five minutes, until the grant is available throughout AWS KMS. This state is known as *eventual consistency*. For details, see Eventual consistency in the AWS Key Management Service Developer Guide .

For detailed information about grants, including grant terminology, see Grants in AWS KMS in the AWS Key Management Service Developer Guide . For examples of creating grants in several programming languages, see Use CreateGrant with an AWS SDK or CLI.

Cross-account use: Yes. To perform this operation on a KMS key in a different AWS account, specify the key ARN in the value of the `KeyId` parameter.

Required permissions: kms:RevokeGrant (key policy).

Related operations:

- CreateGrant
- ListGrants
- ListRetirableGrants
- RetireGrant

Eventual consistency: The AWS KMS API follows an eventual consistency model. For more information, see AWS KMS eventual consistency.

Request Syntax

```
{
   "DryRun": boolean,
   "GrantId": "string",
   "KeyId": "string"
}
```

Request Parameters

For information about the parameters that are common to all actions, see Common Parameters.

The request accepts the following data in JSON format.

> **Note**
>
> In the following list, the required parameters are described first.

GrantId

Identifies the grant to revoke. To get the grant ID, use CreateGrant, ListGrants, or ListRetirableGrants.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 128.

Required: Yes

KeyId

A unique identifier for the KMS key associated with the grant. To get the key ID and key ARN for a KMS key, use ListKeys or DescribeKey.

Specify the key ID or key ARN of the KMS key. To specify a KMS key in a different AWS account, you must use the key ARN.

For example:

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

To get the key ID and key ARN for a KMS key, use ListKeys or DescribeKey.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 2048.

Required: Yes

DryRun

Checks if your request will succeed. `DryRun` is an optional parameter.

To learn more about how to use this parameter, see Testing your permissions in the *AWS Key Management Service Developer Guide*.

Type: Boolean

Required: No

Response Elements

If the action is successful, the service sends back an HTTP 200 response with an empty HTTP body.

Errors

For information about the errors that are common to all actions, see Common Error Types.

DependencyTimeoutException

The system timed out while trying to fulfill the request. You can retry the request.

HTTP Status Code: 500

DryRunOperationException

The request was rejected because the DryRun parameter was specified.

HTTP Status Code: 400

InvalidArnException

The request was rejected because a specified ARN, or an ARN in a key policy, is not valid.

HTTP Status Code: 400

InvalidGrantIdException

The request was rejected because the specified `GrantId` is not valid.

HTTP Status Code: 400

KMSInternalException

The request was rejected because an internal exception occurred. The request can be retried.

HTTP Status Code: 500

KMSInvalidStateException

The request was rejected because the state of the specified resource is not valid for this request.

This exceptions means one of the following:

- The key state of the KMS key is not compatible with the operation.

  To find the key state, use the DescribeKey operation. For more information about which key states are compatible with each AWS KMS operation, see Key states of AWS KMS keys in the AWS Key Management Service Developer Guide .
- For cryptographic operations on KMS keys in custom key stores, this exception represents a general failure with many possible causes. To identify the cause, see the error message that accompanies the exception.

HTTP Status Code: 400

NotFoundException

The request was rejected because the specified entity or resource could not be found.

HTTP Status Code: 400

Examples

Example Request

The following example is formatted for legibility.

```
POST / HTTP/1.1
Host: kms.us-west-2.amazonaws.com
Content-Length: 128
X-Amz-Target: TrentService.RevokeGrant
X-Amz-Date: 20161210T000739Z
Content-Type: application/x-amz-json-1.1
Authorization: AWS4-HMAC-SHA256\
 Credential=AKIAI44QH8DHBEXAMPLE/20161210/us-west-2/kms/aws4_request,\
 SignedHeaders=content-type;host;x-amz-date;x-amz-target,\
 Signature=3f4073c96c38c8bc006b3a74a67fb2108cfe2d6ff23f96f09047924919806a7d

{
```

```
    "KeyId": "1234abcd-12ab-34cd-56ef-1234567890ab",
    "GrantId": "f271e8328717f8bde5d03f4981f06a6b3fc18bcae2da12ac38bd9186e7925d11"
}
```

Example Response

This example illustrates one usage of RevokeGrant.

```
HTTP/1.1 200 OK
Server: Server
Date: Sat, 10 Dec 2016 00:07:40 GMT
Content-Type: application/x-amz-json-1.1
Content-Length: 0
Connection: keep-alive
x-amzn-RequestId: aa49887b-be6c-11e6-b749-7394871b1b43
```

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface V2
- AWS SDK for .NET V4
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- AWS SDK for PHP V3
- AWS SDK for Python (Boto3)
- AWS SDK for Ruby V3

RotateKeyOnDemand

Immediately initiates rotation of the key material of the specified symmetric encryption KMS key.

You can perform on-demand rotation of the key material in customer managed KMS keys, regardless of whether or not automatic key rotation is enabled. On-demand rotations do not change existing automatic rotation schedules. For example, consider a KMS key that has automatic key rotation enabled with a rotation period of 730 days. If the key is scheduled to automatically rotate on April 14, 2024, and you perform an on-demand rotation on April 10, 2024, the key will automatically rotate, as scheduled, on April 14, 2024 and every 730 days thereafter.

> **Note**
>
> You can perform on-demand key rotation a **maximum of 25 times** per KMS key. You can use the AWS KMS console to view the number of remaining on-demand rotations available for a KMS key.

You can use GetKeyRotationStatus to identify any in progress on-demand rotations. You can use ListKeyRotations to identify the date that completed on-demand rotations were performed. You can monitor rotation of the key material for your KMS keys in AWS CloudTrail and Amazon CloudWatch.

On-demand key rotation is supported only on symmetric encryption KMS keys. You cannot perform on-demand rotation of asymmetric KMS keys, HMAC KMS keys, or KMS keys in a custom key store. When you initiate on-demand key rotation on a symmetric encryption KMS key with imported key material, you must have already imported new key material and that key material's state should be `PENDING_ROTATION`. Use the `ListKeyRotations` operation to check the state of all key materials associated with a KMS key. To perform on-demand rotation of a set of related multi-Region keys, import new key material in the primary Region key, import the same key material in each replica Region key, and invoke the on-demand rotation on the primary Region key.

You cannot initiate on-demand rotation of AWS managed KMS keys. AWS KMS always rotates the key material of AWS managed keys every year. Rotation of AWS owned KMS keys is managed by the AWS service that owns the key.

The KMS key that you use for this operation must be in a compatible key state. For details, see Key states of AWS KMS keys in the *AWS Key Management Service Developer Guide*.

Cross-account use: No. You cannot perform this operation on a KMS key in a different AWS account.

Required permissions: kms:RotateKeyOnDemand (key policy)

Related operations:

- EnableKeyRotation
- DisableKeyRotation
- GetKeyRotationStatus
- ImportKeyMaterial
- ListKeyRotations

Eventual consistency: The AWS KMS API follows an eventual consistency model. For more information, see AWS KMS eventual consistency.

Request Syntax

```
{
   "KeyId": "string"
}
```

Request Parameters

For information about the parameters that are common to all actions, see Common Parameters.

The request accepts the following data in JSON format.

> **Note**
>
> In the following list, the required parameters are described first.

KeyId

Identifies a symmetric encryption KMS key. You cannot perform on-demand rotation of asymmetric KMS keys, HMAC KMS keys, multi-Region KMS keys with imported key material, or KMS keys in a custom key store. To perform on-demand rotation of a set of related multi-Region keys, invoke the on-demand rotation on the primary key.

Specify the key ID or key ARN of the KMS key.

For example:

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

To get the key ID and key ARN for a KMS key, use ListKeys or DescribeKey.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 2048.

Required: Yes

Response Syntax

```
{
   "KeyId": "string"
}
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

KeyId

Identifies the symmetric encryption KMS key that you initiated on-demand rotation on.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 2048.

Errors

For information about the errors that are common to all actions, see Common Error Types.

ConflictException

The request was rejected because an automatic rotation of this key is currently in progress or scheduled to begin within the next 20 minutes.

HTTP Status Code: 400

DependencyTimeoutException

The system timed out while trying to fulfill the request. You can retry the request.

HTTP Status Code: 500

DisabledException

The request was rejected because the specified KMS key is not enabled.

HTTP Status Code: 400

InvalidArnException

The request was rejected because a specified ARN, or an ARN in a key policy, is not valid.

HTTP Status Code: 400

KMSInternalException

The request was rejected because an internal exception occurred. The request can be retried.

HTTP Status Code: 500

KMSInvalidStateException

The request was rejected because the state of the specified resource is not valid for this request.

This exceptions means one of the following:

- The key state of the KMS key is not compatible with the operation.

  To find the key state, use the DescribeKey operation. For more information about which key states are compatible with each AWS KMS operation, see Key states of AWS KMS keys in the AWS Key Management Service Developer Guide .
- For cryptographic operations on KMS keys in custom key stores, this exception represents a general failure with many possible causes. To identify the cause, see the error message that accompanies the exception.

HTTP Status Code: 400

LimitExceededException

The request was rejected because a length constraint or quota was exceeded. For more information, see Quotas in the *AWS Key Management Service Developer Guide*.

HTTP Status Code: 400

NotFoundException

The request was rejected because the specified entity or resource could not be found.

HTTP Status Code: 400

UnsupportedOperationException

The request was rejected because a specified parameter is not supported or a specified resource is not valid for this operation.

HTTP Status Code: 400

Examples

Example Request

The following example is formatted for legibility.

```
POST / HTTP/1.1
Host: kms.us-east-2.amazonaws.com
Content-Length: 48
X-Amz-Target: TrentService.RotateKeyOnDemand
X-Amz-Date: 20240405T151426Z
Content-Type: application/x-amz-json-1.1
Authorization: AWS4-HMAC-SHA256\
 Credential=AKIAI44QH8DHBEXAMPLE/20161107/us-east-2/kms/aws4_request,\
 SignedHeaders=content-type;host;x-amz-date;x-amz-target,\
 Signature=4783e177036ca78627fe0cda9dcfdaf4ad7c8312d0e7c3d71d814b0c4cff1c0b

{"KeyId": "1234abcd-12ab-34cd-56ef-1234567890ab"}
```

Example Response

This example illustrates one usage of RotateKeyOnDemand.

```
HTTP/1.1 200 OK
Server: Server
Date: Fri, 05 Apr 2024 15:14:26 GMT
Content-Type: application/x-amz-json-1.1
Content-Length: 0
Connection: keep-alive
x-amzn-RequestId: 2077c3bf-a538-11e6-b6fb-794e83344f84

{"KeyId": "1234abcd-12ab-34cd-56ef-1234567890ab"}
```

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface V2
- AWS SDK for .NET V4
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- AWS SDK for PHP V3
- AWS SDK for Python (Boto3)
- AWS SDK for Ruby V3

ScheduleKeyDeletion

Schedules the deletion of a KMS key. By default, AWS KMS applies a waiting period of 30 days, but you can specify a waiting period of 7-30 days. When this operation is successful, the key state of the KMS key changes to `PendingDeletion` and the key can't be used in any cryptographic operations. It remains in this state for the duration of the waiting period. Before the waiting period ends, you can use CancelKeyDeletion to cancel the deletion of the KMS key. After the waiting period ends, AWS KMS deletes the KMS key, its key material, and all AWS KMS data associated with it, including all aliases that refer to it.

> **Important**
>
> Deleting a KMS key is a destructive and potentially dangerous operation. When a KMS key is deleted, all data that was encrypted under the KMS key is unrecoverable. (The only exception is a multi-Region replica key, or an asymmetric or HMAC KMS key with imported key material.) To prevent the use of a KMS key without deleting it, use DisableKey.

You can schedule the deletion of a multi-Region primary key and its replica keys at any time. However, AWS KMS will not delete a multi-Region primary key with existing replica keys. If you schedule the deletion of a primary key with replicas, its key state changes to `PendingReplicaDeletion` and it cannot be replicated or used in cryptographic operations. This status can continue indefinitely. When the last of its replicas keys is deleted (not just scheduled), the key state of the primary key changes to `PendingDeletion` and its waiting period (`PendingWindowInDays`) begins. For details, see Deleting multi-Region keys in the *AWS Key Management Service Developer Guide*.

When AWS KMS deletes a KMS key from an AWS CloudHSM key store, it makes a best effort to delete the associated key material from the associated AWS CloudHSM cluster. However, you might need to manually delete the orphaned key material from the cluster and its backups. Deleting a KMS key from an external key store has no effect on the associated external key. However, for both types of custom key stores, deleting a KMS key is destructive and irreversible. You cannot decrypt ciphertext encrypted under the KMS key by using only its associated external key or AWS CloudHSM key. Also, you cannot recreate a KMS key in an external key store by creating a new KMS key with the same key material.

For more information about scheduling a KMS key for deletion, see Deleting KMS keys in the *AWS Key Management Service Developer Guide*.

The KMS key that you use for this operation must be in a compatible key state. For details, see Key states of AWS KMS keys in the *AWS Key Management Service Developer Guide*.

Cross-account use: No. You cannot perform this operation on a KMS key in a different AWS account.

Required permissions: kms:ScheduleKeyDeletion (key policy)

Related operations

- CancelKeyDeletion
- DisableKey

Eventual consistency: The AWS KMS API follows an eventual consistency model. For more information, see AWS KMS eventual consistency.

Request Syntax

```
{
   "KeyId": "string",
   "PendingWindowInDays": number
}
```

Request Parameters

For information about the parameters that are common to all actions, see Common Parameters.

The request accepts the following data in JSON format.

> **Note**
>
> In the following list, the required parameters are described first.

KeyId

The unique identifier of the KMS key to delete.

Specify the key ID or key ARN of the KMS key.

For example:

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

To get the key ID and key ARN for a KMS key, use ListKeys or DescribeKey.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 2048.

Required: Yes

PendingWindowInDays

The waiting period, specified in number of days. After the waiting period ends, AWS KMS deletes the KMS key.

If the KMS key is a multi-Region primary key with replica keys, the waiting period begins when the last of its replica keys is deleted. Otherwise, the waiting period begins immediately.

This value is optional. If you include a value, it must be between 7 and 30, inclusive. If you do not include a value, it defaults to 30. You can use the `kms:ScheduleKeyDeletionPendingWindowInDays` condition key to further constrain the values that principals can specify in the `PendingWindowInDays` parameter.

Type: Integer

Valid Range: Minimum value of 7. Maximum value of 30.

Required: No

Response Syntax

```
{
   "DeletionDate": number,
   "KeyId": "string",
   "KeyState": "string",
   "PendingWindowInDays": number
}
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

DeletionDate

The date and time after which AWS KMS deletes the KMS key.

If the KMS key is a multi-Region primary key with replica keys, this field does not appear. The deletion date for the primary key isn't known until its last replica key is deleted.

Type: Timestamp

KeyId

The Amazon Resource Name (key ARN) of the KMS key whose deletion is scheduled.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 2048.

KeyState

The current status of the KMS key.

For more information about how key state affects the use of a KMS key, see Key states of AWS KMS keys in the *AWS Key Management Service Developer Guide*.

Type: String

Valid Values: `Creating | Enabled | Disabled | PendingDeletion | PendingImport | PendingReplicaDeletion | Unavailable | Updating`

PendingWindowInDays

The waiting period before the KMS key is deleted.

If the KMS key is a multi-Region primary key with replicas, the waiting period begins when the last of its replica keys is deleted. Otherwise, the waiting period begins immediately.

Type: Integer

Valid Range: Minimum value of 7. Maximum value of 30.

Errors

For information about the errors that are common to all actions, see Common Error Types.

DependencyTimeoutException

The system timed out while trying to fulfill the request. You can retry the request.

HTTP Status Code: 500

InvalidArnException

The request was rejected because a specified ARN, or an ARN in a key policy, is not valid.

HTTP Status Code: 400

KMSInternalException

The request was rejected because an internal exception occurred. The request can be retried.

HTTP Status Code: 500

KMSInvalidStateException

The request was rejected because the state of the specified resource is not valid for this request.

This exceptions means one of the following:

- The key state of the KMS key is not compatible with the operation.

  To find the key state, use the DescribeKey operation. For more information about which key states are compatible with each AWS KMS operation, see Key states of AWS KMS keys in the AWS Key Management Service Developer Guide .
- For cryptographic operations on KMS keys in custom key stores, this exception represents a general failure with many possible causes. To identify the cause, see the error message that accompanies the exception.

HTTP Status Code: 400

NotFoundException

The request was rejected because the specified entity or resource could not be found.

HTTP Status Code: 400

Examples

The following examples are formatted for legibility.

Example Request

This example illustrates one usage of ScheduleKeyDeletion.

```
POST / HTTP/1.1
Host: kms.us-east-2.amazonaws.com
Content-Length: 75
X-Amz-Target: TrentService.ScheduleKeyDeletion
X-Amz-Date: 20161210T003358Z
Content-Type: application/x-amz-json-1.1
Authorization: AWS4-HMAC-SHA256\
 Credential=AKIAI44QH8DHBEXAMPLE/20161210/us-east-2/kms/aws4_request,\
 SignedHeaders=content-type;host;x-amz-date;x-amz-target,\
 Signature=c42c52cf0e4057e004b73a905b0e5da215f63dd33117e7316f760e6223433abb

{
  "PendingWindowInDays": 7,
  "KeyId": "1234abcd-12ab-34cd-56ef-1234567890ab"
}
```

Example Response

This example illustrates one usage of ScheduleKeyDeletion.

```
HTTP/1.1 200 OK
Server: Server
Date: Sat, 10 Dec 2016 00:33:58 GMT
Content-Type: application/x-amz-json-1.1
Content-Length: 114
Connection: keep-alive
x-amzn-RequestId: 5704ddf7-be70-11e6-b0c0-3343f53dee45

{
  "DeletionDate": 1.4820192E9,
  "KeyId": "arn:aws:kms:us-
east-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
  "PendingWindowInDays": 7
}
```

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface V2
- AWS SDK for .NET V4
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- AWS SDK for PHP V3
- AWS SDK for Python (Boto3)
- AWS SDK for Ruby V3

Sign

Creates a digital signature for a message or message digest by using the private key in an asymmetric signing KMS key. To verify the signature, use the Verify operation, or use the public key in the same asymmetric KMS key outside of AWS KMS. For information about asymmetric KMS keys, see Asymmetric KMS keys in the *AWS Key Management Service Developer Guide*.

Digital signatures are generated and verified by using asymmetric key pair, such as an RSA, ECC, or ML-DSA pair that is represented by an asymmetric KMS key. The key owner (or an authorized user) uses their private key to sign a message. Anyone with the public key can verify that the message was signed with that particular private key and that the message hasn't changed since it was signed.

To use the `Sign` operation, provide the following information:

- Use the `KeyId` parameter to identify an asymmetric KMS key with a `KeyUsage` value of `SIGN_VERIFY`. To get the `KeyUsage` value of a KMS key, use the DescribeKey operation. The caller must have `kms:Sign` permission on the KMS key.
- Use the `Message` parameter to specify the message or message digest to sign. You can submit messages of up to 4096 bytes. To sign a larger message, generate a hash digest of the message, and then provide the hash digest in the `Message` parameter. To indicate whether the message is a full message, a digest, or an ML-DSA EXTERNAL_MU, use the `MessageType` parameter.
- Choose a signing algorithm that is compatible with the KMS key.

Important

When signing a message, be sure to record the KMS key and the signing algorithm. This information is required to verify the signature.

Note

Best practices recommend that you limit the time during which any signature is effective. This deters an attack where the actor uses a signed message to establish validity repeatedly or long after the message is superseded. Signatures do not include a timestamp, but you

can include a timestamp in the signed message to help you detect when its time to refresh the signature.

To verify the signature that this operation generates, use the Verify operation. Or use the GetPublicKey operation to download the public key and then use the public key to verify the signature outside of AWS KMS.

The KMS key that you use for this operation must be in a compatible key state. For details, see Key states of AWS KMS keys in the *AWS Key Management Service Developer Guide*.

Cross-account use: Yes. To perform this operation with a KMS key in a different AWS account, specify the key ARN or alias ARN in the value of the `KeyId` parameter.

Required permissions: kms:Sign (key policy)

Related operations: Verify

Eventual consistency: The AWS KMS API follows an eventual consistency model. For more information, see AWS KMS eventual consistency.

Request Syntax

```
{
   "DryRun": boolean,
   "GrantTokens": [ "string" ],
   "KeyId": "string",
   "Message": blob,
   "MessageType": "string",
   "SigningAlgorithm": "string"
}
```

Request Parameters

For information about the parameters that are common to all actions, see Common Parameters.

The request accepts the following data in JSON format.

Note

In the following list, the required parameters are described first.

KeyId

Identifies an asymmetric KMS key. AWS KMS uses the private key in the asymmetric KMS key to sign the message. The `KeyUsage` type of the KMS key must be `SIGN_VERIFY`. To find the `KeyUsage` of a KMS key, use the DescribeKey operation.

To specify a KMS key, use its key ID, key ARN, alias name, or alias ARN. When using an alias name, prefix it with `"alias/"`. To specify a KMS key in a different AWS account, you must use the key ARN or alias ARN.

For example:

- Key ID: `1234abcd-12ab-34cd-56ef-1234567890ab`
- Key ARN: `arn:aws:kms:us-east-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab`
- Alias name: `alias/ExampleAlias`
- Alias ARN: `arn:aws:kms:us-east-2:111122223333:alias/ExampleAlias`

To get the key ID and key ARN for a KMS key, use ListKeys or DescribeKey. To get the alias name and alias ARN, use ListAliases.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 2048.

Required: Yes

Message

Specifies the message or message digest to sign. Messages can be 0-4096 bytes. To sign a larger message, provide a message digest.

If you provide a message digest, use the `DIGEST` value of `MessageType` to prevent the digest from being hashed again while signing.

Type: Base64-encoded binary data object

Length Constraints: Minimum length of 1. Maximum length of 4096.

Required: Yes

SigningAlgorithm

Specifies the signing algorithm to use when signing the message.

Choose an algorithm that is compatible with the type and size of the specified asymmetric KMS key. When signing with RSA key pairs, RSASSA-PSS algorithms are preferred. We include RSASSA-PKCS1-v1_5 algorithms for compatibility with existing applications.

Type: String

Valid Values: `RSASSA_PSS_SHA_256 | RSASSA_PSS_SHA_384 | RSASSA_PSS_SHA_512 | RSASSA_PKCS1_V1_5_SHA_256 | RSASSA_PKCS1_V1_5_SHA_384 | RSASSA_PKCS1_V1_5_SHA_512 | ECDSA_SHA_256 | ECDSA_SHA_384 | ECDSA_SHA_512 | SM2DSA | ML_DSA_SHAKE_256 | ED25519_SHA_512 | ED25519_PH_SHA_512`

Required: Yes

DryRun

Checks if your request will succeed. `DryRun` is an optional parameter.

To learn more about how to use this parameter, see Testing your permissions in the *AWS Key Management Service Developer Guide*.

Type: Boolean

Required: No

GrantTokens

A list of grant tokens.

Use a grant token when your permission to call this operation comes from a new grant that has not yet achieved *eventual consistency*. For more information, see Grant token and Using a grant token in the *AWS Key Management Service Developer Guide*.

Type: Array of strings

Array Members: Minimum number of 0 items. Maximum number of 10 items.

Length Constraints: Minimum length of 1. Maximum length of 8192.

Required: No

MessageType

Tells AWS KMS whether the value of the `Message` parameter should be hashed as part of the signing algorithm. Use `RAW` for unhashed messages; use `DIGEST` for message digests, which

are already hashed; use EXTERNAL_MU for 64-byte representative μ used in ML-DSA signing as defined in NIST FIPS 204 Section 6.2.

When the value of MessageType is RAW, AWS KMS uses the standard signing algorithm, which begins with a hash function. When the value is DIGEST, AWS KMS skips the hashing step in the signing algorithm. When the value is EXTERNAL_MU AWS KMS skips the concatenated hashing of the public key hash and the message done in the ML-DSA signing algorithm.

> **Important**
>
> Use the DIGEST or EXTERNAL_MU value only when the value of the Message parameter is a message digest. If you use the DIGEST value with an unhashed message, the security of the signing operation can be compromised.

When using ECC_NIST_EDWARDS25519 KMS keys:

- ED25519_SHA_512 signing algorithm requires AWS KMS MessageType:RAW
- ED25519_PH_SHA_512 signing algorithm requires AWS KMS MessageType:DIGEST

> **Important**
>
> When you specify the ED25519_PH_SHA_512 signing algorithm with MessageType:DIGEST, AWS KMS still performs the SHA-512 prehash described in Step 1 of Section 7.8.1 in FIPS 186-5. This means the input is hashed twice: once by you and once by KMS.

When the value of MessageType is DIGEST, the length of the Message value must match the length of hashed messages for the specified signing algorithm.

When the value of MessageType is EXTERNAL_MU the length of the Message value must be 64 bytes.

You can submit a message digest and omit the MessageType or specify RAW so the digest is hashed again while signing. However, this can cause verification failures when verifying with a system that assumes a single hash.

The hashing algorithm that Sign uses is based on the SigningAlgorithm value.

- Signing algorithms that end in SHA_256 use the SHA_256 hashing algorithm.
- Signing algorithms that end in SHA_384 use the SHA_384 hashing algorithm.
- Signing algorithms that end in SHA_512 use the SHA_512 hashing algorithm.
- Signing algorithms that end in SHAKE_256 use the SHAKE_256 hashing algorithm.
- SM2DSA uses the SM3 hashing algorithm. For details, see Offline verification with SM2 key pairs.

Type: String

Valid Values: `RAW | DIGEST | EXTERNAL_MU`

Required: No

Response Syntax

```
{
   "KeyId": "string",
   "Signature": blob,
   "SigningAlgorithm": "string"
}
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

KeyId

The Amazon Resource Name (key ARN) of the asymmetric KMS key that was used to sign the message.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 2048.

Signature

The cryptographic signature that was generated for the message.

- When used with the supported RSA signing algorithms, the encoding of this value is defined by PKCS #1 in RFC 8017.
- When used with the ECDSA_SHA_256, ECDSA_SHA_384, or ECDSA_SHA_512 signing algorithms, this value is a DER-encoded object as defined by ANSI X9.62–2005 and RFC 3279 Section 2.2.3. This is the most commonly used signature format and is appropriate for most uses.

When you use the HTTP API or the AWS CLI, the value is Base64-encoded. Otherwise, it is not Base64-encoded.

Type: Base64-encoded binary data object

Length Constraints: Minimum length of 1. Maximum length of 6144.

SigningAlgorithm

The signing algorithm that was used to sign the message.

Type: String

Valid Values: `RSASSA_PSS_SHA_256 | RSASSA_PSS_SHA_384 | RSASSA_PSS_SHA_512 | RSASSA_PKCS1_V1_5_SHA_256 | RSASSA_PKCS1_V1_5_SHA_384 | RSASSA_PKCS1_V1_5_SHA_512 | ECDSA_SHA_256 | ECDSA_SHA_384 | ECDSA_SHA_512 | SM2DSA | ML_DSA_SHAKE_256 | ED25519_SHA_512 | ED25519_PH_SHA_512`

Errors

For information about the errors that are common to all actions, see Common Error Types.

DependencyTimeoutException

The system timed out while trying to fulfill the request. You can retry the request.

HTTP Status Code: 500

DisabledException

The request was rejected because the specified KMS key is not enabled.

HTTP Status Code: 400

DryRunOperationException

The request was rejected because the DryRun parameter was specified.

HTTP Status Code: 400

InvalidGrantTokenException

The request was rejected because the specified grant token is not valid.

HTTP Status Code: 400

InvalidKeyUsageException

The request was rejected for one of the following reasons:

- The `KeyUsage` value of the KMS key is incompatible with the API operation.
- The encryption algorithm or signing algorithm specified for the operation is incompatible with the type of key material in the KMS key (`KeySpec`).

For encrypting, decrypting, re-encrypting, and generating data keys, the `KeyUsage` must be `ENCRYPT_DECRYPT`. For signing and verifying messages, the `KeyUsage` must be `SIGN_VERIFY`. For generating and verifying message authentication codes (MACs), the `KeyUsage` must be `GENERATE_VERIFY_MAC`. For deriving key agreement secrets, the `KeyUsage` must be `KEY_AGREEMENT`. For encapsulating and decapsulating shared secrets, the `KeyUsage` must be `ENCAPSULATE_DECAPSULATE`. To find the `KeyUsage` of a KMS key, use the DescribeKey operation.

To find the encryption or signing algorithms supported for a particular KMS key, use the DescribeKey operation.

HTTP Status Code: 400

KeyUnavailableException

The request was rejected because the specified KMS key was not available. You can retry the request.

HTTP Status Code: 500

KMSInternalException

The request was rejected because an internal exception occurred. The request can be retried.

HTTP Status Code: 500

KMSInvalidStateException

The request was rejected because the state of the specified resource is not valid for this request.

This exceptions means one of the following:

- The key state of the KMS key is not compatible with the operation.

  To find the key state, use the DescribeKey operation. For more information about which key states are compatible with each AWS KMS operation, see Key states of AWS KMS keys in the AWS Key Management Service Developer Guide .
- For cryptographic operations on KMS keys in custom key stores, this exception represents a general failure with many possible causes. To identify the cause, see the error message that accompanies the exception.

HTTP Status Code: 400

NotFoundException

The request was rejected because the specified entity or resource could not be found.

HTTP Status Code: 400

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface V2
- AWS SDK for .NET V4
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- AWS SDK for PHP V3
- AWS SDK for Python (Boto3)
- AWS SDK for Ruby V3

TagResource

Adds or edits tags on a customer managed key.

> **Note**
>
> Tagging or untagging a KMS key can allow or deny permission to the KMS key. For details, see ABAC for AWS KMS in the *AWS Key Management Service Developer Guide*.

Each tag consists of a tag key and a tag value, both of which are case-sensitive strings. The tag value can be an empty (null) string. To add a tag, specify a new tag key and a tag value. To edit a tag, specify an existing tag key and a new tag value.

You can use this operation to tag a customer managed key, but you cannot tag an AWS managed key, an AWS owned key, a custom key store, or an alias.

You can also add tags to a KMS key while creating it (CreateKey) or replicating it (ReplicateKey).

For information about using tags in AWS KMS, see Tagging keys. For general information about tags, including the format and syntax, see Tagging AWS resources in the *Amazon Web Services General Reference*.

The KMS key that you use for this operation must be in a compatible key state. For details, see Key states of AWS KMS keys in the *AWS Key Management Service Developer Guide*.

Cross-account use: No. You cannot perform this operation on a KMS key in a different AWS account.

Required permissions: kms:TagResource (key policy)

Related operations

- CreateKey
- ListResourceTags
- ReplicateKey
- UntagResource

Eventual consistency: The AWS KMS API follows an eventual consistency model. For more information, see AWS KMS eventual consistency.

Request Syntax

```
{
   "KeyId": "string",
   "Tags": [
      {
         "TagKey": "string",
         "TagValue": "string"
      }
   ]
}
```

Request Parameters

For information about the parameters that are common to all actions, see Common Parameters.

The request accepts the following data in JSON format.

> **Note**
>
> In the following list, the required parameters are described first.

KeyId

Identifies a customer managed key in the account and Region.

Specify the key ID or key ARN of the KMS key.

For example:

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

To get the key ID and key ARN for a KMS key, use ListKeys or DescribeKey.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 2048.

Required: Yes

Tags

One or more tags. Each tag consists of a tag key and a tag value. The tag value can be an empty (null) string.

> **Important**
>
> Do not include confidential or sensitive information in this field. This field may be displayed in plaintext in CloudTrail logs and other output.

You cannot have more than one tag on a KMS key with the same tag key. If you specify an existing tag key with a different tag value, AWS KMS replaces the current tag value with the specified one.

Type: Array of Tag objects

Required: Yes

Response Elements

If the action is successful, the service sends back an HTTP 200 response with an empty HTTP body.

Errors

For information about the errors that are common to all actions, see Common Error Types.

InvalidArnException

The request was rejected because a specified ARN, or an ARN in a key policy, is not valid.

HTTP Status Code: 400

KMSInternalException

The request was rejected because an internal exception occurred. The request can be retried.

HTTP Status Code: 500

KMSInvalidStateException

The request was rejected because the state of the specified resource is not valid for this request.

This exceptions means one of the following:

- The key state of the KMS key is not compatible with the operation.

  To find the key state, use the DescribeKey operation. For more information about which key states are compatible with each AWS KMS operation, see Key states of AWS KMS keys in the AWS Key Management Service Developer Guide .
- For cryptographic operations on KMS keys in custom key stores, this exception represents a general failure with many possible causes. To identify the cause, see the error message that accompanies the exception.

HTTP Status Code: 400

LimitExceededException

The request was rejected because a length constraint or quota was exceeded. For more information, see Quotas in the *AWS Key Management Service Developer Guide*.

HTTP Status Code: 400

NotFoundException

The request was rejected because the specified entity or resource could not be found.

HTTP Status Code: 400

TagException

The request was rejected because one or more tags are not valid.

HTTP Status Code: 400

Examples

Example Request

The following example is formatted for legibility.

```
POST / HTTP/1.1
Host: kms.us-east-2.amazonaws.com
Content-Length: 102
X-Amz-Target: TrentService.TagResource
X-Amz-Date: 20170109T200202Z
```

```
Content-Type: application/x-amz-json-1.1
Authorization: AWS4-HMAC-SHA256\
 Credential=AKIAI44QH8DHBEXAMPLE/20170109/us-east-2/kms/aws4_request,\
 SignedHeaders=content-type;host;x-amz-date;x-amz-target,\
 Signature=5a5e6b9950567ea2b9ead41df706fd8f3e4a900553957c5c7f1992daaa67b8ff

{
  "KeyId": "1234abcd-12ab-34cd-56ef-1234567890ab",
  "Tags": [{
    "TagKey": "Purpose",
    "TagValue": "Test"
  }]
}
```

Example Response

This example illustrates one usage of TagResource.

```
HTTP/1.1 200 OK
Server: Server
Date: Mon, 09 Jan 2017 20:02:03 GMT
Content-Type: application/x-amz-json-1.1
Content-Length: 0
Connection: keep-alive
x-amzn-RequestId: 7ce02bcb-d6a6-11e6-bfed-ebe31947a596
```

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface V2
- AWS SDK for .NET V4
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin

- AWS SDK for PHP V3
- AWS SDK for Python (Boto3)
- AWS SDK for Ruby V3

UntagResource

Deletes tags from a customer managed key. To delete a tag, specify the tag key and the KMS key.

> **Note**
>
> Tagging or untagging a KMS key can allow or deny permission to the KMS key. For details, see ABAC for AWS KMS in the *AWS Key Management Service Developer Guide*.

When it succeeds, the `UntagResource` operation doesn't return any output. Also, if the specified tag key isn't found on the KMS key, it doesn't throw an exception or return a response. To confirm that the operation worked, use the ListResourceTags operation.

For information about using tags in AWS KMS, see Tagging keys. For general information about tags, including the format and syntax, see Tagging AWS resources in the *Amazon Web Services General Reference*.

The KMS key that you use for this operation must be in a compatible key state. For details, see Key states of AWS KMS keys in the *AWS Key Management Service Developer Guide*.

Cross-account use: No. You cannot perform this operation on a KMS key in a different AWS account.

Required permissions: kms:UntagResource (key policy)

Related operations

- CreateKey
- ListResourceTags
- ReplicateKey
- TagResource

Eventual consistency: The AWS KMS API follows an eventual consistency model. For more information, see AWS KMS eventual consistency.

Request Syntax

```
{
```

```
    "KeyId": "string",
    "TagKeys": [ "string" ]
}
```

Request Parameters

For information about the parameters that are common to all actions, see Common Parameters.

The request accepts the following data in JSON format.

> **Note**
>
> In the following list, the required parameters are described first.

KeyId

Identifies the KMS key from which you are removing tags.

Specify the key ID or key ARN of the KMS key.

For example:

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

To get the key ID and key ARN for a KMS key, use ListKeys or DescribeKey.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 2048.

Required: Yes

TagKeys

One or more tag keys. Specify only the tag keys, not the tag values.

Type: Array of strings

Length Constraints: Minimum length of 1. Maximum length of 128.

Required: Yes

Response Elements

If the action is successful, the service sends back an HTTP 200 response with an empty HTTP body.

Errors

For information about the errors that are common to all actions, see Common Error Types.

InvalidArnException

The request was rejected because a specified ARN, or an ARN in a key policy, is not valid.

HTTP Status Code: 400

KMSInternalException

The request was rejected because an internal exception occurred. The request can be retried.

HTTP Status Code: 500

KMSInvalidStateException

The request was rejected because the state of the specified resource is not valid for this request.

This exceptions means one of the following:

- The key state of the KMS key is not compatible with the operation.

  To find the key state, use the DescribeKey operation. For more information about which key states are compatible with each AWS KMS operation, see Key states of AWS KMS keys in the AWS Key Management Service Developer Guide .
- For cryptographic operations on KMS keys in custom key stores, this exception represents a general failure with many possible causes. To identify the cause, see the error message that accompanies the exception.

HTTP Status Code: 400

NotFoundException

The request was rejected because the specified entity or resource could not be found.

HTTP Status Code: 400

TagException

The request was rejected because one or more tags are not valid.

HTTP Status Code: 400

Examples

Example Request

The following example is formatted for legibility.

```
POST / HTTP/1.1
Host: kms.us-east-2.amazonaws.com
Content-Length: 87
X-Amz-Target: TrentService.UntagResource
X-Amz-Date: 20170109T200704Z
Content-Type: application/x-amz-json-1.1
Authorization: AWS4-HMAC-SHA256\
 Credential=AKIAI44QH8DHBEXAMPLE/20170109/us-east-2/kms/aws4_request,\
 SignedHeaders=content-type;host;x-amz-date;x-amz-target,\
 Signature=f1c9c01e545fa02e2dba096b66d5f697800a1b8e06a1776058206dc393b8d1b4

{
  "KeyId": "1234abcd-12ab-34cd-56ef-1234567890ab",
  "TagKeys": [
    "Purpose",
    "CostCenter"
  ]
}
```

Example Response

This example illustrates one usage of UntagResource.

```
HTTP/1.1 200 OK
Server: Server
Date: Mon, 09 Jan 2017 20:07:04 GMT
Content-Type: application/x-amz-json-1.1
Content-Length: 0
Connection: keep-alive
x-amzn-RequestId: 30b417a1-d6a7-11e6-a164-b5365990e84e
```

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface V2
- AWS SDK for .NET V4
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- AWS SDK for PHP V3
- AWS SDK for Python (Boto3)
- AWS SDK for Ruby V3

UpdateAlias

Associates an existing AWS KMS alias with a different KMS key. Each alias is associated with only one KMS key at a time, although a KMS key can have multiple aliases. The alias and the KMS key must be in the same AWS account and Region.

> **Note**
>
> Adding, deleting, or updating an alias can allow or deny permission to the KMS key. For details, see ABAC for AWS KMS in the *AWS Key Management Service Developer Guide*.

The current and new KMS key must be the same type (both symmetric or both asymmetric or both HMAC), and they must have the same key usage. This restriction prevents errors in code that uses aliases. If you must assign an alias to a different type of KMS key, use DeleteAlias to delete the old alias and CreateAlias to create a new alias.

You cannot use `UpdateAlias` to change an alias name. To change an alias name, use DeleteAlias to delete the old alias and CreateAlias to create a new alias.

Because an alias is not a property of a KMS key, you can create, update, and delete the aliases of a KMS key without affecting the KMS key. Also, aliases do not appear in the response from the DescribeKey operation. To get the aliases of all KMS keys in the account, use the ListAliases operation.

The KMS key that you use for this operation must be in a compatible key state. For details, see Key states of AWS KMS keys in the *AWS Key Management Service Developer Guide*.

Cross-account use: No. You cannot perform this operation on a KMS key in a different AWS account.

Required permissions

- kms:UpdateAlias on the alias (IAM policy).
- kms:UpdateAlias on the current KMS key (key policy).
- kms:UpdateAlias on the new KMS key (key policy).

For details, see Controlling access to aliases in the *AWS Key Management Service Developer Guide*.

Related operations:

- CreateAlias
- DeleteAlias
- ListAliases

Eventual consistency: The AWS KMS API follows an eventual consistency model. For more information, see AWS KMS eventual consistency.

Request Syntax

```
{
   "AliasName": "string",
   "TargetKeyId": "string"
}
```

Request Parameters

For information about the parameters that are common to all actions, see Common Parameters.

The request accepts the following data in JSON format.

> **Note**
>
> In the following list, the required parameters are described first.

AliasName

Identifies the alias that is changing its KMS key. This value must begin with `alias/` followed by the alias name, such as `alias/ExampleAlias`. You cannot use `UpdateAlias` to change the alias name.

> **Important**
>
> Do not include confidential or sensitive information in this field. This field may be displayed in plaintext in CloudTrail logs and other output.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 256.

Pattern: `^alias/[a-zA-Z0-9/_-]+$`

Required: Yes

TargetKeyId

Identifies the customer managed key to associate with the alias. You don't have permission to associate an alias with an AWS managed key.

The KMS key must be in the same AWS account and Region as the alias. Also, the new target KMS key must be the same type as the current target KMS key (both symmetric or both asymmetric or both HMAC) and they must have the same key usage.

Specify the key ID or key ARN of the KMS key.

For example:

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

To get the key ID and key ARN for a KMS key, use ListKeys or DescribeKey.

To verify that the alias is mapped to the correct KMS key, use ListAliases.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 2048.

Required: Yes

Response Elements

If the action is successful, the service sends back an HTTP 200 response with an empty HTTP body.

Errors

For information about the errors that are common to all actions, see Common Error Types.

DependencyTimeoutException

The system timed out while trying to fulfill the request. You can retry the request.

HTTP Status Code: 500

KMSInternalException

The request was rejected because an internal exception occurred. The request can be retried.

HTTP Status Code: 500

KMSInvalidStateException

The request was rejected because the state of the specified resource is not valid for this request.

This exceptions means one of the following:

- The key state of the KMS key is not compatible with the operation.

  To find the key state, use the DescribeKey operation. For more information about which key states are compatible with each AWS KMS operation, see Key states of AWS KMS keys in the AWS Key Management Service Developer Guide .
- For cryptographic operations on KMS keys in custom key stores, this exception represents a general failure with many possible causes. To identify the cause, see the error message that accompanies the exception.

HTTP Status Code: 400

LimitExceededException

The request was rejected because a length constraint or quota was exceeded. For more information, see Quotas in the *AWS Key Management Service Developer Guide*.

HTTP Status Code: 400

NotFoundException

The request was rejected because the specified entity or resource could not be found.

HTTP Status Code: 400

Examples

Example Request

The following example is formatted for legibility.

```
POST / HTTP/1.1
Host: kms.us-east-2.amazonaws.com
Content-Length: 90
X-Amz-Target: TrentService.UpdateAlias
X-Amz-Date: 20161212T193252Z
Content-Type: application/x-amz-json-1.1
Authorization: AWS4-HMAC-SHA256\
 Credential=AKIAI44QH8DHBEXAMPLE/20161212/us-east-2/kms/aws4_request,\
 SignedHeaders=content-type;host;x-amz-date;x-amz-target,\
 Signature=3d6375048a5917aff38f25b92e66bceb16b29562193f7ab7f869b4c53f115c20

{
  "TargetKeyId": "1234abcd-12ab-34cd-56ef-1234567890ab",
  "AliasName": "alias/ExampleAlias"
}
```

Example Response

This example illustrates one usage of UpdateAlias.

```
HTTP/1.1 200 OK
Server: Server
Date: Mon, 12 Dec 2016 19:32:53 GMT
Content-Type: application/x-amz-json-1.1
Content-Length: 0
Connection: keep-alive
x-amzn-RequestId: c64706c8-c0a1-11e6-b0c0-3343f53dee45
```

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface V2

- AWS SDK for .NET V4
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- AWS SDK for PHP V3
- AWS SDK for Python (Boto3)
- AWS SDK for Ruby V3

UpdateCustomKeyStore

Changes the properties of a custom key store. You can use this operation to change the properties of an AWS CloudHSM key store or an external key store.

Use the required `CustomKeyStoreId` parameter to identify the custom key store. Use the remaining optional parameters to change its properties. This operation does not return any property values. To verify the updated property values, use the DescribeCustomKeyStores operation.

This operation is part of the custom key stores feature in AWS KMS, which combines the convenience and extensive integration of AWS KMS with the isolation and control of a key store that you own and manage.

Important

When updating the properties of an external key store, verify that the updated settings connect your key store, via the external key store proxy, to the same external key manager as the previous settings, or to a backup or snapshot of the external key manager with the same cryptographic keys. If the updated connection settings fail, you can fix them and retry, although an extended delay might disrupt AWS services. However, if AWS KMS permanently loses its access to cryptographic keys, ciphertext encrypted under those keys is unrecoverable.

Note

For external key stores:
Some external key managers provide a simpler method for updating an external key store. For details, see your external key manager documentation.
When updating an external key store in the AWS KMS console, you can upload a JSON-based proxy configuration file with the desired values. You cannot upload the proxy configuration file to the `UpdateCustomKeyStore` operation. However, you can use the file to help you determine the correct values for the `UpdateCustomKeyStore` parameters.

For an AWS CloudHSM key store, you can use this operation to change the custom key store friendly name (`NewCustomKeyStoreName`), to tell AWS KMS about a change to the `kmsuser`

crypto user password (`KeyStorePassword`), or to associate the custom key store with a different, but related, AWS CloudHSM cluster (`CloudHsmClusterId`). To update most properties of an AWS CloudHSM key store, the `ConnectionState` of the AWS CloudHSM key store must be `DISCONNECTED`. However, you can update the `CustomKeyStoreName` of an AWS CloudHSM key store when it is in the `CONNECTED` or `DISCONNECTED` state.

For an external key store, you can use this operation to change the custom key store friendly name (`NewCustomKeyStoreName`), or to tell AWS KMS about a change to the external key store proxy authentication credentials (`XksProxyAuthenticationCredential`), connection method (`XksProxyConnectivity`), external proxy endpoint (`XksProxyUriEndpoint`) and path (`XksProxyUriPath`). For external key stores with an `XksProxyConnectivity` of `VPC_ENDPOINT_SERVICE`, you can also update the Amazon VPC endpoint service name (`XksProxyVpcEndpointServiceName`). To update most properties of an external key store, the `ConnectionState` of the external key store must be `DISCONNECTED`. However, you can update the `CustomKeyStoreName`, `XksProxyAuthenticationCredential`, and `XksProxyUriPath` of an external key store when it is in the CONNECTED or DISCONNECTED state.

If your update requires a `DISCONNECTED` state, before using `UpdateCustomKeyStore`, use the DisconnectCustomKeyStore operation to disconnect the custom key store. After the `UpdateCustomKeyStore` operation completes, use the ConnectCustomKeyStore to reconnect the custom key store. To find the `ConnectionState` of the custom key store, use the DescribeCustomKeyStores operation.

Before updating the custom key store, verify that the new values allow AWS KMS to connect the custom key store to its backing key store. For example, before you change the `XksProxyUriPath` value, verify that the external key store proxy is reachable at the new path.

If the operation succeeds, it returns a JSON object with no properties.

Cross-account use: No. You cannot perform this operation on a custom key store in a different AWS account.

Required permissions: kms:UpdateCustomKeyStore (IAM policy)

Related operations:

- ConnectCustomKeyStore
- CreateCustomKeyStore
- DeleteCustomKeyStore

- DescribeCustomKeyStores
- DisconnectCustomKeyStore

Eventual consistency: The AWS KMS API follows an eventual consistency model. For more information, see AWS KMS eventual consistency.

Request Syntax

```
{
   "CloudHsmClusterId": "string",
   "CustomKeyStoreId": "string",
   "KeyStorePassword": "string",
   "NewCustomKeyStoreName": "string",
   "XksProxyAuthenticationCredential": {
      "AccessKeyId": "string",
      "RawSecretAccessKey": "string"
   },
   "XksProxyConnectivity": "string",
   "XksProxyUriEndpoint": "string",
   "XksProxyUriPath": "string",
   "XksProxyVpcEndpointServiceName": "string",
   "XksProxyVpcEndpointServiceOwner": "string"
}
```

Request Parameters

For information about the parameters that are common to all actions, see Common Parameters.

The request accepts the following data in JSON format.

> **Note**
>
> In the following list, the required parameters are described first.

CustomKeyStoreId

Identifies the custom key store that you want to update. Enter the ID of the custom key store. To find the ID of a custom key store, use the DescribeCustomKeyStores operation.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 64.

Required: Yes

CloudHsmClusterId

Associates the custom key store with a related AWS CloudHSM cluster. This parameter is valid only for custom key stores with a `CustomKeyStoreType` of `AWS_CLOUDHSM`.

Enter the cluster ID of the cluster that you used to create the custom key store or a cluster that shares a backup history and has the same cluster certificate as the original cluster. You cannot use this parameter to associate a custom key store with an unrelated cluster. In addition, the replacement cluster must fulfill the requirements for a cluster associated with a custom key store. To view the cluster certificate of a cluster, use the DescribeClusters operation.

To change this value, the AWS CloudHSM key store must be disconnected.

Type: String

Length Constraints: Minimum length of 19. Maximum length of 24.

Pattern: `cluster-[2-7a-zA-Z]{11,16}`

Required: No

KeyStorePassword

Enter the current password of the `kmsuser` crypto user (CU) in the AWS CloudHSM cluster that is associated with the custom key store. This parameter is valid only for custom key stores with a `CustomKeyStoreType` of `AWS_CLOUDHSM`.

This parameter tells AWS KMS the current password of the `kmsuser` crypto user (CU). It does not set or change the password of any users in the AWS CloudHSM cluster.

To change this value, the AWS CloudHSM key store must be disconnected.

Type: String

Length Constraints: Minimum length of 7. Maximum length of 32.

Required: No

NewCustomKeyStoreName

Changes the friendly name of the custom key store to the value that you specify. The custom key store name must be unique in the AWS account.

> **Important**
>
> Do not include confidential or sensitive information in this field. This field may be displayed in plaintext in CloudTrail logs and other output.

To change this value, the custom key store can be connected or disconnected.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 256.

Required: No

XksProxyAuthenticationCredential

Changes the credentials that AWS KMS uses to sign requests to the external key store proxy (XKS proxy). This parameter is valid only for custom key stores with a `CustomKeyStoreType` of `EXTERNAL_KEY_STORE`.

You must specify both the `AccessKeyId` and `SecretAccessKey` value in the authentication credential, even if you are only updating one value.

This parameter doesn't establish or change your authentication credentials on the proxy. It just tells AWS KMS the credential that you established with your external key store proxy. For example, if you rotate the credential on your external key store proxy, you can use this parameter to update the credential in AWS KMS.

You can change this value when the external key store is connected or disconnected.

Type: XksProxyAuthenticationCredentialType object

Required: No

XksProxyConnectivity

Changes the connectivity setting for the external key store. To indicate that the external key store proxy uses a Amazon VPC endpoint service to communicate with AWS KMS, specify `VPC_ENDPOINT_SERVICE`. Otherwise, specify `PUBLIC_ENDPOINT`.

If you change the `XksProxyConnectivity` to `VPC_ENDPOINT_SERVICE`, you must also change the `XksProxyUriEndpoint` and add an `XksProxyVpcEndpointServiceName` value.

If you change the `XksProxyConnectivity` to `PUBLIC_ENDPOINT`, you must also change the `XksProxyUriEndpoint` and specify a null or empty string for the `XksProxyVpcEndpointServiceName` value.

To change this value, the external key store must be disconnected.

Type: String

Valid Values: `PUBLIC_ENDPOINT | VPC_ENDPOINT_SERVICE`

Required: No

XksProxyUriEndpoint

Changes the URI endpoint that AWS KMS uses to connect to your external key store proxy (XKS proxy). This parameter is valid only for custom key stores with a `CustomKeyStoreType` of `EXTERNAL_KEY_STORE`.

For external key stores with an `XksProxyConnectivity` value of `PUBLIC_ENDPOINT`, the protocol must be HTTPS.

For external key stores with an `XksProxyConnectivity` value of `VPC_ENDPOINT_SERVICE`, specify `https://` followed by the private DNS name associated with the VPC endpoint service. Each external key store must use a different private DNS name.

The combined `XksProxyUriEndpoint` and `XksProxyUriPath` values must be unique in the AWS account and Region.

To change this value, the external key store must be disconnected.

Type: String

Length Constraints: Minimum length of 10. Maximum length of 128.

Pattern: `^https://[a-zA-Z0-9.-]+$`

Required: No

XksProxyUriPath

Changes the base path to the proxy APIs for this external key store. To find this value, see the documentation for your external key manager and external key store proxy (XKS

proxy). This parameter is valid only for custom key stores with a `CustomKeyStoreType` of `EXTERNAL_KEY_STORE`.

The value must start with `/` and must end with `/kms/xks/v1`, where `v1` represents the version of the AWS KMS external key store proxy API. You can include an optional prefix between the required elements such as */example*`/kms/xks/v1`.

The combined `XksProxyUriEndpoint` and `XksProxyUriPath` values must be unique in the AWS account and Region.

You can change this value when the external key store is connected or disconnected.

Type: String

Length Constraints: Minimum length of 10. Maximum length of 128.

Pattern: `^(/[a-zA-Z0-9\/_-]+/kms/xks/v\d{1,2})$|^(/kms/xks/v\d{1,2})$`

Required: No

XksProxyVpcEndpointServiceName

Changes the name that AWS KMS uses to identify the Amazon VPC endpoint service for your external key store proxy (XKS proxy). This parameter is valid when the `CustomKeyStoreType` is `EXTERNAL_KEY_STORE` and the `XksProxyConnectivity` is `VPC_ENDPOINT_SERVICE`.

To change this value, the external key store must be disconnected.

Type: String

Length Constraints: Minimum length of 20. Maximum length of 64.

Pattern: `^(com|eu)\.amazonaws\.vpce\.([a-z]+-){2,3}\d+\.vpce-svc-[0-9a-z]+$`

Required: No

XksProxyVpcEndpointServiceOwner

Changes the AWS account ID that AWS KMS uses to identify the Amazon VPC endpoint service for your external key store proxy (XKS proxy). This parameter is optional. If not specified, the current AWS account ID for the VPC endpoint service will not be updated.

To change this value, the external key store must be disconnected.

Type: String

Length Constraints: Fixed length of 12.

Pattern: `[0-9]{12}`

Required: No

Response Elements

If the action is successful, the service sends back an HTTP 200 response with an empty HTTP body.

Errors

For information about the errors that are common to all actions, see Common Error Types.

CloudHsmClusterInvalidConfigurationException

The request was rejected because the associated AWS CloudHSM cluster did not meet the configuration requirements for an AWS CloudHSM key store.

- The AWS CloudHSM cluster must be configured with private subnets in at least two different Availability Zones in the Region.
- The security group for the cluster (cloudhsm-cluster-*<cluster-id>*-sg) must include inbound rules and outbound rules that allow TCP traffic on ports 2223-2225. The **Source** in the inbound rules and the **Destination** in the outbound rules must match the security group ID. These rules are set by default when you create the AWS CloudHSM cluster. Do not delete or change them. To get information about a particular security group, use the DescribeSecurityGroups operation.
- The AWS CloudHSM cluster must contain at least as many HSMs as the operation requires. To add HSMs, use the AWS CloudHSM CreateHsm operation.

  For the CreateCustomKeyStore, UpdateCustomKeyStore, and CreateKey operations, the AWS CloudHSM cluster must have at least two active HSMs, each in a different Availability Zone. For the ConnectCustomKeyStore operation, the AWS CloudHSM must contain at least one active HSM.

For information about the requirements for an AWS CloudHSM cluster that is associated with an AWS CloudHSM key store, see Assemble the Prerequisites in the *AWS Key Management Service Developer Guide*. For information about creating a private subnet for an AWS CloudHSM cluster,

see Create a Private Subnet in the *AWS CloudHSM User Guide*. For information about cluster security groups, see Configure a Default Security Group in the AWS CloudHSM User Guide .

HTTP Status Code: 400

CloudHsmClusterNotActiveException

The request was rejected because the AWS CloudHSM cluster associated with the AWS CloudHSM key store is not active. Initialize and activate the cluster and try the command again. For detailed instructions, see Getting Started in the *AWS CloudHSM User Guide*.

HTTP Status Code: 400

CloudHsmClusterNotFoundException

The request was rejected because AWS KMS cannot find the AWS CloudHSM cluster with the specified cluster ID. Retry the request with a different cluster ID.

HTTP Status Code: 400

CloudHsmClusterNotRelatedException

The request was rejected because the specified AWS CloudHSM cluster has a different cluster certificate than the original cluster. You cannot use the operation to specify an unrelated cluster for an AWS CloudHSM key store.

Specify an AWS CloudHSM cluster that shares a backup history with the original cluster. This includes clusters that were created from a backup of the current cluster, and clusters that were created from the same backup that produced the current cluster.

AWS CloudHSM clusters that share a backup history have the same cluster certificate. To view the cluster certificate of an AWS CloudHSM cluster, use the DescribeClusters operation.

HTTP Status Code: 400

CustomKeyStoreInvalidStateException

The request was rejected because of the `ConnectionState` of the custom key store. To get the `ConnectionState` of a custom key store, use the DescribeCustomKeyStores operation.

This exception is thrown under the following conditions:

- You requested the ConnectCustomKeyStore operation on a custom key store with a `ConnectionState` of `DISCONNECTING` or `FAILED`. This operation is valid for all other

`ConnectionState` values. To reconnect a custom key store in a `FAILED` state, disconnect it (DisconnectCustomKeyStore), then connect it (`ConnectCustomKeyStore`).

- You requested the CreateKey operation in a custom key store that is not connected. This operations is valid only when the custom key store `ConnectionState` is `CONNECTED`.
- You requested the DisconnectCustomKeyStore operation on a custom key store with a `ConnectionState` of `DISCONNECTING` or `DISCONNECTED`. This operation is valid for all other `ConnectionState` values.
- You requested the UpdateCustomKeyStore or DeleteCustomKeyStore operation on a custom key store that is not disconnected. `UpdateCustomKeyStore` can be called on a custom key store in the `CONNECTED` state only to update `NewCustomKeyStoreName`. For all other properties, the custom key store `ConnectionState` must be `DISCONNECTED`.
- You requested the GenerateRandom operation in an AWS CloudHSM key store that is not connected. This operation is valid only when the AWS CloudHSM key store `ConnectionState` is `CONNECTED`.

HTTP Status Code: 400

CustomKeyStoreNameInUseException

The request was rejected because the specified custom key store name is already assigned to another custom key store in the account. Try again with a custom key store name that is unique in the account.

HTTP Status Code: 400

CustomKeyStoreNotFoundException

The request was rejected because AWS KMS cannot find a custom key store with the specified key store name or ID.

HTTP Status Code: 400

KMSInternalException

The request was rejected because an internal exception occurred. The request can be retried.

HTTP Status Code: 500

XksProxyIncorrectAuthenticationCredentialException

The request was rejected because the proxy credentials failed to authenticate to the specified external key store proxy. The specified external key store proxy rejected a status request from

AWS KMS due to invalid credentials. This can indicate an error in the credentials or in the identification of the external key store proxy.

HTTP Status Code: 400

XksProxyInvalidConfigurationException

The request was rejected because the external key store proxy is not configured correctly. To identify the cause, see the error message that accompanies the exception.

HTTP Status Code: 400

XksProxyInvalidResponseException

AWS KMS cannot interpret the response it received from the external key store proxy. The problem might be a poorly constructed response, but it could also be a transient network issue. If you see this error repeatedly, report it to the proxy vendor.

HTTP Status Code: 400

XksProxyUriEndpointInUseException

The request was rejected because the `XksProxyUriEndpoint` is already associated with another external key store in this AWS Region. To identify the cause, see the error message that accompanies the exception.

HTTP Status Code: 400

XksProxyUriInUseException

The request was rejected because the concatenation of the `XksProxyUriEndpoint` and `XksProxyUriPath` is already associated with another external key store in this AWS Region. Each external key store in a Region must use a unique external key store proxy API address.

HTTP Status Code: 400

XksProxyUriUnreachableException

AWS KMS was unable to reach the specified `XksProxyUriPath`. The path must be reachable before you create the external key store or update its settings.

This exception is also thrown when the external key store proxy response to a `GetHealthStatus` request indicates that all external key manager instances are unavailable.

HTTP Status Code: 400

XksProxyVpcEndpointServiceInUseException

The request was rejected because the specified Amazon VPC endpoint service is already associated with another external key store in this AWS Region. Each external key store in a Region must use a different Amazon VPC endpoint service.

HTTP Status Code: 400

XksProxyVpcEndpointServiceInvalidConfigurationException

The request was rejected because the Amazon VPC endpoint service configuration does not fulfill the requirements for an external key store. To identify the cause, see the error message that accompanies the exception and review the requirements for Amazon VPC endpoint service connectivity for an external key store.

HTTP Status Code: 400

XksProxyVpcEndpointServiceNotFoundException

The request was rejected because AWS KMS could not find the specified VPC endpoint service. Use DescribeCustomKeyStores to verify the VPC endpoint service name for the external key store. Also, confirm that the `Allow principals` list for the VPC endpoint service includes the AWS KMS service principal for the Region, such as `cks.kms.us-east-1.amazonaws.com`.

HTTP Status Code: 400

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface V2
- AWS SDK for .NET V4
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin

- AWS SDK for PHP V3
- AWS SDK for Python (Boto3)
- AWS SDK for Ruby V3

UpdateKeyDescription

Updates the description of a KMS key. To see the description of a KMS key, use DescribeKey.

The KMS key that you use for this operation must be in a compatible key state. For details, see Key states of AWS KMS keys in the *AWS Key Management Service Developer Guide*.

Cross-account use: No. You cannot perform this operation on a KMS key in a different AWS account.

Required permissions: kms:UpdateKeyDescription (key policy)

Related operations

- CreateKey
- DescribeKey

Eventual consistency: The AWS KMS API follows an eventual consistency model. For more information, see AWS KMS eventual consistency.

Request Syntax

```
{
   "Description": "string",
   "KeyId": "string"
}
```

Request Parameters

For information about the parameters that are common to all actions, see Common Parameters.

The request accepts the following data in JSON format.

> **Note**
>
> In the following list, the required parameters are described first.

Description

New description for the KMS key.

> **Important**
>
> Do not include confidential or sensitive information in this field. This field may be displayed in plaintext in CloudTrail logs and other output.

Type: String

Length Constraints: Minimum length of 0. Maximum length of 8192.

Required: Yes

KeyId

Updates the description of the specified KMS key.

Specify the key ID or key ARN of the KMS key.

For example:

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

To get the key ID and key ARN for a KMS key, use ListKeys or DescribeKey.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 2048.

Required: Yes

Response Elements

If the action is successful, the service sends back an HTTP 200 response with an empty HTTP body.

Errors

For information about the errors that are common to all actions, see Common Error Types.

DependencyTimeoutException

The system timed out while trying to fulfill the request. You can retry the request.

HTTP Status Code: 500

InvalidArnException

The request was rejected because a specified ARN, or an ARN in a key policy, is not valid.

HTTP Status Code: 400

KMSInternalException

The request was rejected because an internal exception occurred. The request can be retried.

HTTP Status Code: 500

KMSInvalidStateException

The request was rejected because the state of the specified resource is not valid for this request.

This exceptions means one of the following:

- The key state of the KMS key is not compatible with the operation.

  To find the key state, use the DescribeKey operation. For more information about which key states are compatible with each AWS KMS operation, see Key states of AWS KMS keys in the AWS Key Management Service Developer Guide .
- For cryptographic operations on KMS keys in custom key stores, this exception represents a general failure with many possible causes. To identify the cause, see the error message that accompanies the exception.

HTTP Status Code: 400

NotFoundException

The request was rejected because the specified entity or resource could not be found.

HTTP Status Code: 400

Examples

Example Request

The following example is formatted for legibility.

```
POST / HTTP/1.1
Host: kms.us-east-2.amazonaws.com
```

```
Content-Length: 150
X-Amz-Target: TrentService.UpdateKeyDescription
X-Amz-Date: 20161212T201249Z
Content-Type: application/x-amz-json-1.1
Authorization: AWS4-HMAC-SHA256\
 Credential=AKIAI44QH8DHBEXAMPLE/20161212/us-east-2/kms/aws4_request,\
 SignedHeaders=content-type;host;x-amz-date;x-amz-target,\
 Signature=cd81d09965e5df1156eb0416ec8b2e3f9dea9dbc4ca9285b472c319bcbbaec71

{
  "KeyId": "1234abcd-12ab-34cd-56ef-1234567890ab",
  "Description": "Example description that explains what this KMS key is used for."
}
```

Example Response

This example illustrates one usage of UpdateKeyDescription.

```
HTTP/1.1 200 OK
Server: Server
Date: Mon, 12 Dec 2016 20:12:50 GMT
Content-Type: application/x-amz-json-1.1
Content-Length: 0
Connection: keep-alive
x-amzn-RequestId: 5b089880-c0a7-11e6-89c4-3d6791a06780
```

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface V2
- AWS SDK for .NET V4
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin

- AWS SDK for PHP V3
- AWS SDK for Python (Boto3)
- AWS SDK for Ruby V3

UpdatePrimaryRegion

Changes the primary key of a multi-Region key.

This operation changes the replica key in the specified Region to a primary key and changes the former primary key to a replica key. For example, suppose you have a primary key in `us-east-1` and a replica key in `eu-west-2`. If you run `UpdatePrimaryRegion` with a `PrimaryRegion` value of `eu-west-2`, the primary key is now the key in `eu-west-2`, and the key in `us-east-1` becomes a replica key. For details, see Change the primary key in a set of multi-Region keys in the *AWS Key Management Service Developer Guide*.

This operation supports *multi-Region keys*, an AWS KMS feature that lets you create multiple interoperable KMS keys in different AWS Regions. Because these KMS keys have the same key ID, key material, and other metadata, you can use them interchangeably to encrypt data in one AWS Region and decrypt it in a different AWS Region without re-encrypting the data or making a cross-Region call. For more information about multi-Region keys, see Multi-Region keys in AWS KMS in the *AWS Key Management Service Developer Guide*.

The *primary key* of a multi-Region key is the source for properties that are always shared by primary and replica keys, including the key material, key ID, key spec, key usage, key material origin, and automatic key rotation. It's the only key that can be replicated. You cannot delete the primary key until all replica keys are deleted.

The key ID and primary Region that you specify uniquely identify the replica key that will become the primary key. The primary Region must already have a replica key. This operation does not create a KMS key in the specified Region. To find the replica keys, use the DescribeKey operation on the primary key or any replica key. To create a replica key, use the ReplicateKey operation.

You can run this operation while using the affected multi-Region keys in cryptographic operations. This operation should not delay, interrupt, or cause failures in cryptographic operations.

Even after this operation completes, the process of updating the primary Region might still be in progress for a few more seconds. Operations such as `DescribeKey` might display both the old and new primary keys as replicas. The old and new primary keys have a transient key state of `Updating`. The original key state is restored when the update is complete. While the key state is `Updating`, you can use the keys in cryptographic operations, but you cannot replicate the new primary key or perform certain management operations, such as enabling or disabling these keys. For details about the `Updating` key state, see Key states of AWS KMS keys in the *AWS Key Management Service Developer Guide*.

This operation does not return any output. To verify that primary key is changed, use the DescribeKey operation.

Cross-account use: No. You cannot use this operation in a different AWS account.

Required permissions:

- `kms:UpdatePrimaryRegion` on the current primary key (in the primary key's Region). Include this permission primary key's key policy.
- `kms:UpdatePrimaryRegion` on the current replica key (in the replica key's Region). Include this permission in the replica key's key policy.

Related operations

- CreateKey
- ReplicateKey

Eventual consistency: The AWS KMS API follows an eventual consistency model. For more information, see AWS KMS eventual consistency.

Request Syntax

```
{
   "KeyId": "string",
   "PrimaryRegion": "string"
}
```

Request Parameters

For information about the parameters that are common to all actions, see Common Parameters.

The request accepts the following data in JSON format.

> **Note**
>
> In the following list, the required parameters are described first.

KeyId

Identifies the current primary key. When the operation completes, this KMS key will be a replica key.

Specify the key ID or key ARN of a multi-Region primary key.

For example:

- Key ID: `mrk-1234abcd12ab34cd56ef1234567890ab`
- Key ARN: `arn:aws:kms:us-east-2:111122223333:key/mrk-1234abcd12ab34cd56ef1234567890ab`

To get the key ID and key ARN for a KMS key, use ListKeys or DescribeKey.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 2048.

Required: Yes

PrimaryRegion

The AWS Region of the new primary key. Enter the Region ID, such as `us-east-1` or `ap-southeast-2`. There must be an existing replica key in this Region.

When the operation completes, the multi-Region key in this Region will be the primary key.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 32.

Pattern: `^([a-z]+-){2,3}\d+$`

Required: Yes

Response Elements

If the action is successful, the service sends back an HTTP 200 response with an empty HTTP body.

Errors

For information about the errors that are common to all actions, see Common Error Types.

DisabledException

The request was rejected because the specified KMS key is not enabled.

HTTP Status Code: 400

InvalidArnException

The request was rejected because a specified ARN, or an ARN in a key policy, is not valid.

HTTP Status Code: 400

KMSInternalException

The request was rejected because an internal exception occurred. The request can be retried.

HTTP Status Code: 500

KMSInvalidStateException

The request was rejected because the state of the specified resource is not valid for this request.

This exceptions means one of the following:

- The key state of the KMS key is not compatible with the operation.

  To find the key state, use the DescribeKey operation. For more information about which key states are compatible with each AWS KMS operation, see Key states of AWS KMS keys in the AWS Key Management Service Developer Guide .
- For cryptographic operations on KMS keys in custom key stores, this exception represents a general failure with many possible causes. To identify the cause, see the error message that accompanies the exception.

HTTP Status Code: 400

NotFoundException

The request was rejected because the specified entity or resource could not be found.

HTTP Status Code: 400

UnsupportedOperationException

The request was rejected because a specified parameter is not supported or a specified resource is not valid for this operation.

HTTP Status Code: 400

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface V2
- AWS SDK for .NET V4
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- AWS SDK for PHP V3
- AWS SDK for Python (Boto3)
- AWS SDK for Ruby V3

Verify

Verifies a digital signature that was generated by the Sign operation.

Verification confirms that an authorized user signed the message with the specified KMS key and signing algorithm, and the message hasn't changed since it was signed. If the signature is verified, the value of the `SignatureValid` field in the response is `True`. If the signature verification fails, the `Verify` operation fails with an `KMSInvalidSignatureException` exception.

A digital signature is generated by using the private key in an asymmetric KMS key. The signature is verified by using the public key in the same asymmetric KMS key. For information about asymmetric KMS keys, see Asymmetric KMS keys in the *AWS Key Management Service Developer Guide*.

To use the `Verify` operation, specify the same asymmetric KMS key, message, and signing algorithm that were used to produce the signature. The message type does not need to be the same as the one used for signing, but it must indicate whether the value of the `Message` parameter should be hashed as part of the verification process.

You can also verify the digital signature by using the public key of the KMS key outside of AWS KMS. Use the GetPublicKey operation to download the public key in the asymmetric KMS key and then use the public key to verify the signature outside of AWS KMS. The advantage of using the `Verify` operation is that it is performed within AWS KMS. As a result, it's easy to call, the operation is performed within the FIPS boundary, it is logged in AWS CloudTrail, and you can use key policy and IAM policy to determine who is authorized to use the KMS key to verify signatures.

To verify a signature outside of AWS KMS with an SM2 public key (China Regions only), you must specify the distinguishing ID. By default, AWS KMS uses `1234567812345678` as the distinguishing ID. For more information, see Offline verification with SM2 key pairs.

The KMS key that you use for this operation must be in a compatible key state. For details, see Key states of AWS KMS keys in the *AWS Key Management Service Developer Guide*.

Cross-account use: Yes. To perform this operation with a KMS key in a different AWS account, specify the key ARN or alias ARN in the value of the `KeyId` parameter.

Required permissions: kms:Verify (key policy)

Related operations: Sign

Eventual consistency: The AWS KMS API follows an eventual consistency model. For more information, see AWS KMS eventual consistency.

Request Syntax

```
{
   "DryRun": boolean,
   "GrantTokens": [ "string" ],
   "KeyId": "string",
   "Message": blob,
   "MessageType": "string",
   "Signature": blob,
   "SigningAlgorithm": "string"
}
```

Request Parameters

For information about the parameters that are common to all actions, see Common Parameters.

The request accepts the following data in JSON format.

> **Note**
>
> In the following list, the required parameters are described first.

KeyId

Identifies the asymmetric KMS key that will be used to verify the signature. This must be the same KMS key that was used to generate the signature. If you specify a different KMS key, the signature verification fails.

To specify a KMS key, use its key ID, key ARN, alias name, or alias ARN. When using an alias name, prefix it with `"alias/"`. To specify a KMS key in a different AWS account, you must use the key ARN or alias ARN.

For example:

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

- Alias name: `alias/ExampleAlias`
- Alias ARN: `arn:aws:kms:us-east-2:111122223333:alias/ExampleAlias`

To get the key ID and key ARN for a KMS key, use ListKeys or DescribeKey. To get the alias name and alias ARN, use ListAliases.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 2048.

Required: Yes

Message

Specifies the message that was signed. You can submit a raw message of up to 4096 bytes, or a hash digest of the message. If you submit a digest, use the `MessageType` parameter with a value of `DIGEST`.

If the message specified here is different from the message that was signed, the signature verification fails. A message and its hash digest are considered to be the same message.

Type: Base64-encoded binary data object

Length Constraints: Minimum length of 1. Maximum length of 4096.

Required: Yes

Signature

The signature that the `Sign` operation generated.

Type: Base64-encoded binary data object

Length Constraints: Minimum length of 1. Maximum length of 6144.

Required: Yes

SigningAlgorithm

The signing algorithm that was used to sign the message. If you submit a different algorithm, the signature verification fails.

Type: String

Valid Values: RSASSA_PSS_SHA_256 | RSASSA_PSS_SHA_384 | RSASSA_PSS_SHA_512 | RSASSA_PKCS1_V1_5_SHA_256 | RSASSA_PKCS1_V1_5_SHA_384 | RSASSA_PKCS1_V1_5_SHA_512 | ECDSA_SHA_256 | ECDSA_SHA_384 | ECDSA_SHA_512 | SM2DSA | ML_DSA_SHAKE_256 | ED25519_SHA_512 | ED25519_PH_SHA_512

Required: Yes

DryRun

Checks if your request will succeed. DryRun is an optional parameter.

To learn more about how to use this parameter, see Testing your permissions in the *AWS Key Management Service Developer Guide*.

Type: Boolean

Required: No

GrantTokens

A list of grant tokens.

Use a grant token when your permission to call this operation comes from a new grant that has not yet achieved *eventual consistency*. For more information, see Grant token and Using a grant token in the *AWS Key Management Service Developer Guide*.

Type: Array of strings

Array Members: Minimum number of 0 items. Maximum number of 10 items.

Length Constraints: Minimum length of 1. Maximum length of 8192.

Required: No

MessageType

Tells AWS KMS whether the value of the Message parameter should be hashed as part of the signing algorithm. Use RAW for unhashed messages; use DIGEST for message digests, which are already hashed; use EXTERNAL_MU for 64-byte representative μ used in ML-DSA signing as defined in NIST FIPS 204 Section 6.2.

When the value of MessageType is RAW, AWS KMS uses the standard signing algorithm, which begins with a hash function. When the value is DIGEST, AWS KMS skips the hashing step in the

signing algorithm. When the value is EXTERNAL_MU AWS KMS skips the concatenated hashing of the public key hash and the message done in the ML-DSA signing algorithm.

> **Important**
>
> Use the DIGEST or EXTERNAL_MU value only when the value of the Message parameter is a message digest. If you use the DIGEST value with an unhashed message, the security of the signing operation can be compromised.

When using ECC_NIST_EDWARDS25519 KMS keys:

- ED25519_SHA_512 signing algorithm requires AWS KMS MessageType:RAW
- ED25519_PH_SHA_512 signing algorithm requires AWS KMS MessageType:DIGEST

> **Important**
>
> When you specify the ED25519_PH_SHA_512 signing algorithm with MessageType:DIGEST, AWS KMS still performs the SHA-512 prehash described in Step 1 of Section 7.8.1 in FIPS 186-5. This means the input is hashed twice: once by you and once by KMS.

When the value of MessageType is DIGEST, the length of the Message value must match the length of hashed messages for the specified signing algorithm.

When the value of MessageType is EXTERNAL_MU the length of the Message value must be 64 bytes.

You can submit a message digest and omit the MessageType or specify RAW so the digest is hashed again while signing. However, if the signed message is hashed once while signing, but twice while verifying, verification fails, even when the message hasn't changed.

The hashing algorithm that Verify uses is based on the SigningAlgorithm value.

- Signing algorithms that end in SHA_256 use the SHA_256 hashing algorithm.
- Signing algorithms that end in SHA_384 use the SHA_384 hashing algorithm.
- Signing algorithms that end in SHA_512 use the SHA_512 hashing algorithm.
- Signing algorithms that end in SHAKE_256 use the SHAKE_256 hashing algorithm.

- SM2DSA uses the SM3 hashing algorithm. For details, see Offline verification with SM2 key pairs.

Type: String

Valid Values: `RAW | DIGEST | EXTERNAL_MU`

Required: No

Response Syntax

```
{
   "KeyId": "string",
   "SignatureValid": boolean,
   "SigningAlgorithm": "string"
}
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

KeyId

The Amazon Resource Name (key ARN) of the asymmetric KMS key that was used to verify the signature.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 2048.

SignatureValid

A Boolean value that indicates whether the signature was verified. A value of `True` indicates that the `Signature` was produced by signing the `Message` with the specified `KeyID` and `SigningAlgorithm`. If the signature is not verified, the `Verify` operation fails with a `KMSInvalidSignatureException` exception.

Type: Boolean

SigningAlgorithm

The signing algorithm that was used to verify the signature.

Type: String

Valid Values: `RSASSA_PSS_SHA_256 | RSASSA_PSS_SHA_384 | RSASSA_PSS_SHA_512 | RSASSA_PKCS1_V1_5_SHA_256 | RSASSA_PKCS1_V1_5_SHA_384 | RSASSA_PKCS1_V1_5_SHA_512 | ECDSA_SHA_256 | ECDSA_SHA_384 | ECDSA_SHA_512 | SM2DSA | ML_DSA_SHAKE_256 | ED25519_SHA_512 | ED25519_PH_SHA_512`

Errors

For information about the errors that are common to all actions, see Common Error Types.

DependencyTimeoutException

The system timed out while trying to fulfill the request. You can retry the request.

HTTP Status Code: 500

DisabledException

The request was rejected because the specified KMS key is not enabled.

HTTP Status Code: 400

DryRunOperationException

The request was rejected because the DryRun parameter was specified.

HTTP Status Code: 400

InvalidGrantTokenException

The request was rejected because the specified grant token is not valid.

HTTP Status Code: 400

InvalidKeyUsageException

The request was rejected for one of the following reasons:

- The `KeyUsage` value of the KMS key is incompatible with the API operation.

- The encryption algorithm or signing algorithm specified for the operation is incompatible with the type of key material in the KMS key (`KeySpec`).

For encrypting, decrypting, re-encrypting, and generating data keys, the `KeyUsage` must be `ENCRYPT_DECRYPT`. For signing and verifying messages, the `KeyUsage` must be `SIGN_VERIFY`. For generating and verifying message authentication codes (MACs), the `KeyUsage` must be `GENERATE_VERIFY_MAC`. For deriving key agreement secrets, the `KeyUsage` must be `KEY_AGREEMENT`. For encapsulating and decapsulating shared secrets, the `KeyUsage` must be `ENCAPSULATE_DECAPSULATE`. To find the `KeyUsage` of a KMS key, use the DescribeKey operation.

To find the encryption or signing algorithms supported for a particular KMS key, use the DescribeKey operation.

HTTP Status Code: 400

KeyUnavailableException

The request was rejected because the specified KMS key was not available. You can retry the request.

HTTP Status Code: 500

KMSInternalException

The request was rejected because an internal exception occurred. The request can be retried.

HTTP Status Code: 500

KMSInvalidSignatureException

The request was rejected because the signature verification failed. Signature verification fails when it cannot confirm that signature was produced by signing the specified message with the specified KMS key and signing algorithm.

HTTP Status Code: 400

KMSInvalidStateException

The request was rejected because the state of the specified resource is not valid for this request.

This exceptions means one of the following:

- The key state of the KMS key is not compatible with the operation.

To find the key state, use the DescribeKey operation. For more information about which key states are compatible with each AWS KMS operation, see Key states of AWS KMS keys in the AWS Key Management Service Developer Guide .

- For cryptographic operations on KMS keys in custom key stores, this exception represents a general failure with many possible causes. To identify the cause, see the error message that accompanies the exception.

HTTP Status Code: 400

NotFoundException

The request was rejected because the specified entity or resource could not be found.

HTTP Status Code: 400

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface V2
- AWS SDK for .NET V4
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- AWS SDK for PHP V3
- AWS SDK for Python (Boto3)
- AWS SDK for Ruby V3

VerifyMac

Verifies the hash-based message authentication code (HMAC) for a specified message, HMAC KMS key, and MAC algorithm. To verify the HMAC, `VerifyMac` computes an HMAC using the message, HMAC KMS key, and MAC algorithm that you specify, and compares the computed HMAC to the HMAC that you specify. If the HMACs are identical, the verification succeeds; otherwise, it fails. Verification indicates that the message hasn't changed since the HMAC was calculated, and the specified key was used to generate and verify the HMAC.

HMAC KMS keys and the HMAC algorithms that AWS KMS uses conform to industry standards defined in RFC 2104.

This operation is part of AWS KMS support for HMAC KMS keys. For details, see HMAC keys in AWS KMS in the *AWS Key Management Service Developer Guide*.

The KMS key that you use for this operation must be in a compatible key state. For details, see Key states of AWS KMS keys in the *AWS Key Management Service Developer Guide*.

Cross-account use: Yes. To perform this operation with a KMS key in a different AWS account, specify the key ARN or alias ARN in the value of the `KeyId` parameter.

Required permissions: kms:VerifyMac (key policy)

Related operations: GenerateMac

Eventual consistency: The AWS KMS API follows an eventual consistency model. For more information, see AWS KMS eventual consistency.

Request Syntax

```
{
   "DryRun": boolean,
   "GrantTokens": [ "string" ],
   "KeyId": "string",
   "Mac": blob,
   "MacAlgorithm": "string",
   "Message": blob
}
```

Request Parameters

For information about the parameters that are common to all actions, see Common Parameters.

The request accepts the following data in JSON format.

> **Note**
>
> In the following list, the required parameters are described first.

KeyId

The KMS key that will be used in the verification.

Enter a key ID of the KMS key that was used to generate the HMAC. If you identify a different KMS key, the `VerifyMac` operation fails.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 2048.

Required: Yes

Mac

The HMAC to verify. Enter the HMAC that was generated by the GenerateMac operation when you specified the same message, HMAC KMS key, and MAC algorithm as the values specified in this request.

Type: Base64-encoded binary data object

Length Constraints: Minimum length of 1. Maximum length of 6144.

Required: Yes

MacAlgorithm

The MAC algorithm that will be used in the verification. Enter the same MAC algorithm that was used to compute the HMAC. This algorithm must be supported by the HMAC KMS key identified by the `KeyId` parameter.

Type: String

Valid Values: `HMAC_SHA_224` | `HMAC_SHA_256` | `HMAC_SHA_384` | `HMAC_SHA_512`

Required: Yes

Message

The message that will be used in the verification. Enter the same message that was used to generate the HMAC.

GenerateMac and `VerifyMac` do not provide special handling for message digests. If you generated an HMAC for a hash digest of a message, you must verify the HMAC for the same hash digest.

Type: Base64-encoded binary data object

Length Constraints: Minimum length of 1. Maximum length of 4096.

Required: Yes

DryRun

Checks if your request will succeed. `DryRun` is an optional parameter.

To learn more about how to use this parameter, see Testing your permissions in the *AWS Key Management Service Developer Guide*.

Type: Boolean

Required: No

GrantTokens

A list of grant tokens.

Use a grant token when your permission to call this operation comes from a new grant that has not yet achieved *eventual consistency*. For more information, see Grant token and Using a grant token in the *AWS Key Management Service Developer Guide*.

Type: Array of strings

Array Members: Minimum number of 0 items. Maximum number of 10 items.

Length Constraints: Minimum length of 1. Maximum length of 8192.

Required: No

Response Syntax

```
{
   "KeyId": "string",
   "MacAlgorithm": "string",
   "MacValid": boolean
}
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

KeyId

The HMAC KMS key used in the verification.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 2048.

MacAlgorithm

The MAC algorithm used in the verification.

Type: String

Valid Values: `HMAC_SHA_224 | HMAC_SHA_256 | HMAC_SHA_384 | HMAC_SHA_512`

MacValid

A Boolean value that indicates whether the HMAC was verified. A value of `True` indicates that the HMAC (`Mac`) was generated with the specified `Message`, HMAC KMS key (`KeyID`) and `MacAlgorithm`..

If the HMAC is not verified, the `VerifyMac` operation fails with a `KMSInvalidMacException` exception. This exception indicates that one or more of the inputs changed since the HMAC was computed.

Type: Boolean

Errors

For information about the errors that are common to all actions, see Common Error Types.

DisabledException

The request was rejected because the specified KMS key is not enabled.

HTTP Status Code: 400

DryRunOperationException

The request was rejected because the DryRun parameter was specified.

HTTP Status Code: 400

InvalidGrantTokenException

The request was rejected because the specified grant token is not valid.

HTTP Status Code: 400

InvalidKeyUsageException

The request was rejected for one of the following reasons:

- The `KeyUsage` value of the KMS key is incompatible with the API operation.
- The encryption algorithm or signing algorithm specified for the operation is incompatible with the type of key material in the KMS key (`KeySpec`).

For encrypting, decrypting, re-encrypting, and generating data keys, the `KeyUsage` must be `ENCRYPT_DECRYPT`. For signing and verifying messages, the `KeyUsage` must be `SIGN_VERIFY`. For generating and verifying message authentication codes (MACs), the `KeyUsage` must be `GENERATE_VERIFY_MAC`. For deriving key agreement secrets, the `KeyUsage` must be `KEY_AGREEMENT`. For encapsulating and decapsulating shared secrets, the `KeyUsage` must be `ENCAPSULATE_DECAPSULATE`. To find the `KeyUsage` of a KMS key, use the DescribeKey operation.

To find the encryption or signing algorithms supported for a particular KMS key, use the DescribeKey operation.

HTTP Status Code: 400

KeyUnavailableException

The request was rejected because the specified KMS key was not available. You can retry the request.

HTTP Status Code: 500

KMSInternalException

The request was rejected because an internal exception occurred. The request can be retried.

HTTP Status Code: 500

KMSInvalidMacException

The request was rejected because the HMAC verification failed. HMAC verification fails when the HMAC computed by using the specified message, HMAC KMS key, and MAC algorithm does not match the HMAC specified in the request.

HTTP Status Code: 400

KMSInvalidStateException

The request was rejected because the state of the specified resource is not valid for this request.

This exceptions means one of the following:

- The key state of the KMS key is not compatible with the operation.

  To find the key state, use the DescribeKey operation. For more information about which key states are compatible with each AWS KMS operation, see Key states of AWS KMS keys in the AWS Key Management Service Developer Guide .
- For cryptographic operations on KMS keys in custom key stores, this exception represents a general failure with many possible causes. To identify the cause, see the error message that accompanies the exception.

HTTP Status Code: 400

NotFoundException

The request was rejected because the specified entity or resource could not be found.

HTTP Status Code: 400

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface V2
- AWS SDK for .NET V4
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- AWS SDK for PHP V3
- AWS SDK for Python (Boto3)
- AWS SDK for Ruby V3

Data Types

The AWS Key Management Service API contains several data types that various actions use. This section describes each data type in detail.

> **Note**
>
> The order of each element in a data type structure is not guaranteed. Applications should not assume a particular order.

The following data types are supported:

- AliasListEntry
- CustomKeyStoresListEntry
- GrantConstraints
- GrantListEntry
- KeyLastUsageData
- KeyListEntry
- KeyMetadata
- MultiRegionConfiguration
- MultiRegionKey
- RecipientInfo
- RotationsListEntry
- Tag
- XksKeyConfigurationType
- XksProxyAuthenticationCredentialType
- XksProxyConfigurationType

AliasListEntry

Contains information about an alias.

Contents

> **Note**
>
> In the following list, the required parameters are described first.

AliasArn

String that contains the key ARN.

Type: String

Length Constraints: Minimum length of 20. Maximum length of 2048.

Required: No

AliasName

String that contains the alias. This value begins with `alias/`.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 256.

Pattern: `^[a-zA-Z0-9:/_-]+$`

Required: No

CreationDate

Date and time that the alias was most recently created in the account and Region. Formatted as Unix time.

Type: Timestamp

Required: No

LastUpdatedDate

Date and time that the alias was most recently associated with a KMS key in the account and Region. Formatted as Unix time.

Type: Timestamp

Required: No

TargetKeyId

String that contains the key identifier of the KMS key associated with the alias.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 2048.

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

CustomKeyStoresListEntry

Contains information about each custom key store in the custom key store list.

Contents

> **Note**
>
> In the following list, the required parameters are described first.

CloudHsmClusterId

A unique identifier for the AWS CloudHSM cluster that is associated with an AWS CloudHSM key store. This field appears only when the `CustomKeyStoreType` is `AWS_CLOUDHSM`.

Type: String

Length Constraints: Minimum length of 19. Maximum length of 24.

Pattern: `cluster-[2-7a-zA-Z]{11,16}`

Required: No

ConnectionErrorCode

Describes the connection error. This field appears in the response only when the `ConnectionState` is `FAILED`.

Many failures can be resolved by updating the properties of the custom key store. To update a custom key store, disconnect it (DisconnectCustomKeyStore), correct the errors (UpdateCustomKeyStore), and try to connect again (ConnectCustomKeyStore). For additional help resolving these errors, see How to Fix a Connection Failure in *AWS Key Management Service Developer Guide*.

All custom key stores:

- `INTERNAL_ERROR` — AWS KMS could not complete the request due to an internal error. Retry the request. For `ConnectCustomKeyStore` requests, disconnect the custom key store before trying to connect again.
- `NETWORK_ERRORS` — Network errors are preventing AWS KMS from connecting the custom key store to its backing key store.

AWS CloudHSM key stores:

- `CLUSTER_NOT_FOUND` — AWS KMS cannot find the AWS CloudHSM cluster with the specified cluster ID.
- `INSUFFICIENT_CLOUDHSM_HSMS` — The associated AWS CloudHSM cluster does not contain any active HSMs. To connect a custom key store to its AWS CloudHSM cluster, the cluster must contain at least one active HSM.
- `INSUFFICIENT_FREE_ADDRESSES_IN_SUBNET` — At least one private subnet associated with the AWS CloudHSM cluster doesn't have any available IP addresses. A AWS CloudHSM key store connection requires one free IP address in each of the associated private subnets, although two are preferable. For details, see How to Fix a Connection Failure in the *AWS Key Management Service Developer Guide*.
- `INVALID_CREDENTIALS` — The `KeyStorePassword` for the custom key store doesn't match the current password of the `kmsuser` crypto user in the AWS CloudHSM cluster. Before you can connect your custom key store to its AWS CloudHSM cluster, you must change the `kmsuser` account password and update the `KeyStorePassword` value for the custom key store.
- `SUBNET_NOT_FOUND` — A subnet in the AWS CloudHSM cluster configuration was deleted. If AWS KMS cannot find all of the subnets in the cluster configuration, attempts to connect the custom key store to the AWS CloudHSM cluster fail. To fix this error, create a cluster from a recent backup and associate it with your custom key store. (This process creates a new cluster configuration with a VPC and private subnets.) For details, see How to Fix a Connection Failure in the *AWS Key Management Service Developer Guide*.
- `USER_LOCKED_OUT` — The `kmsuser` CU account is locked out of the associated AWS CloudHSM cluster due to too many failed password attempts. Before you can connect your custom key store to its AWS CloudHSM cluster, you must change the `kmsuser` account password and update the key store password value for the custom key store.
- `USER_LOGGED_IN` — The `kmsuser` CU account is logged into the associated AWS CloudHSM cluster. This prevents AWS KMS from rotating the `kmsuser` account password and logging into the cluster. Before you can connect your custom key store to its AWS CloudHSM cluster, you must log the `kmsuser` CU out of the cluster. If you changed the `kmsuser` password to log into the cluster, you must also and update the key store password value for the custom key store. For help, see How to Log Out and Reconnect in the *AWS Key Management Service Developer Guide*.
- `USER_NOT_FOUND` — AWS KMS cannot find a `kmsuser` CU account in the associated AWS CloudHSM cluster. Before you can connect your custom key store to its AWS CloudHSM

cluster, you must create a `kmsuser` CU account in the cluster, and then update the key store password value for the custom key store.

External key stores:

- `INVALID_CREDENTIALS` — One or both of the `XksProxyAuthenticationCredential` values is not valid on the specified external key store proxy.
- `XKS_PROXY_ACCESS_DENIED` — AWS KMS requests are denied access to the external key store proxy. If the external key store proxy has authorization rules, verify that they permit AWS KMS to communicate with the proxy on your behalf.
- `XKS_PROXY_INVALID_CONFIGURATION` — A configuration error is preventing the external key store from connecting to its proxy. Verify the value of the `XksProxyUriPath`.
- `XKS_PROXY_INVALID_RESPONSE` — AWS KMS cannot interpret the response from the external key store proxy. If you see this connection error code repeatedly, notify your external key store proxy vendor.
- `XKS_PROXY_INVALID_TLS_CONFIGURATION` — AWS KMS cannot connect to the external key store proxy because the TLS configuration is invalid. Verify that the XKS proxy supports TLS 1.2 or 1.3. Also, verify that the TLS certificate is not expired, and that it matches the hostname in the `XksProxyUriEndpoint` value, and that it is signed by a certificate authority included in the Trusted Certificate Authorities list.
- `XKS_PROXY_NOT_REACHABLE` — AWS KMS can't communicate with your external key store proxy. Verify that the `XksProxyUriEndpoint` and `XksProxyUriPath` are correct. Use the tools for your external key store proxy to verify that the proxy is active and available on its network. Also, verify that your external key manager instances are operating properly. Connection attempts fail with this connection error code if the proxy reports that all external key manager instances are unavailable.
- `XKS_PROXY_TIMED_OUT` — AWS KMS can connect to the external key store proxy, but the proxy does not respond to AWS KMS in the time allotted. If you see this connection error code repeatedly, notify your external key store proxy vendor.
- `XKS_VPC_ENDPOINT_SERVICE_INVALID_CONFIGURATION` — The Amazon VPC endpoint service configuration doesn't conform to the requirements for an AWS KMS external key store.
  - The VPC endpoint service must be an endpoint service for interface endpoints in the caller's AWS account.
  - It must have a network load balancer (NLB) connected to at least two subnets, each in a different Availability Zone.

- The `Allow principals` list must include the AWS KMS service principal for the Region, `cks.kms.<region>.amazonaws.com`, such as `cks.kms.us-east-1.amazonaws.com`.
- It must *not* require acceptance of connection requests.
- It must have a private DNS name. The private DNS name for an external key store with `VPC_ENDPOINT_SERVICE` connectivity must be unique in its AWS Region.
- The domain of the private DNS name must have a verification status of `verified`.
- The TLS certificate specifies the private DNS hostname at which the endpoint is reachable.

- `XKS_VPC_ENDPOINT_SERVICE_NOT_FOUND` — AWS KMS can't find the VPC endpoint service that it uses to communicate with the external key store proxy. Verify that the `XksProxyVpcEndpointServiceName` is correct and the AWS KMS service principal has service consumer permissions on the Amazon VPC endpoint service.

Type: String

Valid Values: `INVALID_CREDENTIALS | CLUSTER_NOT_FOUND | NETWORK_ERRORS | INTERNAL_ERROR | INSUFFICIENT_CLOUDHSM_HSMS | USER_LOCKED_OUT | USER_NOT_FOUND | USER_LOGGED_IN | SUBNET_NOT_FOUND | INSUFFICIENT_FREE_ADDRESSES_IN_SUBNET | XKS_PROXY_ACCESS_DENIED | XKS_PROXY_NOT_REACHABLE | XKS_VPC_ENDPOINT_SERVICE_NOT_FOUND | XKS_PROXY_INVALID_RESPONSE | XKS_PROXY_INVALID_CONFIGURATION | XKS_VPC_ENDPOINT_SERVICE_INVALID_CONFIGURATION | XKS_PROXY_TIMED_OUT | XKS_PROXY_INVALID_TLS_CONFIGURATION`

Required: No

ConnectionState

Indicates whether the custom key store is connected to its backing key store. For an AWS CloudHSM key store, the `ConnectionState` indicates whether it is connected to its AWS CloudHSM cluster. For an external key store, the `ConnectionState` indicates whether it is connected to the external key store proxy that communicates with your external key manager.

You can create and use KMS keys in your custom key stores only when its `ConnectionState` is `CONNECTED`.

The `ConnectionState` value is `DISCONNECTED` only if the key store has never been connected or you use the DisconnectCustomKeyStore operation to disconnect it. If the value is `CONNECTED` but you are having trouble using the custom key store, make sure that the backing key store is reachable and active. For an AWS CloudHSM key store, verify that its associated AWS CloudHSM

cluster is active and contains at least one active HSM. For an external key store, verify that the external key store proxy and external key manager are connected and enabled.

A value of `FAILED` indicates that an attempt to connect was unsuccessful. The `ConnectionErrorCode` field in the response indicates the cause of the failure. For help resolving a connection failure, see Troubleshooting a custom key store in the *AWS Key Management Service Developer Guide*.

Type: String

Valid Values: `CONNECTED | CONNECTING | FAILED | DISCONNECTED | DISCONNECTING`

Required: No

CreationDate

The date and time when the custom key store was created.

Type: Timestamp

Required: No

CustomKeyStoreId

A unique identifier for the custom key store.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 64.

Required: No

CustomKeyStoreName

The user-specified friendly name for the custom key store.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 256.

Required: No

CustomKeyStoreType

Indicates the type of the custom key store. `AWS_CLOUDHSM` indicates a custom key store backed by an AWS CloudHSM cluster. `EXTERNAL_KEY_STORE` indicates a custom key store backed by an external key store proxy and external key manager outside of AWS.

Type: String

Valid Values: `AWS_CLOUDHSM | EXTERNAL_KEY_STORE`

Required: No

TrustAnchorCertificate

The trust anchor certificate of the AWS CloudHSM cluster associated with an AWS CloudHSM key store. When you initialize the cluster, you create this certificate and save it in the `customerCA.crt` file.

This field appears only when the `CustomKeyStoreType` is `AWS_CLOUDHSM`.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 5000.

Required: No

XksProxyConfiguration

Configuration settings for the external key store proxy (XKS proxy). The external key store proxy translates AWS KMS requests into a format that your external key manager can understand. The proxy configuration includes connection information that AWS KMS requires.

This field appears only when the `CustomKeyStoreType` is `EXTERNAL_KEY_STORE`.

Type: XksProxyConfigurationType object

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

GrantConstraints

Use this structure to allow cryptographic operations in the grant only when the operation request meets the specified constraints.

AWS KMS supports the following grant constraints:

- `EncryptionContextEquals` and `EncryptionContextSubset` — These encryption context constraints apply only to cryptographic operations that support an encryption context, that is, all cryptographic operations with a symmetric KMS key. Encryption context grant constraints are not applied to operations that do not support an encryption context, such as cryptographic operations with asymmetric KMS keys and management operations, such as DescribeKey or RetireGrant.

  > **Important**
  >
  > In a cryptographic operation, the encryption context in the decryption operation must be an exact, case-sensitive match for the keys and values in the encryption context of the encryption operation. Only the order of the pairs can vary.
  > However, in a grant constraint, the key in each key-value pair is not case sensitive, but the value is case sensitive.
  > To avoid confusion, do not use multiple encryption context pairs that differ only by case.
  > To require a fully case-sensitive encryption context, use the `kms:EncryptionContext:` and `kms:EncryptionContextKeys` conditions in an IAM or key policy. For details, see kms:EncryptionContext:context-key in the AWS Key Management Service Developer Guide .

- `SourceArn` — This grant constraint allows the permissions in the grant only when the request is made on behalf of a specific AWS resource, identified by its Amazon Resource Name (ARN). This is effectively the same as having the aws:SourceArn global condition key in the grant. The SourceArn constraint is supported on grants for all types of KMS keys and can also be applied to the DescribeKey operation when specified in the request. However, it does not apply to RetireGrant operation.

Contents

> **Note**
>
> In the following list, the required parameters are described first.

EncryptionContextEquals

A list of key-value pairs that must match the encryption context in the cryptographic operation request. The grant allows the operation only when the encryption context in the request is the same as the encryption context specified in this constraint.

Type: String to string map

Required: No

EncryptionContextSubset

A list of key-value pairs that must be included in the encryption context of the cryptographic operation request. The grant allows the cryptographic operation only when the encryption context in the request includes the key-value pairs specified in this constraint, although it can include additional key-value pairs.

Type: String to string map

Required: No

SourceArn

The Amazon Resource Name (ARN) of an AWS resource on behalf of which the request is made. This is effectively the same as having the aws:SourceArn global condition key in the grant. The SourceArn constraint ensures that the principal can use the KMS key only when the request is made on behalf of the specified resource.

Type: String

Length Constraints: Minimum length of 20. Maximum length of 512.

Pattern: `^arn:aws[a-z0-9-]*:[a-z0-9-]+:[a-z0-9-]*:[0-9]{12}:.+$`

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

GrantListEntry

Contains information about a grant.

Contents

> **Note**
>
> In the following list, the required parameters are described first.

Constraints

The constraints on the grant, such as encryption context pairs or a SourceArn, that restrict the subsequent operations the grant allows.

Type: GrantConstraints object

Required: No

CreationDate

The date and time when the grant was created.

Type: Timestamp

Required: No

GranteePrincipal

The identity that gets the permissions in the grant.

When a grant is created with the `GranteePrincipal` field, the `ListGrants` response usually contains the user or role designated as the grantee principal in the grant. However, if the grantee principal is an AWS service, the `GranteePrincipal` field contains an AWS service principal, which might correspond to several different grantee principals, such as an IAM user, IAM role, or AWS account.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 256.

Pattern: `^[\w+=,.@:/-]+$`

Required: No

GranteeServicePrincipal

The AWS service principal that gets the permissions in the grant.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 128.

Pattern: `^([A-Za-z0-9\-]+)\.([A-Za-z0-9\-]+)(\.[A-Za-z0-9\-]+)+$`

Required: No

GrantId

The unique identifier for the grant.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 128.

Required: No

IssuingAccount

The AWS account under which the grant was issued.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 256.

Pattern: `^[\w+=,.@:/-]+$`

Required: No

KeyId

The unique identifier for the KMS key to which the grant applies.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 2048.

Required: No

Name

The friendly name that identifies the grant. If a name was provided in the CreateGrant request, that name is returned. Otherwise this value is null.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 256.

Pattern: `^[a-zA-Z0-9:/_-]+$`

Required: No

Operations

The list of operations permitted by the grant.

Type: Array of strings

Valid Values: `Decrypt | Encrypt | GenerateDataKey | GenerateDataKeyWithoutPlaintext | ReEncryptFrom | ReEncryptTo | Sign | Verify | GetPublicKey | CreateGrant | RetireGrant | DescribeKey | GenerateDataKeyPair | GenerateDataKeyPairWithoutPlaintext | GenerateMac | VerifyMac | DeriveSharedSecret | Encapsulate | Decapsulate`

Required: No

RetiringPrincipal

The principal that can retire the grant.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 256.

Pattern: `^[\w+=,.@:/-]+$`

Required: No

RetiringServicePrincipal

The AWS service principal that can retire the grant.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 128.

Pattern: `^([A-Za-z0-9\-]+)\.([A-Za-z0-9\-]+)(\.[A-Za-z0-9\-]+)+$`

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

KeyLastUsageData

Contains usage information about the last time the KMS key was used for a successful cryptographic operation.

Contents

> **Note**
>
> In the following list, the required parameters are described first.

CloudTrailEventId

The CloudTrail `eventId` associated with the last successful cryptographic operation. Absent if the key has not been used since AWS KMS began tracking.

Type: String

Required: No

KmsRequestId

The AWS KMS request ID associated with the last successful cryptographic operation. Absent if the key has not been used since AWS KMS began tracking.

Type: String

Required: No

Operation

The last successful cryptographic operation the KMS key was used for. Absent if the key has not been used since AWS KMS began tracking.

Type: String

Valid Values: `Decrypt | DeriveSharedSecret | Encrypt | GenerateDataKey | GenerateDataKeyPair | GenerateDataKeyPairWithoutPlaintext | GenerateDataKeyWithoutPlaintext | GenerateMac | ReEncrypt | Sign | Verify | VerifyMac | Encapsulate | Decapsulate`

Required: No

Timestamp

The date and time when the KMS key was most recently used for a successful cryptographic operation. Absent if the key has not been used since AWS KMS began tracking.

Type: Timestamp

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

KeyListEntry

Contains information about each entry in the key list.

Contents

> **Note**
>
> In the following list, the required parameters are described first.

KeyArn

ARN of the key.

Type: String

Length Constraints: Minimum length of 20. Maximum length of 2048.

Required: No

KeyId

Unique identifier of the key.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 2048.

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

KeyMetadata

Contains metadata about a KMS key.

This data type is used as a response element for the CreateKey, DescribeKey, and ReplicateKey operations.

Contents

> **Note**
>
> In the following list, the required parameters are described first.

KeyId

The globally unique identifier for the KMS key.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 2048.

Required: Yes

Arn

The Amazon Resource Name (ARN) of the KMS key. For examples, see AWS Key Management Service (AWS KMS) in the Example ARNs section of the *AWS General Reference*.

Type: String

Length Constraints: Minimum length of 20. Maximum length of 2048.

Required: No

AWSAccountId

The twelve-digit account ID of the AWS account that owns the KMS key.

Type: String

Required: No

CloudHsmClusterId

The cluster ID of the AWS CloudHSM cluster that contains the key material for the KMS key. When you create a KMS key in an AWS CloudHSM custom key store, AWS KMS creates the key material for the KMS key in the associated AWS CloudHSM cluster. This field is present only when the KMS key is created in an AWS CloudHSM key store.

Type: String

Length Constraints: Minimum length of 19. Maximum length of 24.

Pattern: `cluster-[2-7a-zA-Z]{11,16}`

Required: No

CreationDate

The date and time when the KMS key was created.

Type: Timestamp

Required: No

CurrentKeyMaterialId

Identifies the current key material. This value is present for symmetric encryption keys with `AWS_KMS` or `EXTERNAL` origin. These KMS keys support automatic or on-demand key rotation and can have multiple key materials associated with them. AWS KMS uses the current key material for both encryption and decryption, and the non-current key material for decryption operations only.

Type: String

Length Constraints: Fixed length of 64.

Pattern: `^[a-f0-9]+$`

Required: No

CustomerMasterKeySpec

This member has been deprecated.

Instead, use the `KeySpec` field.

The KeySpec and CustomerMasterKeySpec fields have the same value. We recommend that you use the KeySpec field in your code. However, to avoid breaking changes, AWS KMS supports both fields.

Type: String

Valid Values: RSA_2048 | RSA_3072 | RSA_4096 | ECC_NIST_P256 | ECC_NIST_P384 | ECC_NIST_P521 | ECC_SECG_P256K1 | SYMMETRIC_DEFAULT | HMAC_224 | HMAC_256 | HMAC_384 | HMAC_512 | SM2

Required: No

CustomKeyStoreId

A unique identifier for the custom key store that contains the KMS key. This field is present only when the KMS key is created in a custom key store.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 64.

Required: No

DeletionDate

The date and time after which AWS KMS deletes this KMS key. This value is present only when the KMS key is scheduled for deletion, that is, when its KeyState is PendingDeletion.

When the primary key in a multi-Region key is scheduled for deletion but still has replica keys, its key state is PendingReplicaDeletion and the length of its waiting period is displayed in the PendingDeletionWindowInDays field.

Type: Timestamp

Required: No

Description

The description of the KMS key.

Type: String

Length Constraints: Minimum length of 0. Maximum length of 8192.

Required: No

Enabled

Specifies whether the KMS key is enabled. When `KeyState` is `Enabled` this value is true, otherwise it is false.

Type: Boolean

Required: No

EncapsulationAlgorithms

The key encapsulation algorithm that the KMS key supports. This field appears only for KMS keys with a `KeyUsage` value of `ENCAPSULATE_DECAPSULATE`.

Type: Array of strings

Valid Values: `ML_KEM`

Required: No

EncryptionAlgorithms

The encryption algorithms that the KMS key supports. You cannot use the KMS key with other encryption algorithms within AWS KMS.

This value is present only when the `KeyUsage` of the KMS key is `ENCRYPT_DECRYPT`.

Type: Array of strings

Valid Values: `SYMMETRIC_DEFAULT | RSAES_OAEP_SHA_1 | RSAES_OAEP_SHA_256 | SM2PKE`

Required: No

ExpirationModel

Specifies whether the KMS key's key material expires. This value is present only when `Origin` is `EXTERNAL`, otherwise this value is omitted.

Type: String

Valid Values: `KEY_MATERIAL_EXPIRES | KEY_MATERIAL_DOES_NOT_EXPIRE`

Required: No

KeyAgreementAlgorithms

The key agreement algorithm used to derive a shared secret.

Type: Array of strings

Valid Values: ECDH

Required: No

KeyManager

The manager of the KMS key. KMS keys in your AWS account are either customer managed or AWS managed. For more information about the difference, see KMS keys in the *AWS Key Management Service Developer Guide*.

Type: String

Valid Values: AWS | CUSTOMER

Required: No

KeySpec

Describes the type of key material in the KMS key.

Type: String

Valid Values: RSA_2048 | RSA_3072 | RSA_4096 | ECC_NIST_P256 | ECC_NIST_P384 | ECC_NIST_P521 | ECC_SECG_P256K1 | SYMMETRIC_DEFAULT | HMAC_224 | HMAC_256 | HMAC_384 | HMAC_512 | SM2 | ML_DSA_44 | ML_DSA_65 | ML_DSA_87 | ECC_NIST_EDWARDS25519 | ML_KEM_512 | ML_KEM_768 | ML_KEM_1024

Required: No

KeyState

The current status of the KMS key.

For more information about how key state affects the use of a KMS key, see Key states of AWS KMS keys in the *AWS Key Management Service Developer Guide*.

Type: String

Valid Values: `Creating | Enabled | Disabled | PendingDeletion | PendingImport | PendingReplicaDeletion | Unavailable | Updating`

Required: No

KeyUsage

The cryptographic operations for which you can use the KMS key.

Type: String

Valid Values: `SIGN_VERIFY | ENCRYPT_DECRYPT | GENERATE_VERIFY_MAC | KEY_AGREEMENT | ENCAPSULATE_DECAPSULATE`

Required: No

MacAlgorithms

The message authentication code (MAC) algorithm that the HMAC KMS key supports.

This value is present only when the `KeyUsage` of the KMS key is `GENERATE_VERIFY_MAC`.

Type: Array of strings

Valid Values: `HMAC_SHA_224 | HMAC_SHA_256 | HMAC_SHA_384 | HMAC_SHA_512`

Required: No

MultiRegion

Indicates whether the KMS key is a multi-Region (`True`) or regional (`False`) key. This value is `True` for multi-Region primary and replica keys and `False` for regional KMS keys.

For more information about multi-Region keys, see Multi-Region keys in AWS KMS in the *AWS Key Management Service Developer Guide*.

Type: Boolean

Required: No

MultiRegionConfiguration

Lists the primary and replica keys in same multi-Region key. This field is present only when the value of the `MultiRegion` field is `True`.

For more information about any listed KMS key, use the DescribeKey operation.

- `MultiRegionKeyType` indicates whether the KMS key is a `PRIMARY` or `REPLICA` key.
- `PrimaryKey` displays the key ARN and Region of the primary key. This field displays the current KMS key if it is the primary key.
- `ReplicaKeys` displays the key ARNs and Regions of all replica keys. This field includes the current KMS key if it is a replica key.

Type: MultiRegionConfiguration object

Required: No

Origin

The source of the key material for the KMS key. When this value is `AWS_KMS`, AWS KMS created the key material. When this value is `EXTERNAL`, the key material was imported or the KMS key doesn't have any key material. When this value is `AWS_CLOUDHSM`, the key material was created in the AWS CloudHSM cluster associated with a custom key store.

Type: String

Valid Values: `AWS_KMS | EXTERNAL | AWS_CLOUDHSM | EXTERNAL_KEY_STORE`

Required: No

PendingDeletionWindowInDays

The waiting period before the primary key in a multi-Region key is deleted. This waiting period begins when the last of its replica keys is deleted. This value is present only when the `KeyState` of the KMS key is `PendingReplicaDeletion`. That indicates that the KMS key is the primary key in a multi-Region key, it is scheduled for deletion, and it still has existing replica keys.

When a single-Region KMS key or a multi-Region replica key is scheduled for deletion, its deletion date is displayed in the `DeletionDate` field. However, when the primary key in a multi-Region key is scheduled for deletion, its waiting period doesn't begin until all of its replica keys are deleted. This value displays that waiting period. When the last replica key in the multi-Region key is deleted, the `KeyState` of the scheduled primary key changes from `PendingReplicaDeletion` to `PendingDeletion` and the deletion date appears in the `DeletionDate` field.

Type: Integer

Valid Range: Minimum value of 1. Maximum value of 365.

Required: No

SigningAlgorithms

The signing algorithms that the KMS key supports. You cannot use the KMS key with other signing algorithms within AWS KMS.

This field appears only when the `KeyUsage` of the KMS key is `SIGN_VERIFY`.

Type: Array of strings

Valid Values: `RSASSA_PSS_SHA_256 | RSASSA_PSS_SHA_384 | RSASSA_PSS_SHA_512 | RSASSA_PKCS1_V1_5_SHA_256 | RSASSA_PKCS1_V1_5_SHA_384 | RSASSA_PKCS1_V1_5_SHA_512 | ECDSA_SHA_256 | ECDSA_SHA_384 | ECDSA_SHA_512 | SM2DSA | ML_DSA_SHAKE_256 | ED25519_SHA_512 | ED25519_PH_SHA_512`

Required: No

ValidTo

The earliest time at which any imported key material permanently associated with this KMS key expires. When a key material expires, AWS KMS deletes the key material and the KMS key becomes unusable. This value is present only for KMS keys whose `Origin` is `EXTERNAL` and the `ExpirationModel` is `KEY_MATERIAL_EXPIRES`, otherwise this value is omitted.

Type: Timestamp

Required: No

XksKeyConfiguration

Information about the external key that is associated with a KMS key in an external key store.

For more information, see External key in the *AWS Key Management Service Developer Guide*.

Type: XksKeyConfigurationType object

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

MultiRegionConfiguration

Describes the configuration of this multi-Region key. This field appears only when the KMS key is a primary or replica of a multi-Region key.

For more information about any listed KMS key, use the DescribeKey operation.

Contents

> **Note**
>
> In the following list, the required parameters are described first.

MultiRegionKeyType

Indicates whether the KMS key is a `PRIMARY` or `REPLICA` key.

Type: String

Valid Values: `PRIMARY | REPLICA`

Required: No

PrimaryKey

Displays the key ARN and Region of the primary key. This field includes the current KMS key if it is the primary key.

Type: MultiRegionKey object

Required: No

ReplicaKeys

displays the key ARNs and Regions of all replica keys. This field includes the current KMS key if it is a replica key.

Type: Array of MultiRegionKey objects

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

MultiRegionKey

Describes the primary or replica key in a multi-Region key.

Contents

> **Note**
>
> In the following list, the required parameters are described first.

Arn

Displays the key ARN of a primary or replica key of a multi-Region key.

Type: String

Length Constraints: Minimum length of 20. Maximum length of 2048.

Required: No

Region

Displays the AWS Region of a primary or replica key in a multi-Region key.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 32.

Pattern: `^([a-z]+-){2,3}\d+$`

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

RecipientInfo

Contains information about the party that receives the response from the API operation.

This data type is designed to support AWS Nitro Enclaves and AWS NitroTPM, which lets you create an attested environment in Amazon EC2. For information about the interaction between AWS KMS and AWS Nitro Enclaves or AWS NitroTPM, see Cryptographic attestation support in AWS KMS in the *AWS Key Management Service Developer Guide*.

Contents

> **Note**
>
> In the following list, the required parameters are described first.

AttestationDocument

The attestation document for an AWS Nitro Enclave or a NitroTPM. This document includes the enclave's public key.

Type: Base64-encoded binary data object

Length Constraints: Minimum length of 1. Maximum length of 262144.

Required: No

KeyEncryptionAlgorithm

The encryption algorithm that AWS KMS should use with the public key for an AWS Nitro Enclave or NitroTPM to encrypt plaintext values for the response. The only valid value is `RSAES_OAEP_SHA_256`.

Type: String

Valid Values: `RSAES_OAEP_SHA_256`

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

RotationsListEntry

Each entry contains information about one of the key materials associated with a KMS key.

Contents

> **Note**
>
> In the following list, the required parameters are described first.

ExpirationModel

Indicates if the key material is configured to automatically expire. There are two possible values for this field: `KEY_MATERIAL_EXPIRES` and `KEY_MATERIAL_DOES_NOT_EXPIRE`. For any key material that expires, the expiration date and time is indicated in `ValidTo`. This field is only present for symmetric encryption KMS keys with `EXTERNAL` origin.

Type: String

Valid Values: `KEY_MATERIAL_EXPIRES | KEY_MATERIAL_DOES_NOT_EXPIRE`

Required: No

ImportState

Indicates if the key material is currently imported into AWS KMS. It has two possible values: `IMPORTED` or `PENDING_IMPORT`. This field is only present for symmetric encryption KMS keys with `EXTERNAL` origin.

Type: String

Valid Values: `IMPORTED | PENDING_IMPORT`

Required: No

KeyId

Unique identifier of the key.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 2048.

Required: No

KeyMaterialDescription

User-specified description of the key material. This field is only present for symmetric encryption KMS keys with EXTERNAL origin.

Type: String

Length Constraints: Minimum length of 0. Maximum length of 256.

Pattern: ^[a-zA-Z0-9:/_\s.-]+$

Required: No

KeyMaterialId

Unique identifier of the key material.

Type: String

Length Constraints: Fixed length of 64.

Pattern: ^[a-f0-9]+$

Required: No

KeyMaterialState

There are four possible values for this field: CURRENT, NON_CURRENT, PENDING_MULTI_REGION_IMPORT_AND_ROTATION and PENDING_ROTATION. AWS KMS uses CURRENT key material for both encryption and decryption and NON_CURRENT key material only for decryption. PENDING_ROTATION identifies key material that has been imported for on-demand key rotation but the rotation hasn't completed. The key material state PENDING_MULTI_REGION_IMPORT_AND_ROTATION is unique to multi-region, symmetric encryption keys with imported key material. It indicates key material that has been imported into the primary Region key but not all of the replica Region keys. When this key material is imported in to all of the replica Region keys, the key material state will change to PENDING_ROTATION. Key material in PENDING_MULTI_REGION_IMPORT_AND_ROTATION or PENDING_ROTATION state is not permanently associated with the KMS key.
You can delete this key material and import different key material in its place. The PENDING_MULTI_REGION_IMPORT_AND_ROTATION and PENDING_ROTATION values are only used in symmetric encryption keys with imported key material. The other values, CURRENT and NON_CURRENT, are used for all KMS keys that support automatic or on-demand key rotation.

Type: String

Valid Values: NON_CURRENT | CURRENT | PENDING_ROTATION | PENDING_MULTI_REGION_IMPORT_AND_ROTATION

Required: No

RotationDate

Date and time that the key material rotation completed. Formatted as Unix time. This field is not present for the first key material or an imported key material in PENDING_ROTATION state.

Type: Timestamp

Required: No

RotationType

Identifies whether the key material rotation was a scheduled automatic rotation or an on-demand rotation. This field is not present for the first key material or an imported key material in PENDING_ROTATION state.

Type: String

Valid Values: AUTOMATIC | ON_DEMAND

Required: No

ValidTo

Date and time at which the key material expires. This field is only present for symmetric encryption KMS keys with EXTERNAL origin in rotation list entries with an ExpirationModel value of KEY_MATERIAL_EXPIRES.

Type: Timestamp

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS SDK for C++

- AWS SDK for Java V2
- AWS SDK for Ruby V3

Tag

A key-value pair. A tag consists of a tag key and a tag value. Tag keys and tag values are both required, but tag values can be empty (null) strings.

> **Important**
>
> Do not include confidential or sensitive information in this field. This field may be displayed in plaintext in CloudTrail logs and other output.

For information about the rules that apply to tag keys and tag values, see User-Defined Tag Restrictions in the *AWS Billing and Cost Management User Guide*.

Contents

> **Note**
>
> In the following list, the required parameters are described first.

TagKey

The key of the tag.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 128.

Required: Yes

TagValue

The value of the tag.

Type: String

Length Constraints: Minimum length of 0. Maximum length of 256.

Required: Yes

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

XksKeyConfigurationType

Information about the external key that is associated with a KMS key in an external key store.

This element appears in a CreateKey or DescribeKey response only for a KMS key in an external key store.

The *external key* is a symmetric encryption key that is hosted by an external key manager outside of AWS. When you use the KMS key in an external key store in a cryptographic operation, the cryptographic operation is performed in the external key manager using the specified external key. For more information, see External key in the *AWS Key Management Service Developer Guide*.

Contents

> **Note**
>
> In the following list, the required parameters are described first.

Id

The ID of the external key in its external key manager. This is the ID that the external key store proxy uses to identify the external key.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 128.

Pattern: `^[a-zA-Z0-9-_.]+$`

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS SDK for C++
- AWS SDK for Java V2

- AWS SDK for Ruby V3

XksProxyAuthenticationCredentialType

AWS KMS uses the authentication credential to sign requests that it sends to the external key store proxy (XKS proxy) on your behalf. You establish these credentials on your external key store proxy and report them to AWS KMS.

The `XksProxyAuthenticationCredential` includes two required elements.

Contents

> **Note**
>
> In the following list, the required parameters are described first.

AccessKeyId

A unique identifier for the raw secret access key.

Type: String

Length Constraints: Minimum length of 20. Maximum length of 30.

Pattern: `^[A-Z2-7]+$`

Required: Yes

RawSecretAccessKey

A secret string of 43-64 characters. Valid characters are a-z, A-Z, 0-9, /, +, and =.

Type: String

Length Constraints: Minimum length of 43. Maximum length of 64.

Pattern: `^[a-zA-Z0-9\/+=]+$`

Required: Yes

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

XksProxyConfigurationType

Detailed information about the external key store proxy (XKS proxy). Your external key store proxy translates AWS KMS requests into a format that your external key manager can understand. These fields appear in a DescribeCustomKeyStores response only when the `CustomKeyStoreType` is `EXTERNAL_KEY_STORE`.

Contents

> **Note**
>
> In the following list, the required parameters are described first.

AccessKeyId

The part of the external key store proxy authentication credential that uniquely identifies the secret access key.

Type: String

Length Constraints: Minimum length of 20. Maximum length of 30.

Pattern: `^[A-Z2-7]+$`

Required: No

Connectivity

Indicates whether the external key store proxy uses a public endpoint or an Amazon VPC endpoint service to communicate with AWS KMS.

Type: String

Valid Values: `PUBLIC_ENDPOINT | VPC_ENDPOINT_SERVICE`

Required: No

UriEndpoint

The URI endpoint for the external key store proxy.

If the external key store proxy has a public endpoint, it is displayed here.

If the external key store proxy uses an Amazon VPC endpoint service name, this field displays the private DNS name associated with the VPC endpoint service.

Type: String

Length Constraints: Minimum length of 10. Maximum length of 128.

Pattern: `^https://[a-zA-Z0-9.-]+$`

Required: No

UriPath

The path to the external key store proxy APIs.

Type: String

Length Constraints: Minimum length of 10. Maximum length of 128.

Pattern: `^(/[a-zA-Z0-9\/_-]+/kms/xks/v\d{1,2})$|^(/kms/xks/v\d{1,2})$`

Required: No

VpcEndpointServiceName

The Amazon VPC endpoint service used to communicate with the external key store proxy. This field appears only when the external key store proxy uses an Amazon VPC endpoint service to communicate with AWS KMS.

Type: String

Length Constraints: Minimum length of 20. Maximum length of 64.

Pattern: `^(com|eu)\.amazonaws\.vpce\.([a-z]+-){2,3}\d+\.vpce-svc-[0-9a-z]+$`

Required: No

VpcEndpointServiceOwner

The AWS account ID that owns the Amazon VPC endpoint service used to communicate with the external key store proxy (XKS). This field appears only when the XKS uses an VPC endpoint service to communicate with AWS KMS.

Type: String

Length Constraints: Fixed length of 12.

Pattern: `[0-9]{12}`

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

Common Parameters

The following list contains the parameters that all actions use for signing Signature Version 4 requests with a query string. Any action-specific parameters are listed in the topic for that action. For more information about Signature Version 4, see Signing AWS API requests in the *IAM User Guide*.

X-Amz-Algorithm

The hash algorithm that you used to create the request signature.

Condition: Specify this parameter when you include authentication information in a query string instead of in the HTTP authorization header.

Type: string

Valid Values: `AWS4-HMAC-SHA256`

Required: Conditional

X-Amz-Credential

The credential scope value, which is a string that includes your access key, the date, the region you are targeting, the service you are requesting, and a termination string ("aws4_request"). The value is expressed in the following format: *access_key*/*YYYYMMDD*/*region*/*service*/aws4_request.

For more information, see Create a signed AWS API request in the *IAM User Guide*.

Condition: Specify this parameter when you include authentication information in a query string instead of in the HTTP authorization header.

Type: string

Required: Conditional

X-Amz-Date

The date that is used to create the signature. The format must be ISO 8601 basic format (YYYYMMDD'T'HHMMSS'Z'). For example, the following date time is a valid X-Amz-Date value: `20120325T120000Z`.

Condition: X-Amz-Date is optional for all requests; it can be used to override the date used for signing requests. If the Date header is specified in the ISO 8601 basic format, X-Amz-Date is not required. When X-Amz-Date is used, it always overrides the value of the Date header. For more information, see Elements of an AWS API request signature in the *IAM User Guide*.

Type: string

Required: Conditional

X-Amz-Security-Token

The temporary security token that was obtained through a call to AWS Security Token Service (AWS STS). For a list of services that support temporary security credentials from AWS STS, see AWS services that work with IAM in the *IAM User Guide*.

Condition: If you're using temporary security credentials from AWS STS, you must include the security token.

Type: string

Required: Conditional

X-Amz-Signature

Specifies the hex-encoded signature that was calculated from the string to sign and the derived signing key.

Condition: Specify this parameter when you include authentication information in a query string instead of in the HTTP authorization header.

Type: string

Required: Conditional

X-Amz-SignedHeaders

Specifies all the HTTP headers that were included as part of the canonical request. For more information about specifying signed headers, see Create a signed AWS API request in the *IAM User Guide*.

Condition: Specify this parameter when you include authentication information in a query string instead of in the HTTP authorization header.

Type: string

Required: Conditional

Common Error Types

This section lists common error types that this AWS service may return. Not all services return all error types listed here. For errors specific to an API action for this service, see the topic for that API action.

AccessDeniedException

You don't have permission to perform this action. Verify that your IAM policy includes the required permissions.

HTTP Status Code: 403

ExpiredTokenException

The security token included in the request has expired. Request a new security token and try again.

HTTP Status Code: 403

IncompleteSignature

The request signature doesn't conform to AWS standards. Verify that you're using valid AWS credentials and that your request is properly formatted. If you're using an SDK, ensure it's up to date.

HTTP Status Code: 403

InternalFailure

The request can't be processed right now because of an internal server issue. Try again later. If the problem persists, contact AWS Support.

HTTP Status Code: 500

MalformedHttpRequestException

The request body can't be processed. This typically happens when the request body can't be decompressed using the specified content encoding algorithm. Verify that the content encoding header matches the compression format used.

HTTP Status Code: 400

NotAuthorized

You don't have permissions to perform this action. Verify that your IAM policy includes the required permissions.

HTTP Status Code: 401

OptInRequired

Your AWS account needs a subscription for this service. Verify that you've enabled the service in your account.

HTTP Status Code: 403

RequestAbortedException

The request was aborted before a response could be returned. This typically happens when the client closes the connection.

HTTP Status Code: 400

RequestEntityTooLargeException

The request entity is too large. Reduce the size of the request body and try again.

HTTP Status Code: 413

RequestTimeoutException

The request timed out. The server didn't receive the complete request within the expected time frame. Try again.

HTTP Status Code: 408

ServiceUnavailable

The service is temporarily unavailable. Try again later.

HTTP Status Code: 503

ThrottlingException

Your request rate is too high. The AWS SDKs automatically retry requests that receive this exception. Reduce the frequency of requests.

HTTP Status Code: 400

UnknownOperationException

The action or operation isn't recognized. Verify that the action name is spelled correctly and that it's supported by the API version you're using.

HTTP Status Code: 404

UnrecognizedClientException

The X.509 certificate or AWS access key ID you provided doesn't exist in our records. Verify that you're using valid credentials and that they haven't expired.

HTTP Status Code: 403

ValidationError

The input doesn't meet the required format or constraints. Check that all required parameters are included and that values are valid.

HTTP Status Code: 400