AWS Key Management Service

API Reference

API Version 2014-11-01

AWS Key Management Service: API Reference

Table of Contents

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 is replacing the term *customer master key (CMK)* with *AWS KMS 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, 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 the FIPS endpoint in your preferred AWS Region. For more information about the available FIPS endpoints, see Service endpoints in the AWS Key Management Service topic of the *Amazon Web Services General Reference*.

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.

Signing Requests

Requests must be signed by 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 key for everyday work with AWS KMS. Instead, use the access key ID and secret access key for an IAM user. You can also use the AWS Security Token Service to generate temporary security credentials that you can use to sign requests.

All AWS KMS operations require 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 (p. 77)
- Decrypt (p. 37)
- GenerateDataKey (p. 83)
- GenerateDataKeyWithoutPlaintext (p. 99)

This document was last published on June 6, 2022.

Actions

The following actions are supported:

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 (p. 70).

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 (p. 192)

Request Syntax

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

Request Parameters

For information about the parameters that are common to all actions, see Common Parameters (p. 252).

The request accepts the following data in JSON format.

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

KeyId (p. 5)

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 (p. 151) or DescribeKey (p. 56).

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 (p. 5)

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 Errors (p. 254).

DependencyTimeoutException

The system timed out while trying to fulfill the request. The request can be retried.

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.

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 .

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
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java V2
- AWS SDK for JavaScript
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

ConnectCustomKeyStore

Connects or reconnects a custom key store to its associated AWS CloudHSM cluster.

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.

To connect a custom 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.

The connection process can take an extended amount of time to complete; up to 20 minutes. 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 (p. 52) operation.

During the connection process, 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.

The `ConnectCustomKeyStore` operation might fail for various reasons. To find the reason, use the DescribeCustomKeyStores (p. 52) operation and see the `ConnectionErrorCode` in the response. For help interpreting the `ConnectionErrorCode`, see CustomKeyStoresListEntry (p. 236).

To fix the failure, use the DisconnectCustomKeyStore (p. 67) operation to disconnect the custom key store, correct the error, use the UpdateCustomKeyStore (p. 213) operation if necessary, and then use `ConnectCustomKeyStore` again.

If you are having trouble connecting or disconnecting a custom key store, see Troubleshooting a Custom 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 (p. 15)
- DeleteCustomKeyStore (p. 46)
- DescribeCustomKeyStores (p. 52)
- DisconnectCustomKeyStore (p. 67)
- UpdateCustomKeyStore (p. 213)

Request Syntax

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

Request Parameters

For information about the parameters that are common to all actions, see Common Parameters (p. 252).

The request accepts the following data in JSON format.

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

CustomKeyStoreId (p. 8)

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 (p. 52) 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 Errors (p. 254).

CloudHsmClusterInvalidConfigurationException

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

- The 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 cluster. Do not delete or change them. To get information about a particular security group, use the DescribeSecurityGroups operation.
- The cluster must contain at least as many HSMs as the operation requires. To add HSMs, use the AWS CloudHSM CreateHsm operation.

  For the CreateCustomKeyStore (p. 15), UpdateCustomKeyStore (p. 213), and CreateKey (p. 26) operations, the AWS CloudHSM cluster must have at least two active HSMs, each in a different Availability Zone. For the ConnectCustomKeyStore (p. 8) 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 a custom 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 that is associated with the custom 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 (p. 52) operation.

This exception is thrown under the following conditions:

- You requested the CreateKey (p. 26) or GenerateRandom (p. 109) operation in a custom key store that is not connected. These operations are valid only when the custom key store `ConnectionState` is `CONNECTED`.
- You requested the UpdateCustomKeyStore (p. 213) or DeleteCustomKeyStore (p. 46) operation on a custom key store that is not disconnected. This operation is valid only when the custom key store `ConnectionState` is `DISCONNECTED`.
- You requested the ConnectCustomKeyStore (p. 8) operation on a custom key store with a `ConnectionState` of `DISCONNECTING` or `FAILED`. This operation is valid for all other `ConnectionState` values.

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
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java V2
- AWS SDK for JavaScript
- AWS SDK for PHP V3
- AWS SDK for Python
- 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 in 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 (p. 56) operation and in cryptographic operations, such as Encrypt (p. 77) and GenerateDataKey (p. 83). You can also change the KMS key that's associated with the alias (UpdateAlias (p. 209)) or delete the alias (DeleteAlias (p. 43)) 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 Using aliases 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 (p. 136) 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 (p. 43)
- ListAliases (p. 136)
- UpdateAlias (p. 209)

Request Syntax

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

Request Parameters

For information about the parameters that are common to all actions, see Common Parameters (p. 252).

The request accepts the following data in JSON format.

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

AliasName (p. 11)

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

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 (p. 11)

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 Finding the Key ID and 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 (p. 151) or DescribeKey (p. 56).

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 Errors (p. 254).

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. The request can be retried.

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.

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 .

HTTP Status Code: 400

LimitExceededException

The request was rejected because a 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
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java V2
- AWS SDK for JavaScript
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

CreateCustomKeyStore

Creates a custom key store that is associated with an AWS CloudHSM cluster that you own and manage.

This operation is part of the custom key store feature feature in AWS KMS, which combines the convenience and extensive integration of AWS KMS with the isolation and control of a single-tenant key store.

Before you create the custom key store, you must assemble the required elements, including an AWS CloudHSM cluster that fulfills the requirements for a custom key store. For details about the required elements, see Assemble the Prerequisites in the *AWS Key Management Service Developer Guide*.

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 (p. 8) operation to connect the new key store to its AWS CloudHSM cluster. 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.

For help with failures, see Troubleshooting a Custom 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:CreateCustomKeyStore (IAM policy).

Related operations:

- ConnectCustomKeyStore (p. 8)
- DeleteCustomKeyStore (p. 46)
- DescribeCustomKeyStores (p. 52)
- DisconnectCustomKeyStore (p. 67)
- UpdateCustomKeyStore (p. 213)

Request Syntax

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

Request Parameters

For information about the parameters that are common to all actions, see Common Parameters (p. 252).

The request accepts the following data in JSON format.

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

CustomKeyStoreName (p. 15)

Specifies a friendly name for the custom key store. The name must be unique in your AWS account.

Type: String

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

Required: Yes

CloudHsmClusterId (p. 15)

Identifies the AWS CloudHSM cluster for the custom key store. 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.

Required: No

KeyStorePassword (p. 15)

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 (p. 15)

Enter the content of the trust anchor certificate for the 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

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 (p. 16)

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 Errors (p. 254).

CloudHsmClusterInUseException

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

Clusters that share a backup history have the same cluster certificate. To view the cluster certificate of a 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 a custom key store.

- The 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 cluster. Do not delete or change them. To get information about a particular security group, use the DescribeSecurityGroups operation.
- The cluster must contain at least as many HSMs as the operation requires. To add HSMs, use the AWS CloudHSM CreateHsm operation.

  For the CreateCustomKeyStore (p. 15), UpdateCustomKeyStore (p. 213), and CreateKey (p. 26) operations, the AWS CloudHSM cluster must have at least two active HSMs, each in a different Availability Zone. For the ConnectCustomKeyStore (p. 8) 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 a custom 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 that is associated with the custom 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 is not the trust anchor certificate for the specified AWS CloudHSM cluster.

When you initialize the 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

See Also

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

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java V2
- AWS SDK for JavaScript
- AWS SDK for PHP V3
- AWS SDK for Python
- 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 (p. 56)) 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.

For detailed information about grants, including grant terminology, see Grants in AWS KMS in the AWS Key Management Service Developer Guide . For examples of working with grants in several programming languages, see Programming grants.

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 (p. 185) and RevokeGrant (p. 189) operations. To find the grant ID, use the ListGrants (p. 141) or ListRetirableGrants (p. 159) 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 (p. 141)
- ListRetirableGrants (p. 159)
- RetireGrant (p. 185)
- RevokeGrant (p. 189)

Request Syntax

```
{
   "Constraints": {
      "EncryptionContextEquals": {
         "string" : "string"
      },
      "EncryptionContextSubset": {
         "string" : "string"
      }
   },
   "GranteePrincipal": "string",
   "GrantTokens": [ "string" ],
```

```
   "KeyId": "string",
   "Name": "string",
   "Operations": [ "string" ],
   "RetiringPrincipal": "string"
}
```

Request Parameters

For information about the parameters that are common to all actions, see Common Parameters (p. 252).

The request accepts the following data in JSON format.

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

GranteePrincipal (p. 19)

The identity that gets the permissions specified in the grant.

To specify the principal, use the Amazon Resource Name (ARN) of an AWS principal. Valid AWS principals include AWS accounts (root), IAM users, IAM roles, federated users, and assumed role users. For examples of the ARN syntax to use for specifying a principal, see AWS Identity and Access Management (IAM) in the Example ARNs section of the *AWS General Reference*.

Type: String

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

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

Required: Yes

KeyId (p. 19)

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 (p. 151) or DescribeKey (p. 56).

Type: String

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

Required: Yes

Operations (p. 19)

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 (p. 197) operation, or a grant for an asymmetric KMS key that allows the

GenerateDataKey (p. 83) 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`

Required: Yes

Constraints (p. 19)

Specifies a grant constraint.

AWS KMS supports the `EncryptionContextEquals` and `EncryptionContextSubset` grant constraints. Each constraint value can include up to 8 encryption context pairs. The encryption context value in each constraint cannot exceed 384 characters. For information about grant constraints, see Using grant constraints in the *AWS Key Management Service Developer Guide*. For more information about encryption context, see Encryption context in the AWS Key Management Service Developer Guide .

The 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 this structure.

The encryption context grant constraints are supported only on grant operations that include an `EncryptionContext` parameter, such as cryptographic operations on symmetric encryption KMS keys. Grants with grant constraints can include the DescribeKey (p. 56) and RetireGrant (p. 185) operations, but the constraint doesn't apply to these operations. If a grant with a 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.

You cannot use an encryption context grant constraint for cryptographic operations with asymmetric KMS keys or HMAC KMS keys. These keys don't support an encryption context.

Type: GrantConstraints (p. 239) object

Required: No

GrantTokens (p. 19)

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 (p. 19)

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

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 (p. 19)

The principal that has permission to use the RetireGrant (p. 185) operation to retire the grant.

To specify the principal, use the Amazon Resource Name (ARN) of an AWS principal. Valid AWS principals include AWS accounts (root), IAM users, federated users, and assumed role users. For examples of the ARN syntax to use for specifying a principal, see AWS Identity and Access Management (IAM) in the Example ARNs section of the *AWS General Reference*.

The grant determines the retiring principal. Other principals might have permission to retire the grant or revoke the grant. For details, see RevokeGrant (p. 189) and Retiring and revoking grants in the *AWS Key Management Service Developer Guide*.

Type: String

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

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

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 (p. 22)

The unique identifier for the grant.

You can use the `GrantId` in a ListGrants (p. 141), RetireGrant (p. 185), or RevokeGrant (p. 189) operation.

Type: String

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

GrantToken (p. 22)

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 Errors (p. 254).

DependencyTimeoutException

The system timed out while trying to fulfill the request. The request can be retried.

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

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.

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 .

HTTP Status Code: 400

LimitExceededException

The request was rejected because a 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: a2d8d452-9fa8-11e6-b30c-dbb8ea4d97c5

{
  "GrantId": "0c237476b39f8bc44e45212e08498fbe3151305030726c0590dd8d3e9f3d6a60",
  "GrantToken":
 "AQpAM2RhZTk1MGMyNTk2ZmZmMzEyYWVhOWViN2I1MWM4Mzc0MWFiYjc0ZDE1ODkyNGFlNTIzODZhMzgyZjBlNGY3NiKIAgEBAgB4P
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
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java V2
- AWS SDK for JavaScript
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

CreateKey

Creates a unique customer managed KMS key in your AWS account and Region.

In addition to the required parameters, you can use the optional parameters to specify a key policy, description, tags, and other useful elements for any key type.

> **Note**
> AWS KMS is replacing the term *customer master key (CMK)* with *AWS KMS 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

To create a symmetric encryption KMS key, you aren't required to specify any parameters. The default value for `KeySpec`, `SYMMETRIC_DEFAULT`, and the default value for `KeyUsage`, `ENCRYPT_DECRYPT`, create a symmetric encryption KMS key.

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 (p. 83) and GenerateDataKeyPair (p. 89).

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 or an Elliptic Curve (ECC) key pair. The private key in an asymmetric KMS key never leaves AWS KMS unencrypted. However, you can use the GetPublicKey (p. 125) operation to download the public key so it can be used outside of AWS KMS. KMS keys with RSA key pairs can be used to encrypt or decrypt data or sign and verify messages (but not both). KMS keys with ECC key pairs can be used only to sign and verify messages. 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 (p. 105)) and verify (VerifyMac (p. 229)) HMAC codes for messages up to 4096 bytes.

HMAC KMS keys are not supported in all AWS Regions. If you try to create an HMAC KMS key in an AWS Region in which HMAC keys are not supported, the `CreateKey` operation returns an `UnsupportedOperationException`. For a list of Regions in which HMAC KMS keys are supported, see HMAC keys in AWS KMS in the *AWS Key Management Service Developer Guide*.

Multi-Region primary keys, Imported key material

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 (p. 178) operation. To change a replica key to a primary key, and its primary key to a replica key, use the UpdatePrimaryRegion (p. 221) 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*.

To import your own key material, begin by creating a symmetric encryption KMS key with no key material. To do this, use the `Origin` parameter of `CreateKey` with a value of `EXTERNAL`. Next, use GetParametersForImport (p. 120) operation to get a public key and import token, and use the public key to encrypt your key material. Then, use ImportKeyMaterial (p. 130) 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 .

This feature supports only symmetric encryption KMS keys, including multi-Region symmetric encryption KMS keys. You cannot import key material into any other type of KMS key.

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 (p. 178) operation. 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

To create a symmetric encryption KMS key in a custom key store, use the `CustomKeyStoreId` parameter to specify the custom key store. You must also 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.

Custom key stores support only symmetric encryption KMS keys. You cannot create an HMAC KMS key or an asymmetric KMS key in a custom key store. For information about custom key stores in AWS KMS see Custom key stores in AWS KMS in the AWS Key Management Service Developer Guide .

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 (p. 56)
- ListKeys (p. 151)

- ScheduleKeyDeletion (p. 192)

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"
      }
   ]
}
```

Request Parameters

For information about the parameters that are common to all actions, see Common Parameters (p. 252).

The request accepts the following data in JSON format.

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

BypassPolicyLockoutSafetyCheck (p. 28)

A flag to indicate whether to bypass the key policy lockout safety check.

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, refer to the scenario in the Default Key Policy section in the AWS Key Management Service Developer Guide .

Use this parameter only when you include a policy in the request and you intend to prevent the principal that is making the request from making a subsequent PutKeyPolicy (p. 164) request on the KMS key.

The default value is false.

Type: Boolean

Required: No

CustomerMasterKeySpec (p. 28)

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 will support 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`

Required: No

CustomKeyStoreId (p. 28)

Creates the KMS key in the specified custom key store and the key material in its associated AWS CloudHSM cluster. To create a KMS key in a custom key store, you must also specify 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, each in a different Availability Zone in the Region.

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.

To find the ID of a custom key store, use the DescribeCustomKeyStores (p. 52) operation.

The response includes the custom key store ID and the ID of the AWS CloudHSM cluster.

This operation is part of the custom key store feature feature in AWS KMS, which combines the convenience and extensive integration of AWS KMS with the isolation and control of a single-tenant key store.

Type: String

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

Required: No

Description (p. 28)

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).

To set or change the description after the key is created, use UpdateKeyDescription (p. 218).

Type: String

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

Required: No

KeySpec (p. 28)

Specifies the type of KMS key to create. The default value, `SYMMETRIC_DEFAULT`, creates a KMS key with a 256-bit symmetric key for encryption and decryption. For help choosing a key spec for your KMS key, see Choosing a KMS key type 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 cryptographic 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 or kms:Signing Algorithm 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` (AES-256-GCM)
- HMAC keys (symmetric)
  - `HMAC_224`
  - `HMAC_256`
  - `HMAC_384`
  - `HMAC_512`
- Asymmetric RSA key pairs
  - `RSA_2048`
  - `RSA_3072`
  - `RSA_4096`
- Asymmetric NIST-recommended elliptic curve key pairs
  - `ECC_NIST_P256` (secp256r1)
  - `ECC_NIST_P384` (secp384r1)
  - `ECC_NIST_P521` (secp521r1)
- Other asymmetric elliptic curve key pairs
  - `ECC_SECG_P256K1` (secp256k1), commonly used for cryptocurrencies.

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`

Required: No

KeyUsage (p. 28)

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.

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 material, specify `ENCRYPT_DECRYPT` or `SIGN_VERIFY`.
- For asymmetric KMS keys with ECC key material, specify `SIGN_VERIFY`.

Type: String

Valid Values: `SIGN_VERIFY | ENCRYPT_DECRYPT | GENERATE_VERIFY_MAC`

Required: No

MultiRegion (p. 28)

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 (p. 178) operation.

You can create a multi-Region version of a symmetric encryption KMS key, an HMAC KMS key, an asymmetric KMS key, or a KMS key with imported key material. However, you cannot create a multi-Region key in a custom key store.

Type: Boolean

Required: No

Origin (p. 28)

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 the 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*. This value is valid only for symmetric encryption KMS keys.

To create a KMS key in an AWS KMS custom 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 custom key store. This value is valid only for symmetric encryption KMS keys.

Type: String

Valid Values: `AWS_KMS | EXTERNAL | AWS_CLOUDHSM`

Required: No

Policy (p. 28)

The key policy to attach to the KMS key. If you do not specify 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*.

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

- If you don't set `BypassPolicyLockoutSafetyCheck` to `True`, the key policy must allow the principal that is making the `CreateKey` request to make a subsequent PutKeyPolicy (p. 164) request on the KMS key. This reduces the risk that the KMS key becomes unmanageable. For more information, refer to the scenario in the Default Key Policy section of the AWS Key Management Service Developer Guide .
- 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 (for example, an IAM user or role), 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 (p. 28)

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 (p. 202) operation.

> **Note**
> Tagging or untagging a KMS key can allow or deny permission to the KMS key. For details, see ABAC in 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 Tagging Keys.

Type: Array of Tag (p. 251) objects

Required: No

Response Syntax

```
{
   "KeyMetadata": {
      "Arn": "string",
      "AWSAccountId": "string",
      "CloudHsmClusterId": "string",
      "CreationDate": number,
      "CustomerMasterKeySpec": "string",
      "CustomKeyStoreId": "string",
      "DeletionDate": number,
      "Description": "string",
      "Enabled": boolean,
      "EncryptionAlgorithms": [ "string" ],
      "ExpirationModel": "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
   }
}
```

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 (p. 32)

Metadata associated with the KMS key.

Type: KeyMetadata (p. 244) object

Errors

For information about the errors that are common to all actions, see Common Errors (p. 254).

CloudHsmClusterInvalidConfigurationException

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

- The 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 cluster. Do not delete or change them. To get information about a particular security group, use the DescribeSecurityGroups operation.
- The cluster must contain at least as many HSMs as the operation requires. To add HSMs, use the AWS CloudHSM CreateHsm operation.

  For the CreateCustomKeyStore (p. 15), UpdateCustomKeyStore (p. 213), and CreateKey (p. 26) operations, the AWS CloudHSM cluster must have at least two active HSMs, each in a different Availability Zone. For the ConnectCustomKeyStore (p. 8) 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 a custom 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 (p. 52) operation.

This exception is thrown under the following conditions:

- You requested the CreateKey (p. 26) or GenerateRandom (p. 109) operation in a custom key store that is not connected. These operations are valid only when the custom key store `ConnectionState` is `CONNECTED`.
- You requested the UpdateCustomKeyStore (p. 213) or DeleteCustomKeyStore (p. 46) operation on a custom key store that is not disconnected. This operation is valid only when the custom key store `ConnectionState` is `DISCONNECTED`.
- You requested the ConnectCustomKeyStore (p. 8) operation on a custom key store with a `ConnectionState` of `DISCONNECTING` or `FAILED`. This operation is valid for all other `ConnectionState` values.

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. The request can be retried.

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 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

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": 1.499288695918E9,
    "CustomerMasterKeySpec": "SYMMETRIC_DEFAULT",
    "Description": "",
    "Enabled": true,
    "EncryptionAlgorithms": [
        "SYMMETRIC_DEFAULT"
    ],
    "KeyId": "1234abcd-12ab-34cd-56ef-1234567890ab",
    "KeyManager": "CUSTOMER",
    "KeySpec": "SYMMETRIC_DEFAULT",
    "KeyState": "Enabled",
```

```
    "KeyUsage": "ENCRYPT_DECRYPT",
    "MultiRegion": false,
    "Origin": "AWS_KMS"
  }
}
```

See Also

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

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java V2
- AWS SDK for JavaScript
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

Decrypt

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

- Encrypt (p. 77)
- GenerateDataKey (p. 83)
- GenerateDataKeyPair (p. 89)
- GenerateDataKeyWithoutPlaintext (p. 99)
- GenerateDataKeyPairWithoutPlaintext (p. 94)

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 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 user 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*.

Applications in AWS Nitro Enclaves can call this operation by using the AWS Nitro Enclaves Development Kit. For information about the supporting parameters, see How AWS Nitro Enclaves use 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:Decrypt (key policy)

Related operations:

- Encrypt (p. 77)
- GenerateDataKey (p. 83)
- GenerateDataKeyPair (p. 89)
- ReEncrypt (p. 170)

Request Syntax

```
{
   "CiphertextBlob": blob,
   "EncryptionAlgorithm": "string",
   "EncryptionContext": {
      "string" : "string"
   },
   "GrantTokens": [ "string" ],
   "KeyId": "string"
}
```

Request Parameters

For information about the parameters that are common to all actions, see Common Parameters (p. 252).

The request accepts the following data in JSON format.

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

CiphertextBlob (p. 38)

Ciphertext to be decrypted. The blob includes metadata.

Type: Base64-encoded binary data object

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

Required: Yes

EncryptionAlgorithm (p. 38)

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`

Required: No

EncryptionContext (p. 38)

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 (p. 38)

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 (p. 38)

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. 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 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 (p. 151) or DescribeKey (p. 56). To get the alias name and alias ARN, use ListAliases (p. 136).

Type: String

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

Required: No

Response Syntax

```
{
   "EncryptionAlgorithm": "string",
   "KeyId": "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.

EncryptionAlgorithm (p. 39)

The encryption algorithm that was used to decrypt the ciphertext.

Type: String

Valid Values: `SYMMETRIC_DEFAULT | RSAES_OAEP_SHA_1 | RSAES_OAEP_SHA_256`

KeyId (p. 39)

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.

Plaintext (p. 39)

Decrypted plaintext 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 4096.

Errors

For information about the errors that are common to all actions, see Common Errors (p. 254).

DependencyTimeoutException

The system timed out while trying to fulfill the request. The request can be retried.

HTTP Status Code: 500

DisabledException

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

HTTP Status Code: 400

IncorrectKeyException

The request was rejected because the specified KMS key cannot decrypt the data. The `KeyId` in a Decrypt (p. 37) request and the `SourceKeyId` in a ReEncrypt (p. 170) request must identify the same KMS key that was used to encrypt the ciphertext.

HTTP Status Code: 400

InvalidCiphertextException

From the Decrypt (p. 37) or ReEncrypt (p. 170) 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 (p. 130) operation, the request was rejected because AWS KMS could not decrypt the encrypted (wrapped) key material.

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`. To find the `KeyUsage` of a KMS key, use the DescribeKey (p. 56) operation.

To find the encryption or signing algorithms supported for a particular KMS key, use the DescribeKey (p. 56) 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.

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 .

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
+Qsu6QISBSTFw5VO5nsIUpWFKbkCFQa4AAAB9MHsGCSqGSIb3DQEHBqBuMGwCAQAwZwYJKoZIhvcNAQcBMB4GCWCGSAFlAwQBLjARBA
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",
  "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
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java V2
- AWS SDK for JavaScript
- AWS SDK for PHP V3
- AWS SDK for Python
- 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 in 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 (p. 56) operation. To get the aliases of all KMS keys, use the ListAliases (p. 136) operation.

Each KMS key can have multiple aliases. To change the alias of a KMS key, use DeleteAlias (p. 43) to delete the current alias and CreateAlias (p. 11) to create a new alias. To associate an existing alias with a different KMS key, call UpdateAlias (p. 209).

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 (p. 11)
- ListAliases (p. 136)
- UpdateAlias (p. 209)

Request Syntax

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

Request Parameters

For information about the parameters that are common to all actions, see Common Parameters (p. 252).

The request accepts the following data in JSON format.

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

AliasName (p. 43)

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 Errors (p. 254).

DependencyTimeoutException

The system timed out while trying to fulfill the request. The request can be retried.

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.

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 .

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
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java V2
- AWS SDK for JavaScript
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

DeleteCustomKeyStore

Deletes a custom key store. This operation does not delete the AWS CloudHSM cluster that is associated with the custom key store, or affect any users or keys in the cluster.

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 (p. 192) to delete the KMS keys from the key store. When the scheduled waiting period expires, the `ScheduleKeyDeletion` operation deletes the KMS keys. Then it 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.

After all KMS keys are deleted from AWS KMS, use DisconnectCustomKeyStore (p. 67) to disconnect the key store from AWS KMS. Then, you can delete the custom key store.

Instead of deleting the custom key store, consider using DisconnectCustomKeyStore (p. 67) to disconnect it from AWS KMS. 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.

This operation is part of the custom key store feature feature in AWS KMS, which combines the convenience and extensive integration of AWS KMS with the isolation and control of a single-tenant key store.

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 (p. 8)
- CreateCustomKeyStore (p. 15)
- DescribeCustomKeyStores (p. 52)
- DisconnectCustomKeyStore (p. 67)
- UpdateCustomKeyStore (p. 213)

Request Syntax

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

Request Parameters

For information about the parameters that are common to all actions, see Common Parameters (p. 252).

The request accepts the following data in JSON format.

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

CustomKeyStoreId (p. 46)

Enter the ID of the custom key store you want to delete. To find the ID of a custom key store, use the DescribeCustomKeyStores (p. 52) 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 Errors (p. 254).

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 (p. 192) 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 (p. 52) operation.

This exception is thrown under the following conditions:

- You requested the CreateKey (p. 26) or GenerateRandom (p. 109) operation in a custom key store that is not connected. These operations are valid only when the custom key store `ConnectionState` is `CONNECTED`.
- You requested the UpdateCustomKeyStore (p. 213) or DeleteCustomKeyStore (p. 46) operation on a custom key store that is not disconnected. This operation is valid only when the custom key store `ConnectionState` is `DISCONNECTED`.
- You requested the ConnectCustomKeyStore (p. 8) operation on a custom key store with a `ConnectionState` of `DISCONNECTING` or `FAILED`. This operation is valid for all other `ConnectionState` values.

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
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java V2
- AWS SDK for JavaScript
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

DeleteImportedKeyMaterial

Deletes key material that you previously imported. This operation makes the specified KMS key unusable. 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`.

After you delete key material, you can use ImportKeyMaterial (p. 130) to reimport the same key material into the KMS 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:DeleteImportedKeyMaterial (key policy)

Related operations:

- GetParametersForImport (p. 120)
- ImportKeyMaterial (p. 130)

Request Syntax

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

Request Parameters

For information about the parameters that are common to all actions, see Common Parameters (p. 252).

The request accepts the following data in JSON format.

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

KeyId (p. 49)

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 (p. 151) or DescribeKey (p. 56).

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 Errors (p. 254).

DependencyTimeoutException

The system timed out while trying to fulfill the request. The request can be retried.

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.

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 .

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
```

See Also

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

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java V2
- AWS SDK for JavaScript
- AWS SDK for PHP V3
- AWS SDK for Python
- 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 store feature feature in AWS KMS, which combines the convenience and extensive integration of AWS KMS with the isolation and control of a single-tenant key store.

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, 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 (p. 236).

Custom key stores have a `DISCONNECTED` connection state if the key store has never been connected or you use the DisconnectCustomKeyStore (p. 67) operation to disconnect it. If your custom key store state is `CONNECTED` but you are having trouble using it, make sure that its associated AWS CloudHSM cluster is active and contains the minimum number of HSMs required for the operation, if any.

For help repairing your custom key store, see the Troubleshooting Custom Key Stores topic 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 (p. 8)
- CreateCustomKeyStore (p. 15)
- DeleteCustomKeyStore (p. 46)
- DisconnectCustomKeyStore (p. 67)
- UpdateCustomKeyStore (p. 213)

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 (p. 252).

The request accepts the following data in JSON format.

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

CustomKeyStoreId (p. 52)

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, you can use either the `CustomKeyStoreId` or `CustomKeyStoreName` parameter, but not both.

Type: String

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

Required: No

CustomKeyStoreName (p. 52)

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, you can use either the `CustomKeyStoreId` or `CustomKeyStoreName` parameter, but not both.

Type: String

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

Required: No

Limit (p. 52)

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 (p. 52)

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",
            "TrustAnchorCertificate": "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 (p. 53)

Contains metadata about each custom key store.

Type: Array of CustomKeyStoresListEntry (p. 236) objects

NextMarker (p. 53)

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 (p. 53)

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 thisresponse to the `Marker` parameter in a subsequent request.

Type: Boolean

Errors

For information about the errors that are common to all actions, see Common Errors (p. 254).

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
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java V2
- AWS SDK for JavaScript
- AWS SDK for PHP V3
- AWS SDK for Python
- 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 KMS keys in custom key stores, it includes information about the custom key store, such as the key store ID and the AWS CloudHSM cluster ID. For multi-Region keys, it displays the primary key and all related replica keys.

`DescribeKey` does not return the following information:

- Aliases associated with the KMS key. To get this information, use ListAliases (p. 136).
- Whether automatic key rotation is enabled on the KMS key. To get this information, use GetKeyRotationStatus (p. 116). Also, some key states prevent a KMS key from being automatically rotated. For details, see How Automatic Key Rotation Works in *AWS Key Management Service Developer Guide*.
- Tags on the KMS key. To get this information, use ListResourceTags (p. 155).
- Key policies and grants on the KMS key. To get this information, use GetKeyPolicy (p. 112) and ListGrants (p. 141).

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 (p. 112)
- GetKeyRotationStatus (p. 116)
- ListAliases (p. 136)
- ListGrants (p. 141)
- ListKeys (p. 151)
- ListResourceTags (p. 155)
- ListRetirableGrants (p. 159)

Request Syntax

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

Request Parameters

For information about the parameters that are common to all actions, see Common Parameters (p. 252).

The request accepts the following data in JSON format.

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

KeyId (p. 56)

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 (p. 151) or DescribeKey (p. 56). To get the alias name and alias ARN, use ListAliases (p. 136).

Type: String

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

Required: Yes

GrantTokens (p. 56)

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,
      "CustomerMasterKeySpec": "string",
      "CustomKeyStoreId": "string",
      "DeletionDate": number,
```

```
      "Description": "string",
      "Enabled": boolean,
      "EncryptionAlgorithms": [ "string" ],
      "ExpirationModel": "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
   }
}
```

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 (p. 57)

Metadata associated with the key.

Type: KeyMetadata (p. 244) object

Errors

For information about the errors that are common to all actions, see Common Errors (p. 254).

DependencyTimeoutException

The system timed out while trying to fulfill the request. The request can be retried.

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": 1.499288695918E9,
    "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"
    ]
  }
```

```
}
```

See Also

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

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java V2
- AWS SDK for JavaScript
- AWS SDK for PHP V3
- AWS SDK for Python
- 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.

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 .

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:DisableKey (key policy)

Related operations: EnableKey (p. 70)

Request Syntax

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

Request Parameters

For information about the parameters that are common to all actions, see Common Parameters (p. 252).

The request accepts the following data in JSON format.

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

KeyId (p. 61)

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 (p. 151) or DescribeKey (p. 56).

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 Errors (p. 254).

DependencyTimeoutException

The system timed out while trying to fulfill the request. The request can be retried.

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.

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 .

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
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java V2
- AWS SDK for JavaScript
- AWS SDK for PHP V3
- AWS SDK for Python
- 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 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. The key rotation status of these KMS keys is always `false`. 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 (p. 73)) 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 (p. 73).

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 (p. 73)
- GetKeyRotationStatus (p. 116)

Request Syntax

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

Request Parameters

For information about the parameters that are common to all actions, see Common Parameters (p. 252).

The request accepts the following data in JSON format.

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

KeyId (p. 64)

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 (p. 151) or DescribeKey (p. 56).

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 Errors (p. 254).

DependencyTimeoutException

The system timed out while trying to fulfill the request. The request can be retried.

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.

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 .

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
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java V2
- AWS SDK for JavaScript
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

DisconnectCustomKeyStore

Disconnects the custom key store from its associated AWS CloudHSM cluster. While a custom key store is disconnected, you can manage the custom key store and its KMS keys, but you cannot create or use KMS keys in the custom key store. 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.

To find the connection state of a custom key store, use the DescribeCustomKeyStores (p. 52) operation. To reconnect a custom key store, use the ConnectCustomKeyStore (p. 8) operation.

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

This operation is part of the custom key store feature feature in AWS KMS, which combines the convenience and extensive integration of AWS KMS with the isolation and control of a single-tenant key store.

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 (p. 8)
- CreateCustomKeyStore (p. 15)
- DeleteCustomKeyStore (p. 46)
- DescribeCustomKeyStores (p. 52)
- UpdateCustomKeyStore (p. 213)

Request Syntax

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

Request Parameters

For information about the parameters that are common to all actions, see Common Parameters (p. 252).

The request accepts the following data in JSON format.

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

CustomKeyStoreId (p. 67)

Enter the ID of the custom key store you want to disconnect. To find the ID of a custom key store, use the DescribeCustomKeyStores (p. 52) 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 Errors (p. 254).

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 (p. 52) operation.

This exception is thrown under the following conditions:

- You requested the CreateKey (p. 26) or GenerateRandom (p. 109) operation in a custom key store that is not connected. These operations are valid only when the custom key store `ConnectionState` is `CONNECTED`.
- You requested the UpdateCustomKeyStore (p. 213) or DeleteCustomKeyStore (p. 46) operation on a custom key store that is not disconnected. This operation is valid only when the custom key store `ConnectionState` is `DISCONNECTED`.
- You requested the ConnectCustomKeyStore (p. 8) operation on a custom key store with a `ConnectionState` of `DISCONNECTING` or `FAILED`. This operation is valid for all other `ConnectionState` values.

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
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java V2
- AWS SDK for JavaScript
- AWS SDK for PHP V3

- AWS SDK for Python
- 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 (p. 61)

Request Syntax

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

Request Parameters

For information about the parameters that are common to all actions, see Common Parameters (p. 252).

The request accepts the following data in JSON format.

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

KeyId (p. 70)

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 (p. 151) or DescribeKey (p. 56).

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 Errors (p. 254).

DependencyTimeoutException

The system timed out while trying to fulfill the request. The request can be retried.

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.

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 .

HTTP Status Code: 400

LimitExceededException

The request was rejected because a 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
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java V2
- AWS SDK for JavaScript
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

EnableKeyRotation

Enables automatic rotation of the key material of the specified symmetric encryption KMS key.

When you enable automatic rotation of acustomer 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 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 (p. 64) operation.

Automatic key rotation is supported only on symmetric encryption KMS keys. 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. The key rotation status of these KMS keys is always `false`. 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 AWS managed KMS keys. AWS KMS always rotates the key material of AWS managed keys 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 (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 (p. 64)
- GetKeyRotationStatus (p. 116)

Request Syntax

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

Request Parameters

For information about the parameters that are common to all actions, see Common Parameters (p. 252).

The request accepts the following data in JSON format.

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

KeyId (p. 73)

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. The key rotation status of these KMS keys is always `false`. 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 (p. 151) or DescribeKey (p. 56).

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 Errors (p. 254).

DependencyTimeoutException

The system timed out while trying to fulfill the request. The request can be retried.

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.

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 .

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"}
```

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
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java V2

- AWS SDK for JavaScript
- AWS SDK for PHP V3
- AWS SDK for Python
- 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 (p. 83) and GenerateDataKeyPair (p. 89) 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 type.

> **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

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 (p. 37)

- GenerateDataKey (p. 83)
- GenerateDataKeyPair (p. 89)

Request Syntax

```
{
   "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 (p. 252).

The request accepts the following data in JSON format.

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

KeyId (p. 78)

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 (p. 56) 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 (p. 151) or DescribeKey (p. 56). To get the alias name and alias ARN, use ListAliases (p. 136).

Type: String

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

Required: Yes

Plaintext (p. 78)

Data to be encrypted.

Type: Base64-encoded binary data object

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

Required: Yes

EncryptionAlgorithm (p. 78)

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.

Type: String

Valid Values: `SYMMETRIC_DEFAULT | RSAES_OAEP_SHA_1 | RSAES_OAEP_SHA_256`

Required: No

EncryptionContext (p. 78)

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.

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 (p. 78)

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 (p. 79)

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 (p. 79)

The encryption algorithm that was used to encrypt the plaintext.

Type: String

Valid Values: `SYMMETRIC_DEFAULT | RSAES_OAEP_SHA_1 | RSAES_OAEP_SHA_256`

KeyId (p. 79)

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 Errors (p. 254).

DependencyTimeoutException

The system timed out while trying to fulfill the request. The request can be retried.

HTTP Status Code: 500

DisabledException

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

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`. To find the `KeyUsage` of a KMS key, use the DescribeKey (p. 56) operation.

To find the encryption or signing algorithms supported for a particular KMS key, use the DescribeKey (p. 56) 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.

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 .

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
+Qsu6QISBSTFw5VO5nsIUpWFKbkCFQa4AAAB9MHsGCSqGSIb3DQEHBqBuMGwCAQAwZwYJKoZIhvcNAQcBMB4GCWCGSAFlAwQBLjARBA
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
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java V2
- AWS SDK for JavaScript
- AWS SDK for PHP V3
- AWS SDK for Python
- 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 (p. 56) 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 get only an encrypted copy of the data key, use GenerateDataKeyWithoutPlaintext (p. 99). To generate an asymmetric data key pair, use the GenerateDataKeyPair (p. 89) or GenerateDataKeyPairWithoutPlaintext (p. 94) operation. To get a cryptographically secure random byte string, use GenerateRandom (p. 109).

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*.

Applications in AWS Nitro Enclaves can call this operation by using the AWS Nitro Enclaves Development Kit. For information about the supporting parameters, see How AWS Nitro Enclaves use 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 (p. 37) 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 (p. 37)
- Encrypt (p. 77)
- GenerateDataKeyPair (p. 89)
- GenerateDataKeyPairWithoutPlaintext (p. 94)
- GenerateDataKeyWithoutPlaintext (p. 99)

Request Syntax

```
{
   "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 (p. 252).

The request accepts the following data in JSON format.

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

KeyId (p. 84)

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 (p. 56) 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 (p. 151) or DescribeKey (p. 56). To get the alias name and alias ARN, use ListAliases (p. 136).

Type: String

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

Required: Yes

EncryptionContext (p. 84)

Specifies the encryption context that will be used when encrypting the data key.

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 (p. 84)

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 (p. 84)

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 (p. 84)

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

Response Syntax

```
{
```

```
   "CiphertextBlob": blob,
   "KeyId": "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 (p. 85)

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.

KeyId (p. 85)

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.

Plaintext (p. 85)

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.

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 Errors (p. 254).

DependencyTimeoutException

The system timed out while trying to fulfill the request. The request can be retried.

HTTP Status Code: 500

DisabledException

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

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`. To find the `KeyUsage` of a KMS key, use the DescribeKey (p. 56) operation.

To find the encryption or signing algorithms supported for a particular KMS key, use the DescribeKey (p. 56) 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.

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 .

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":
 "AQEDAHjRYf5WytIc0C857tFSnBaPn2F8DgfmThbJlGfR8P3WlwAAAH4wfAYJKoZIhvcNAQcGoG8wbQIBADBoBgkqhkiG9w0BBwEwH
+YdhV8MrkBQPeac0ReRVNDt9qleAt+SHgIRF8P0H+7U=",
  "KeyId": "arn:aws:kms:us-east-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
  "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
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java V2
- AWS SDK for JavaScript
- AWS SDK for PHP V3
- AWS SDK for Python
- 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 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 (p. 37) 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 (p. 56) operation.

Use the `KeyPairSpec` parameter to choose an RSA or Elliptic Curve (ECC) data key pair. AWS KMS recommends that your use ECC key pairs for signing, and use RSA 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 (p. 94) 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 (p. 37) 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.

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 (p. 37)
- Encrypt (p. 77)
- GenerateDataKey (p. 83)
- GenerateDataKeyPairWithoutPlaintext (p. 94)
- GenerateDataKeyWithoutPlaintext (p. 99)

Request Syntax

```
{
   "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 (p. 252).

The request accepts the following data in JSON format.

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

KeyId (p. 90)

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 (p. 56) 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 (p. 151) or DescribeKey (p. 56). To get the alias name and alias ARN, use ListAliases (p. 136).

Type: String

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

Required: Yes

KeyPairSpec (p. 90)

Determines the type of data key pair that is generated.

The AWS KMS rule that restricts the use of asymmetric RSA KMS keys to encrypt and decrypt or to sign and verify (but not both), and the rule that permits you to use ECC KMS keys only to sign and verify, are not effective on data key pairs, which are used outside of AWS KMS.

Type: String

Valid Values: `RSA_2048 | RSA_3072 | RSA_4096 | ECC_NIST_P256 | ECC_NIST_P384 | ECC_NIST_P521 | ECC_SECG_P256K1`

Required: Yes

EncryptionContext (p. 90)

Specifies the encryption context that will be used when encrypting the private key in the data key pair.

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 (p. 90)

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",
   "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.

KeyId (p. 91)

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.

KeyPairSpec (p. 91)

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`

PrivateKeyCiphertextBlob (p. 91)

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 (p. 91)

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.

Type: Base64-encoded binary data object

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

PublicKey (p. 91)

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 Errors (p. 254).

DependencyTimeoutException

The system timed out while trying to fulfill the request. The request can be retried.

HTTP Status Code: 500

DisabledException

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

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`. To find the `KeyUsage` of a KMS key, use the DescribeKey (p. 56) operation.

To find the encryption or signing algorithms supported for a particular KMS key, use the DescribeKey (p. 56) 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.

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 .

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
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java V2
- AWS SDK for JavaScript
- AWS SDK for PHP V3
- AWS SDK for Python
- 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 (p. 89), 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 (p. 37) 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 (p. 56) operation.

Use the `KeyPairSpec` parameter to choose an RSA or Elliptic Curve (ECC) data key pair. AWS KMS recommends that your use ECC key pairs for signing, and use RSA 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.

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 (p. 37)
- Encrypt (p. 77)
- GenerateDataKey (p. 83)
- GenerateDataKeyPair (p. 89)
- GenerateDataKeyWithoutPlaintext (p. 99)

Request Syntax

```
{
   "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 (p. 252).

The request accepts the following data in JSON format.

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

KeyId (p. 94)

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 (p. 56) 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 (p. 151) or DescribeKey (p. 56). To get the alias name and alias ARN, use ListAliases (p. 136).

Type: String

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

Required: Yes

KeyPairSpec (p. 94)

Determines the type of data key pair that is generated.

The AWS KMS rule that restricts the use of asymmetric RSA KMS keys to encrypt and decrypt or to sign and verify (but not both), and the rule that permits you to use ECC KMS keys only to sign and verify, are not effective on data key pairs, which are used outside of AWS KMS.

Type: String

Valid Values: `RSA_2048 | RSA_3072 | RSA_4096 | ECC_NIST_P256 | ECC_NIST_P384 | ECC_NIST_P521 | ECC_SECG_P256K1`

Required: Yes

EncryptionContext (p. 94)

Specifies the encryption context that will be used when encrypting the private key in the data key pair.

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 (p. 94)

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",
   "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 (p. 96)

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.

KeyPairSpec (p. 96)

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`

PrivateKeyCiphertextBlob (p. 96)

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 (p. 96)

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 Errors (p. 254).

DependencyTimeoutException

The system timed out while trying to fulfill the request. The request can be retried.

HTTP Status Code: 500

DisabledException

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

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`. To find the `KeyUsage` of a KMS key, use the DescribeKey (p. 56) operation.

To find the encryption or signing algorithms supported for a particular KMS key, use the DescribeKey (p. 56) 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.

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 .

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
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java V2
- AWS SDK for JavaScript
- AWS SDK for PHP V3
- AWS SDK for Python
- 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 (p. 83) 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 (p. 37) 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 (p. 89) or GenerateDataKeyPairWithoutPlaintext (p. 94) 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 (p. 56) operation.

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 (p. 37)
- Encrypt (p. 77)
- GenerateDataKey (p. 83)
- GenerateDataKeyPair (p. 89)
- GenerateDataKeyPairWithoutPlaintext (p. 94)

Request Syntax

```
{
   "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 (p. 252).

The request accepts the following data in JSON format.

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

KeyId (p. 99)

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 (p. 56) 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 (p. 151) or DescribeKey (p. 56). To get the alias name and alias ARN, use ListAliases (p. 136).

Type: String

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

Required: Yes

EncryptionContext (p. 99)

Specifies the encryption context that will be used when encrypting the data key.

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 (p. 99)

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 (p. 99)

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 (p. 99)

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"
}
```

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 (p. 101)

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 (p. 101)

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.

Errors

For information about the errors that are common to all actions, see Common Errors (p. 254).

DependencyTimeoutException

The system timed out while trying to fulfill the request. The request can be retried.

HTTP Status Code: 500

DisabledException

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

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`. To find the `KeyUsage` of a KMS key, use the DescribeKey (p. 56) operation.

To find the encryption or signing algorithms supported for a particular KMS key, use the DescribeKey (p. 56) 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.

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 .

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":
 "AQEDAHjRYf5WytIc0C857tFSnBaPn2F8DgfmThbJlGfR8P3WlwAAAH4wfAYJKoZIhvcNAQcGoG8wbQIBADBoBgkqhkiG9w0BBwEwH
ntdQTL16wQIBEIA7BE/3LB7F1meU8z4e1vEKBGZgXPwMvkZXbKnf3wxCD9lB4hU29lii4euOqxp8pESb
+7oCN9f1R75ac3s=",
  "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
- AWS SDK for .NET
- AWS SDK for C++

- AWS SDK for Go
- AWS SDK for Java V2
- AWS SDK for JavaScript
- AWS SDK for PHP V3
- AWS SDK for Python
- 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. The MAC algorithm computes the HMAC for the message and the key as described in RFC 2104.

You can use the HMAC that this operation generates with the VerifyMac (p. 229) 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. 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 (p. 229)

Request Syntax

```
{
   "GrantTokens": [ "string" ],
   "KeyId": "string",
   "MacAlgorithm": "string",
   "Message": blob
}
```

Request Parameters

For information about the parameters that are common to all actions, see Common Parameters (p. 252).

The request accepts the following data in JSON format.

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

KeyId (p. 105)

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 (p. 56) operation and see the `KeySpec` field in the response.

Type: String

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

Required: Yes

MacAlgorithm (p. 105)

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 (p. 56) 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 (p. 105)

The message to be hashed. Specify a message of up to 4,096 bytes.

`GenerateMac` and VerifyMac (p. 229) 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

GrantTokens (p. 105)

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 (p. 106)

The HMAC KMS key used in the operation.

Type: String

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

Mac (p. 106)

The hash-based message authentication code (HMAC) for the given message, key, and MAC algorithm.

Type: Base64-encoded binary data object

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

MacAlgorithm (p. 106)

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 Errors (p. 254).

DisabledException

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

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`. To find the `KeyUsage` of a KMS key, use the DescribeKey (p. 56) operation.

To find the encryption or signing algorithms supported for a particular KMS key, use the DescribeKey (p. 56) 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.

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 .

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
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java V2
- AWS SDK for JavaScript
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

GenerateRandom

Returns a random byte string that is cryptographically secure.

By default, the random byte string is generated in AWS KMS. To generate the byte string in the AWS CloudHSM cluster that is associated with a custom key store, specify the custom key store ID.

Applications in AWS Nitro Enclaves can call this operation by using the AWS Nitro Enclaves Development Kit. For information about the supporting parameters, see How AWS Nitro Enclaves use AWS KMS in the *AWS Key Management Service Developer Guide*.

For more information about entropy and random number generation, see AWS Key Management Service Cryptographic Details.

Cross-account use: Not applicable. `GenerateRandom` does not use any account-specific resources, such as KMS keys.

Required permissions: kms:GenerateRandom (IAM policy)

Request Syntax

```
{
   "CustomKeyStoreId": "string",
   "NumberOfBytes": number
}
```

Request Parameters

For information about the parameters that are common to all actions, see Common Parameters (p. 252).

The request accepts the following data in JSON format.

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

CustomKeyStoreId (p. 109)

Generates the random byte string in the AWS CloudHSM cluster that is associated with the specified custom key store. To find the ID of a custom key store, use the DescribeCustomKeyStores (p. 52) operation.

Type: String

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

Required: No

NumberOfBytes (p. 109)

The length of the byte string.

Type: Integer

Valid Range: Minimum value of 1. Maximum value of 1024.

Required: No

Response Syntax

```
{
   "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.

Plaintext (p. 110)

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.

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 Errors (p. 254).

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 (p. 52) operation.

This exception is thrown under the following conditions:

- You requested the CreateKey (p. 26) or GenerateRandom (p. 109) operation in a custom key store that is not connected. These operations are valid only when the custom key store `ConnectionState` is `CONNECTED`.
- You requested the UpdateCustomKeyStore (p. 213) or DeleteCustomKeyStore (p. 46) operation on a custom key store that is not disconnected. This operation is valid only when the custom key store `ConnectionState` is `DISCONNECTED`.
- You requested the ConnectCustomKeyStore (p. 8) operation on a custom key store with a `ConnectionState` of `DISCONNECTING` or `FAILED`. This operation is valid for all other `ConnectionState` values.

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. The request can be retried.

HTTP Status Code: 500

KMSInternalException

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

HTTP Status Code: 500

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
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java V2
- AWS SDK for JavaScript
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

GetKeyPolicy

Gets a key policy attached to the specified KMS key.

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

Required permissions: kms:GetKeyPolicy (key policy)

Related operations: PutKeyPolicy (p. 164)

Request Syntax

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

Request Parameters

For information about the parameters that are common to all actions, see Common Parameters (p. 252).

The request accepts the following data in JSON format.

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

KeyId (p. 112)

Gets the key policy 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 (p. 151) or DescribeKey (p. 56).

Type: String

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

Required: Yes

PolicyName (p. 112)

Specifies the name of the key policy. The only valid name is `default`. To get the names of key policies, use ListKeyPolicies (p. 147).

Type: String

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

Pattern: `[\w]+`

Required: Yes

Response Syntax

```
{
   "Policy": "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 (p. 113)

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]+`

Errors

For information about the errors that are common to all actions, see Common Errors (p. 254).

DependencyTimeoutException

The system timed out while trying to fulfill the request. The request can be retried.

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.

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 .

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
}"}
```

See Also

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

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go

- AWS SDK for Java V2
- AWS SDK for JavaScript
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

GetKeyRotationStatus

Gets a Boolean value that indicates whether automatic rotation of the key material is enabled for the specified KMS key.

When you enable automatic rotation for customer managed KMS keys, 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 monitor rotation of the key material for your KMS keys in AWS CloudTrail and Amazon CloudWatch.

Automatic key rotation is supported only on symmetric encryption KMS keys. 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. The key rotation status of these KMS keys is always `false`. 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 (p. 73)) and disable automatic rotation (DisableKeyRotation (p. 64)) 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`.

> **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 (p. 73).

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 (p. 64)
- EnableKeyRotation (p. 73)

Request Syntax

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

Request Parameters

For information about the parameters that are common to all actions, see Common Parameters (p. 252).

The request accepts the following data in JSON format.

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

KeyId (p. 116)

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 (p. 151) or DescribeKey (p. 56).

Type: String

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

Required: Yes

Response Syntax

```
{
   "KeyRotationEnabled": 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.

KeyRotationEnabled (p. 117)

A Boolean value that specifies whether key rotation is enabled.

Type: Boolean

Errors

For information about the errors that are common to all actions, see Common Errors (p. 254).

DependencyTimeoutException

The system timed out while trying to fulfill the request. The request can be retried.

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.

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 .

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

{"KeyRotationEnabled":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
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java V2
- AWS SDK for JavaScript
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

GetParametersForImport

Returns the items you need to import key material into a symmetric encryption KMS key. For more information about importing key material into AWS KMS, see Importing key material in the *AWS Key Management Service Developer Guide*.

This operation returns a public key and an import token. Use the public key to encrypt the symmetric key material. Store the import token to send with a subsequent ImportKeyMaterial (p. 130) request.

You must specify the key ID of the symmetric encryption KMS key into which you will import key material. This KMS key's `Origin` must be `EXTERNAL`. You must also specify the wrapping algorithm and type of wrapping key (public key) that you will use to encrypt the key material. You cannot perform this operation on an asymmetric KMS key, an HMAC KMS key, or on any KMS key in a different AWS account.

To import key material, you must use the public key and import token from the same response. These items are valid for 24 hours. The expiration date and time appear in the `GetParametersForImport` response. You cannot use an expired token in an ImportKeyMaterial (p. 130) request. If your key and token expire, send another `GetParametersForImport` request.

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 (p. 130)
- DeleteImportedKeyMaterial (p. 49)

Request Syntax

```
{
   "KeyId": "string",
   "WrappingAlgorithm": "string",
   "WrappingKeySpec": "string"
}
```

Request Parameters

For information about the parameters that are common to all actions, see Common Parameters (p. 252).

The request accepts the following data in JSON format.

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

KeyId (p. 120)

The identifier of the symmetric encryption KMS key into which you will import 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 (p. 151) or DescribeKey (p. 56).

Type: String

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

Required: Yes

WrappingAlgorithm (p. 120)

The algorithm you will use to encrypt the key material before importing it with ImportKeyMaterial (p. 130). For more information, see Encrypt the Key Material in the *AWS Key Management Service Developer Guide*.

Type: String

Valid Values: `RSAES_PKCS1_V1_5 | RSAES_OAEP_SHA_1 | RSAES_OAEP_SHA_256`

Required: Yes

WrappingKeySpec (p. 120)

The type of wrapping key (public key) to return in the response. Only 2048-bit RSA public keys are supported.

Type: String

Valid Values: `RSA_2048`

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 (p. 121)

The import token to send in a subsequent ImportKeyMaterial (p. 130) request.

Type: Base64-encoded binary data object

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

KeyId (p. 121)

The Amazon Resource Name (key ARN) of the KMS key to use in a subsequent ImportKeyMaterial (p. 130) 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 (p. 121)

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 (p. 130) request and you must send another `GetParametersForImport` request to get new ones.

Type: Timestamp

PublicKey (p. 121)

The public key to use to encrypt the key material before importing it with ImportKeyMaterial (p. 130).

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 Errors (p. 254).

DependencyTimeoutException

The system timed out while trying to fulfill the request. The request can be retried.

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.

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 .

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: 20161130T225216Z
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_1",
  "KeyId": "1234abcd-12ab-34cd-56ef-1234567890ab",
  "WrappingKeySpec": "RSA_2048"
}
```

Example Response

This example illustrates one usage of GetParametersForImport.

```
HTTP/1.1 200 OK
Server: Server
Date: Wed, 30 Nov 2016 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//
daUB9ZeCKoiJxmiw09YQAABrQwggawBgkqhkiG9w0BBwagggahMIIGnQIBADCCBpYGCSqGSIb3DQEHATAeBglghkgBZQMEAS4wEQQMv
U4Wg2Vw+RMAgEQgIIGZ/wOYGszlrjopP6BW63jlYYn
+gd7jpdpx0dxPmPC5Ka6uuUomx1yMKVdgtMiX85jHr8or7RoLISwsyQH+CRD33V
+pQs+Rm0+XkinHj5Zl371ibHytqM1DwhCs5FdQJM+8kLau7EXTcar7XLQj86DWJRj/
dQW0nDdkQXgXvz7GFWkbYs3IELvTAc5lHOLHgkXeoXom3NtHMvbR2V34tYwaT86gdira9Qj0FDouNaTesEOJN/
QjBedXcnuWumwOzK+w/OL+MD4tR8/
B1jDjeafRv7YSMxiADr2FsfDL0ELhgXhFVC0Wz42oM0jYnoYjZuXx6fQxEmADjBMPjk6W
+SFs4sWOuHs0U8npsWBNOnLAZPqXskqSuPZzb3XMG59s+2ZUcbeARQjYv97861ohWgwzjxur2+wSlaGNYAb
+Xh7EV34n2KSLuJ1lSrZrEWlU1Pato6zzN1x0VHJgU3sMCJMQz1uch8ZGHbI7vvBvvvqTJT/
+087IA8thTTCRLAYTjr81sSEofug71twBrhct3pzKswaNQVmWMptBe54HWiWWZz1peNuIAIJtX9qtNzeuYEJyqfVBera0B5tK1vCOrw
+E4AQcSin0AWERUK9LY3BNM2svFrrl2tPWURtUPokMVI0i4NLw2fsHtLw1CXqwjGuzEGKvRfiaat3WGzAtMao5sSFQz/
XSCB9Ab5OsddOTArBr/ShuX1WYuPIL2+zQP+gadWjAfTgmx9Q4K2MxQUpS72bqUJmfzXqpVi63sKL43tOwJ
+2Bt8Z5JA9xaPkPwiYE5q7dWL4J57cr+Ty/GLXAhat9xIUsTjG5E3FIHLywKiBwlVjH/T5FXxk
+T0TXV/61UPGaxPX2HkFTirq/D2Uhz45pFwwH46nbhJe9NoRodjot+uAblfuAqxz0YElCRt/
```

```
gIMr87l4AF7X48JHfvqmZAYGdhJ1bUhSw8VfTOPkHpUV2k6Eq9DvcSRDsww1FI5+fVf0ZpDEf0W2itRz5Hq
+cRkQL9EZqLICNF0QrhEuEJNBXf3oSckvS1tqPnHaRIRmG71BONqwc7fSU7zmXa
+O95GV3gIgfvnQ3HJy5EHR2dgkjQdP
+hfdw7BcC9NT7ZyO9XefAI5GEr623hrzn6yom4JIiyUjjCQPK8mS75rIgazvyTp0WQKpSSKeJOZswYLNgip8Xv/
UBcehAKwRL0QhbOGhUbZvoRNS8c1FbrCUlcBc4W4aWzA4e7cepqy38/jfwRoh0UvN/
bbaDh8FC+jZyXhyXSTIPvM25HVVrxsDbsN8LkCabokXFlkhiawm3PqVm6QgWWKcpR2Td+ty
+Bdl2tRmGHDsPcHN0WaUEq2AjE7kzL0dv7Jd9OemBNTZSlEoQ8U5+sKbvmSrtFvPIj7zWDpDT9bkZFHcCvwlIE6AflbgBS8z0+xllVg
phBgaiRlDQdDmJmGD1yl+dxnIcoPs14xlcIwBdpw/M
+lvUuX8K4tqLMKzi1MOE0heBhGL0uEebYSkSQSUXUTTCk9hEkqslw0VXgwpgnGBXAOnVtYdUaqFMx5RIVxW471bnU0CYW5MrTTJ7o2j
H4KrdRPdvevc8kTG6I8fdK/ArYCvTk/yYL3L6YZbeqbActUTADX0iBijX/T5QYz/
Dd4H1eX4abHV70CnxftxCHuLMnwR8DpJVnkouQAqb4N7Ap6JIYkvNKFWb8HBlygq5kKcg5dTMAMiPRz80qsQm/
IwGG9JVbKeyhqlKtQOIerspm8J99lcn5s0aB180LKrtXAaFD1AyO3nDZxB3I71QKvOulr1BZ6K4meBKkEw3VqW4PpmxmBKnQVUK1jqw
+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":
 "MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAvH3Yj0wbkLEpUl95Cv1cJVjsVNSjwGq3tCLnzXfhVwVvmzGN8pYj3U8nK
+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
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java V2
- AWS SDK for JavaScript
- AWS SDK for PHP V3
- AWS SDK for Python
- 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 (p. 77), ReEncrypt (p. 170), or Verify (p. 224) 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. For details, see Special Considerations for Downloading Public Keys.

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 or signing.
- EncryptionAlgorithms or SigningAlgorithms: A list of the encryption algorithms or the 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.

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 (p. 26)

Request Syntax

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

Request Parameters

For information about the parameters that are common to all actions, see Common Parameters (p. 252).

The request accepts the following data in JSON format.

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

KeyId (p. 125)

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 (p. 151) or DescribeKey (p. 56). To get the alias name and alias ARN, use ListAliases (p. 136).

Type: String

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

Required: Yes

GrantTokens (p. 125)

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",
   "EncryptionAlgorithms": [ "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 (p. 126)

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 will support 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`

EncryptionAlgorithms (p. 126)

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`

KeyId (p. 126)

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 (p. 126)

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`

KeyUsage (p. 126)

The permitted use of the public key. Valid values are `ENCRYPT_DECRYPT` or `SIGN_VERIFY`.

This information is critical. 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`

PublicKey (p. 126)

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 (p. 126)

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`

Errors

For information about the errors that are common to all actions, see Common Errors (p. 254).

DependencyTimeoutException

The system timed out while trying to fulfill the request. The request can be retried.

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`. To find the `KeyUsage` of a KMS key, use the DescribeKey (p. 56) operation.

To find the encryption or signing algorithms supported for a particular KMS key, use the DescribeKey (p. 56) 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.

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 .

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
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java V2
- AWS SDK for JavaScript
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

ImportKeyMaterial

Imports key material into an existing symmetric encryption KMS key that was created without key material. After you successfully import key material into a KMS key, you can reimport the same key material into that KMS key, but you cannot import different key material.

You cannot perform this operation on an asymmetric KMS key, an HMAC KMS key, or on any KMS key in a different AWS account. For more information about creating KMS keys with no key material and then importing key material, see Importing Key Material in the *AWS Key Management Service Developer Guide*.

Before using this operation, call GetParametersForImport (p. 120). Its response includes a public key and an import token. Use the public key to encrypt the key material. Then, submit the import token from the same `GetParametersForImport` response.

When calling this operation, you must specify the following values:

- The key ID or key ARN of a KMS key with no key material. Its `Origin` must be `EXTERNAL`.

  To create a KMS key with no key material, call CreateKey (p. 26) and set the value of its `Origin` parameter to `EXTERNAL`. To get the `Origin` of a KMS key, call DescribeKey (p. 56).)
- The encrypted key material. To get the public key to encrypt the key material, call GetParametersForImport (p. 120).
- The import token that GetParametersForImport (p. 120) returned. You must use a public key and token from the same `GetParametersForImport` response.
- Whether the key material expires and if so, when. If you set an expiration date, AWS KMS deletes the key material from the KMS key on the specified date, and the KMS key becomes unusable. To use the KMS key again, you must reimport the same key material. The only way to change an expiration date is by reimporting the same key material and specifying a new expiration date.

When this operation is successful, the key state of the KMS key changes from `PendingImport` to `Enabled`, and you can use the 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 (p. 120) to get a new public key and import token for the KMS key and repeat the import procedure. For help, see How To Import 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 (p. 49)
- GetParametersForImport (p. 120)

Request Syntax

```
{
   "EncryptedKeyMaterial": blob,
   "ExpirationModel": "string",
   "ImportToken": blob,
```

```
   "KeyId": "string",
   "ValidTo": number
}
```

Request Parameters

For information about the parameters that are common to all actions, see Common Parameters (p. 252).

The request accepts the following data in JSON format.

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

EncryptedKeyMaterial (p. 130)

The encrypted key material to import. The key material must be encrypted with the public wrapping key that GetParametersForImport (p. 120) 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 (p. 130)

The import token that you received in the response to a previous GetParametersForImport (p. 120) 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 (p. 130)

The identifier of the symmetric encryption KMS key that receives the imported key material. This must be the same KMS key specified in the `KeyID` parameter of the corresponding GetParametersForImport (p. 120) request. The `Origin` of the KMS key must be `EXTERNAL`. You cannot perform this operation on an asymmetric KMS key, an HMAC KMS key, 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 (p. 151) or DescribeKey (p. 56).

Type: String

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

Required: Yes

ExpirationModel (p. 130)

Specifies whether the key material expires. The default is `KEY_MATERIAL_EXPIRES`, in which case you must include the `ValidTo` parameter. When this parameter is set to `KEY_MATERIAL_DOES_NOT_EXPIRE`, you must omit the `ValidTo` parameter.

Type: String

Valid Values: `KEY_MATERIAL_EXPIRES | KEY_MATERIAL_DOES_NOT_EXPIRE`

Required: No

ValidTo (p. 130)

The time at which the imported key material expires. When the key material expires, AWS KMS deletes the key material and the KMS key becomes unusable. You must omit this parameter when the `ExpirationModel` parameter is set to `KEY_MATERIAL_DOES_NOT_EXPIRE`. Otherwise it is required.

Type: Timestamp

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 Errors (p. 254).

DependencyTimeoutException

The system timed out while trying to fulfill the request. The request can be retried.

HTTP Status Code: 500

ExpiredImportTokenException

The request was rejected because the specified import token is expired. Use GetParametersForImport (p. 120) 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 is not the same key material that was previously imported into this 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 (p. 37) or ReEncrypt (p. 170) 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 (p. 130) operation, the request was rejected because AWS KMS could not decrypt the encrypted (wrapped) key material.

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.

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 .

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//
daUB9ZeCKoiJxmiw09YQAABrQwggawBgkqhkiG9w0BBwagggahMIIGnQIBADCCBpYGCSqGSIb3DQEHATAeBglghkgBZQMEAS4wEQQMv
U4Wg2Vw+RMAgEQgIIGZ/wOYGszlrjopP6BW63jlYYn
```

```
+gd7jpdpx0dxPmPC5Ka6uuUomx1yMKVdgtMiX85jHr8or7RoLISwsyQH+CRD33V
+pQs+Rm0+XkinHj5Zl371ibHytqM1DwhCs5FdQJM+8kLau7EXTcar7XLQj86DWJRj/
dQW0nDdkQXgXvz7GFWkbYs3IELvTAc5lHOLHgkXeoXom3NtHMvbR2V34tYwaT86gdira9Qj0FDouNaTesEOJN/
QjBedXcnuWumwOzK+w/OL+MD4tR8/
B1jDjeafRv7YSMxiADr2FsfDL0ELhgXhFVC0Wz42oM0jYnoYjZuXx6fQxEmADjBMPjk6W
+SFs4sWOuHs0U8npsWBNOnLAZPqXskqSuPZzb3XMG59s+2ZUcbeARQjYv97861ohWgwzjxur2+wSlaGNYAb
+Xh7EV34n2KSLuJ1lSrZrEWlU1Pato6zzN1x0VHJgU3sMCJMQz1uch8ZGHbI7vvBvvvqTJT/
+087IA8thTTCRLAYTjr81sSEofug71twBrhct3pzKswaNQVmWMptBe54HWiWWZz1peNuIAIJtX9qtNzeuYEJyqfVBera0B5tK1vCOrw
+E4AQcSin0AWERUK9LY3BNM2svFrrl2tPWURtUPokMVI0i4NLw2fsHtLw1CXqwjGuzEGKvRfiaat3WGzAtMao5sSFQz/
XSCB9Ab5OsddOTArBr/ShuX1WYuPIL2+zQP+gadWjAfTgmx9Q4K2MxQUpS72bqUJmfzXqpVi63sKL43tOwJ
+2Bt8Z5JA9xaPkPwiYE5q7dWL4J57cr+Ty/GLXAhat9xIUsTjG5E3FIHLywKiBwlVjH/T5FXxk
+T0TXV/61UPGaxPX2HkFTirq/D2Uhz45pFwwH46nbhJe9NoRodjot+uAblfuAqxz0YElCRt/
gIMr87l4AF7X48JHfvqmZAYGdhJ1bUhSw8VfTOPkHpUV2k6Eq9DvcSRDsww1FI5+fVf0ZpDEf0W2itRz5Hq
+cRkQL9EZqLICNF0QrhEuEJNBXf3oSckvS1tqPnHaRIRmG71BONqwc7fSU7zmXa
+O95GV3gIgfvnQ3HJy5EHR2dgkjQdP
+hfdw7BcC9NT7ZyO9XefAI5GEr623hrzn6yom4JIiyUjjCQPK8mS75rIgazvyTp0WQKpSSKeJOZswYLNgip8Xv/
UBcehAKwRL0QhbOGhUbZvoRNS8c1FbrCUlcBc4W4aWzA4e7cepqy38/jfwRoh0UvN/
bbaDh8FC+jZyXhyXSTIPvM25HVVrxsDbsN8LkCabokXFlkhiawm3PqVm6QgWWKcpR2Td+ty
+Bdl2tRmGHDsPcHN0WaUEq2AjE7kzL0dv7Jd9OemBNTZSlEoQ8U5+sKbvmSrtFvPIj7zWDpDT9bkZFHcCvwlIE6AflbgBS8z0+xllVg
phBgaiRlDQdDmJmGD1yl+dxnIcoPs14xlcIwBdpw/M
+lvUuX8K4tqLMKzi1MOE0heBhGL0uEebYSkSQSUXUTTCk9hEkqslw0VXgwpgnGBXAOnVtYdUaqFMx5RIVxW471bnU0CYW5MrTTJ7o2j
H4KrdRPdvevc8kTG6I8fdK/ArYCvTk/yYL3L6YZbeqbActUTADX0iBijX/T5QYz/
Dd4H1eX4abHV70CnxftxCHuLMnwR8DpJVnkouQAqb4N7Ap6JIYkvNKFWb8HBlygq5kKcg5dTMAMiPRz80qsQm/
IwGG9JVbKeyhqlKtQOIerspm8J99lcn5s0aB180LKrtXAaFD1AyO3nDZxB3I71QKvOulr1BZ6K4meBKkEw3VqW4PpmxmBKnQVUK1jqw
+2ytZAdDox9zLT7YW457esjUQC6zibfBwb8G97leh704m37Stq6Z752u46frBNSPQlypGuSbqCw1peKeqf/
AVehk+j8RKBegOQSCvEja4KPmQrayXVzu3h1tDktA1/Wj21ercJaW20fcZ1KQG/
GPHuScFgBsWawQf1spqKwZyHAHPaWZCymD9Fo2yHBHi+/ARPwM02iuqDLi9Tqv/g0=",
  "EncryptedKeyMaterial": "CubeyZ4cm/xMEA0UG5jP1iBzh/0E+uUg407JDcXhIC+iuMm
+wPgITaEby+Y3nM/e6gjUls5vy9TdBRFv4+JtksvB5hW4Znb2lUQhTUv+SSAZpaI14kAgTq/
jC2GTLkaC6Vf5zJx2xaLrOKGV2Xu4YgONIGslubHNffTC3aL/YBJ/FXTXaVu7rS2phOFCrZ
+ATittS03w4DiCVoNwo2v0QE0+dVoUNjXNQC1veWxhPlC7FezfK7AIsBSSXotJfANxRkybg8KcmkSoYdzr3N0L0v7oMorgbTgaTvdrI
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

{}
```

See Also

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

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

- AWS SDK for Python
- 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 (p. 11).

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 (p. 11)
- DeleteAlias (p. 43)
- UpdateAlias (p. 209)

Request Syntax

```
{
   "KeyId": "string",
   "Limit": number,
   "Marker": "string"
}
```

Request Parameters

For information about the parameters that are common to all actions, see Common Parameters (p. 252).

The request accepts the following data in JSON format.

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

KeyId (p. 136)

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 (p. 151) or DescribeKey (p. 56).

Type: String

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

Required: No

Limit (p. 136)

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 (p. 136)

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 (p. 137)

A list of aliases.

Type: Array of AliasListEntry (p. 234) objects

NextMarker (p. 137)

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 (p. 137)

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 thisresponse to the `Marker` parameter in a subsequent request.

Type: Boolean

Errors

For information about the errors that are common to all actions, see Common Errors (p. 254).

DependencyTimeoutException

The system timed out while trying to fulfill the request. The request can be retried.

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
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java V2
- AWS SDK for JavaScript
- AWS SDK for PHP V3
- AWS SDK for Python
- 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 or grantee 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 working with grants in several programming languages, see Programming grants.

> **Note**
> The `GranteePrincipal` field in the `ListGrants` response usually contains the user or role designated as the grantee principal in the grant. However, when the grantee principal in the grant is an AWS service, the `GranteePrincipal` field contains the service principal, which might represent several different grantee principals.

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 (p. 19)
- ListRetirableGrants (p. 159)
- RetireGrant (p. 185)
- RevokeGrant (p. 189)

Request Syntax

```
{
   "GranteePrincipal": "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 (p. 252).

The request accepts the following data in JSON format.

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

KeyId (p. 141)

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 (p. 151) or DescribeKey (p. 56).

Type: String

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

Required: Yes

GranteePrincipal (p. 141)

Returns only grants where the specified principal is the grantee principal for the grant.

Type: String

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

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

Required: No

GrantId (p. 141)

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 (p. 141)

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 (p. 141)

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"
            }
         },
         "CreationDate": number,
         "GranteePrincipal": "string",
         "GrantId": "string",
         "IssuingAccount": "string",
         "KeyId": "string",
         "Name": "string",
         "Operations": [ "string" ],
         "RetiringPrincipal": "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 (p. 142)

A list of grants.

Type: Array of GrantListEntry (p. 241) objects

NextMarker (p. 142)

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 (p. 142)

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 thisresponse to the `Marker` parameter in a subsequent request.

Type: Boolean

Errors

For information about the errors that are common to all actions, see Common Errors (p. 254).

DependencyTimeoutException

The system timed out while trying to fulfill the request. The request can be retried.

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.

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 .

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
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java V2
- AWS SDK for JavaScript
- AWS SDK for PHP V3
- AWS SDK for Python
- 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 (p. 112) operation. However, the only valid policy name is `default`.

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 (p. 112)
- PutKeyPolicy (p. 164)

Request Syntax

```
{
   "KeyId": "string",
   "Limit": number,
   "Marker": "string"
}
```

Request Parameters

For information about the parameters that are common to all actions, see Common Parameters (p. 252).

The request accepts the following data in JSON format.

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

KeyId (p. 147)

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 (p. 151) or DescribeKey (p. 56).

Type: String

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

Required: Yes

Limit (p. 147)

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.

Only one policy can be attached to a key.

Type: Integer

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

Required: No

Marker (p. 147)

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 (p. 148)

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 (p. 148)

A list of key policy names. The only valid value is `default`.

Type: Array of strings

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

Pattern: `[\w]+`

Truncated (p. 148)

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 thisresponse to the `Marker` parameter in a subsequent request.

Type: Boolean

Errors

For information about the errors that are common to all actions, see Common Errors (p. 254).

DependencyTimeoutException

The system timed out while trying to fulfill the request. The request can be retried.

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.

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 .

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
}
```

See Also

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

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java V2
- AWS SDK for JavaScript
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

ListKeys

Gets a list of all KMS keys in the caller's AWS account and Region.

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

Required permissions: kms:ListKeys (IAM policy)

Related operations:

- CreateKey (p. 26)
- DescribeKey (p. 56)
- ListAliases (p. 136)
- ListResourceTags (p. 155)

Request Syntax

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

Request Parameters

For information about the parameters that are common to all actions, see Common Parameters (p. 252).

The request accepts the following data in JSON format.

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

Limit (p. 151)

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 (p. 151)

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

```
{
   "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 (p. 152)

A list of KMS keys.

Type: Array of KeyListEntry (p. 243) objects

NextMarker (p. 152)

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 (p. 152)

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 thisresponse to the `Marker` parameter in a subsequent request.

Type: Boolean

Errors

For information about the errors that are common to all actions, see Common Errors (p. 254).

DependencyTimeoutException

The system timed out while trying to fulfill the request. The request can be retried.

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
}
```

See Also

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

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java V2
- AWS SDK for JavaScript
- AWS SDK for PHP V3
- AWS SDK for Python
- 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 Tagging keys.

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 (p. 26)
- ReplicateKey (p. 178)
- TagResource (p. 202)
- UntagResource (p. 206)

Request Syntax

```
{
   "KeyId": "string",
   "Limit": number,
   "Marker": "string"
}
```

Request Parameters

For information about the parameters that are common to all actions, see Common Parameters (p. 252).

The request accepts the following data in JSON format.

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

KeyId (p. 155)

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 (p. 151) or DescribeKey (p. 56).

Type: String

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

Required: Yes

Limit (p. 155)

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 (p. 155)

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 (p. 156)

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 (p. 156)

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 in AWS KMS in the *AWS Key Management Service Developer Guide*.

Type: Array of Tag (p. 251) objects

Truncated (p. 156)

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 thisresponse to the `Marker` parameter in a subsequent request.

Type: Boolean

Errors

For information about the errors that are common to all actions, see Common Errors (p. 254).

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
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java V2
- AWS SDK for JavaScript
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

ListRetirableGrants

Returns information about all grants in the AWS account and Region that have the specified retiring 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 (p. 185) 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 working with grants in several programming languages, see Programming grants.

Cross-account use: You must specify a principal in your AWS account. However, this operation can return grants in any AWS account. 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.

Related operations:

- CreateGrant (p. 19)
- ListGrants (p. 141)
- RetireGrant (p. 185)
- RevokeGrant (p. 189)

Request Syntax

```
{
   "Limit": number,
   "Marker": "string",
   "RetiringPrincipal": "string"
}
```

Request Parameters

For information about the parameters that are common to all actions, see Common Parameters (p. 252).

The request accepts the following data in JSON format.

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

RetiringPrincipal (p. 159)

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 AWS principals include AWS accounts (root), IAM users, federated users, and assumed role users. For examples of the ARN syntax for specifying a principal, see AWS Identity and Access Management (IAM) in the Example ARNs section of the *Amazon Web Services General Reference*.

Type: String

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

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

Required: Yes

Limit (p. 159)

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 (p. 159)

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"
            }
         },
         "CreationDate": number,
         "GranteePrincipal": "string",
         "GrantId": "string",
         "IssuingAccount": "string",
         "KeyId": "string",
         "Name": "string",
         "Operations": [ "string" ],
         "RetiringPrincipal": "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 (p. 160)

A list of grants.

Type: Array of GrantListEntry (p. 241) objects

NextMarker (p. 160)

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 (p. 160)

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 thisresponse to the `Marker` parameter in a subsequent request.

Type: Boolean

Errors

For information about the errors that are common to all actions, see Common Errors (p. 254).

DependencyTimeoutException

The system timed out while trying to fulfill the request. The request can be retried.

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
- AWS SDK for .NET

- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java V2
- AWS SDK for JavaScript
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

PutKeyPolicy

Attaches a key policy to the specified KMS key.

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 Setting a key policy 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)

Related operations: GetKeyPolicy (p. 112)

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 (p. 252).

The request accepts the following data in JSON format.

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

KeyId (p. 164)

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 (p. 151) or DescribeKey (p. 56).

Type: String

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

Required: Yes

Policy (p. 164)

The key policy to attach to the KMS key.

The key policy must meet the following criteria:

- If you don't set `BypassPolicyLockoutSafetyCheck` to true, the key policy must allow the principal that is making the `PutKeyPolicy` request to make a subsequent `PutKeyPolicy` request on the KMS key. This reduces the risk that the KMS key becomes unmanageable. For more information, refer to the scenario in the Default Key Policy section of the *AWS Key Management Service Developer Guide*.
- 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 (for example, an IAM user or role), 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: Yes

PolicyName (p. 164)

The name of the key policy. The only valid value is `default`.

Type: String

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

Pattern: `[\w]+`

Required: Yes

BypassPolicyLockoutSafetyCheck (p. 164)

A flag to indicate whether to bypass the key policy lockout safety check.

> **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, refer to the scenario in the Default Key Policy section 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 default value is false.

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 Errors (p. 254).

DependencyTimeoutException

The system timed out while trying to fulfill the request. The request can be retried.

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.

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 .

HTTP Status Code: 400

LimitExceededException

The request was rejected because a 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
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java V2
- AWS SDK for JavaScript
- AWS SDK for PHP V3
- AWS SDK for Python
- 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 (p. 77) or GenerateDataKey (p. 83). 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 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*.

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, you must use its key ARN or alias ARN.

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 (p. 164) operation to set a key policy.

Related operations:

- Decrypt (p. 37)
- Encrypt (p. 77)
- GenerateDataKey (p. 83)
- GenerateDataKeyPair (p. 89)

Request Syntax

```
{
   "CiphertextBlob": blob,
   "DestinationEncryptionAlgorithm": "string",
   "DestinationEncryptionContext": {
      "string" : "string"
   },
   "DestinationKeyId": "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 (p. 252).

The request accepts the following data in JSON format.

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

CiphertextBlob (p. 171)

Ciphertext of the data to reencrypt.

Type: Base64-encoded binary data object

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

Required: Yes

DestinationKeyId (p. 171)

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 (p. 56) 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 (p. 151) or DescribeKey (p. 56). To get the alias name and alias ARN, use ListAliases (p. 136).

Type: String

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

Required: Yes

DestinationEncryptionAlgorithm (p. 171)

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`

Required: No

DestinationEncryptionContext (p. 171)

Specifies that encryption context to use when the reencrypting the data.

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

GrantTokens (p. 171)

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 (p. 171)

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`

Required: No

SourceEncryptionContext (p. 171)

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 (p. 171)

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. 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 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 (p. 151) or DescribeKey (p. 56). To get the alias name and alias ARN, use ListAliases (p. 136).

Type: String

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

Required: No

Response Syntax

```
{
   "CiphertextBlob": blob,
   "DestinationEncryptionAlgorithm": "string",
   "KeyId": "string",
   "SourceEncryptionAlgorithm": "string",
   "SourceKeyId": "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 (p. 174)

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 (p. 174)

The encryption algorithm that was used to reencrypt the data.

Type: String

Valid Values: `SYMMETRIC_DEFAULT | RSAES_OAEP_SHA_1 | RSAES_OAEP_SHA_256`

KeyId (p. 174)

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 (p. 174)

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`

SourceKeyId (p. 174)

Unique identifier of the KMS key used to originally encrypt the data.

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 Errors (p. 254).

DependencyTimeoutException

The system timed out while trying to fulfill the request. The request can be retried.

HTTP Status Code: 500

DisabledException

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

HTTP Status Code: 400

IncorrectKeyException

The request was rejected because the specified KMS key cannot decrypt the data. The `KeyId` in a Decrypt (p. 37) request and the `SourceKeyId` in a ReEncrypt (p. 170) request must identify the same KMS key that was used to encrypt the ciphertext.

HTTP Status Code: 400

InvalidCiphertextException

From the Decrypt (p. 37) or ReEncrypt (p. 170) 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 (p. 130) operation, the request was rejected because AWS KMS could not decrypt the encrypted (wrapped) key material.

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`. To find the `KeyUsage` of a KMS key, use the DescribeKey (p. 56) operation.

To find the encryption or signing algorithms supported for a particular KMS key, use the DescribeKey (p. 56) 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.

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 .

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
+K5DVkMfunTHr0w6V6crnuAGw80uRwAAAH0wewYJKoZIhvcNAQcGoG4wbAIBADBnBgkqhkiG9w0BBwEwHgYJYIZIAWUDBAEuMBEEDPX
+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":
 "AQECAHjRYf5WytIc0C857tFSnBaPn2F8DgfmThbJlGfR8P3WlwAAAH0wewYJKoZIhvcNAQcGoG4wbAIBADBnBgkqhkiG9w0BBwEwH
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",
  "DestinationEncryptionAlgorithm": "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
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java V2
- AWS SDK for JavaScript
- AWS SDK for PHP V3
- AWS SDK for Python
- 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 (p. 26) 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 states of AWS KMS keys. 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 (p. 5)) 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 (p. 26) 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. For details, see Importing key material into multi-Region keys in the *AWS Key Management Service Developer Guide*.

To convert a replica key to a primary key, use the UpdatePrimaryRegion (p. 221) 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 (p. 26)
- UpdatePrimaryRegion (p. 221)

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 (p. 252).

The request accepts the following data in JSON format.

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

KeyId (p. 179)

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 (p. 56) 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 (p. 151) or DescribeKey (p. 56).

Type: String

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

Required: Yes

ReplicaRegion (p. 179)

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*.

Note
HMAC KMS keys are not supported in all AWS Regions. If you try to replicate an HMAC KMS key in an AWS Region in which HMAC keys are not supported, the `ReplicateKey` operation returns an `UnsupportedOperationException`. For a list of Regions in which HMAC KMS keys are supported, see HMAC keys in AWS KMS in the *AWS Key Management Service Developer Guide*.

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 (p. 179)

A flag to indicate whether to bypass the key policy lockout safety check.

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, refer to the scenario in the Default Key Policy section 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 default value is false.

Type: Boolean

Required: No

Description (p. 179)

A description of the KMS key. The default value is an empty string (no description).

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 (p. 179)

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:

- If you don't set `BypassPolicyLockoutSafetyCheck` to true, the key policy must give the caller `kms:PutKeyPolicy` permission on the replica key. This reduces the risk that the KMS key becomes unmanageable. For more information, refer to the scenario in the Default Key Policy section of the AWS Key Management Service Developer Guide .
- 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 (for example, an IAM user or role), 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 (p. 179)

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 (p. 202) operation.

> **Note**
> Tagging or untagging a KMS key can allow or deny permission to the KMS key. For details, see ABAC in 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 Tagging Keys.

Type: Array of Tag (p. 251) objects

Required: No

Response Syntax

```
{
```

```
   "ReplicaKeyMetadata": {
      "Arn": "string",
      "AWSAccountId": "string",
      "CloudHsmClusterId": "string",
      "CreationDate": number,
      "CustomerMasterKeySpec": "string",
      "CustomKeyStoreId": "string",
      "DeletionDate": number,
      "Description": "string",
      "Enabled": boolean,
      "EncryptionAlgorithms": [ "string" ],
      "ExpirationModel": "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
   },
   "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 (p. 181)

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 (p. 244) object

ReplicaPolicy (p. 181)

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 (p. 181)

The tags on the new replica key. The value is a list of tag key and tag value pairs.

Type: Array of Tag (p. 251) objects

Errors

For information about the errors that are common to all actions, see Common Errors (p. 254).

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.

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 .

HTTP Status Code: 400

LimitExceededException

The request was rejected because a 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
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java V2
- AWS SDK for JavaScript
- AWS SDK for PHP V3
- AWS SDK for Python
- 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 (p. 19) 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 details, see Retiring and revoking grants 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 working with grants in several programming languages, see Programming grants.

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 (p. 19)
- ListGrants (p. 141)
- ListRetirableGrants (p. 159)
- RevokeGrant (p. 189)

Request Syntax

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

Request Parameters

For information about the parameters that are common to all actions, see Common Parameters (p. 252).

The request accepts the following data in JSON format.

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

GrantId (p. 185)

Identifies the grant to retire. To get the grant ID, use CreateGrant (p. 19), ListGrants (p. 141), or ListRetirableGrants (p. 159).

- Grant ID Example - 0123456789012345678901234567890123456789012345678901234567890123

Type: String

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

Required: No

GrantToken (p. 185)

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 (p. 19) 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 (p. 185)

The key ARN KMS key associated with the grant. To find the key ARN, use the ListKeys (p. 151) 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 Errors (p. 254).

DependencyTimeoutException

The system timed out while trying to fulfill the request. The request can be retried.

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

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.

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 .

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
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java V2
- AWS SDK for JavaScript
- AWS SDK for PHP V3
- AWS SDK for Python
- 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 working with grants in several programming languages, see Programming grants.

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 (p. 19)
- ListGrants (p. 141)
- ListRetirableGrants (p. 159)
- RetireGrant (p. 185)

Request Syntax

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

Request Parameters

For information about the parameters that are common to all actions, see Common Parameters (p. 252).

The request accepts the following data in JSON format.

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

GrantId (p. 189)

Identifies the grant to revoke. To get the grant ID, use CreateGrant (p. 19), ListGrants (p. 141), or ListRetirableGrants (p. 159).

Type: String

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

Required: Yes

KeyId (p. 189)

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 (p. 151) or DescribeKey (p. 56).

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 (p. 151) or DescribeKey (p. 56).

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 Errors (p. 254).

DependencyTimeoutException

The system timed out while trying to fulfill the request. The request can be retried.

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

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.

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 .

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
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java V2
- AWS SDK for JavaScript
- AWS SDK for PHP V3
- AWS SDK for Python
- 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 (p. 5) 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.) To prevent the use of a KMS key without deleting it, use DisableKey (p. 61).

If you schedule deletion of a KMS key from a custom key store, when the waiting period expires, `ScheduleKeyDeletion` deletes the KMS key from AWS KMS. Then AWS KMS makes a best effort to delete the 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.

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*.

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 (p. 5)
- DisableKey (p. 61)

Request Syntax

```
{
   "KeyId": "string",
   "PendingWindowInDays": number
}
```

Request Parameters

For information about the parameters that are common to all actions, see Common Parameters (p. 252).

The request accepts the following data in JSON format.

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

KeyId (p. 192)

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 (p. 151) or DescribeKey (p. 56).

Type: String

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

Required: Yes

PendingWindowInDays (p. 192)

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.

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 (p. 193)

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 (p. 193)

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 (p. 193)

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 (p. 193)

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 Errors (p. 254).

DependencyTimeoutException

The system timed out while trying to fulfill the request. The request can be retried.

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.

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 .

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
- AWS SDK for .NET
- AWS SDK for C++

- AWS SDK for Go
- AWS SDK for Java V2
- AWS SDK for JavaScript
- AWS SDK for PHP V3
- AWS SDK for Python
- 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 (p. 224) 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 or ECC 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 (p. 56) 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 or a digest, 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 (p. 224) operation. Or use the GetPublicKey (p. 125) 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 (p. 224)

Request Syntax

```
{
   "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 (p. 252).

The request accepts the following data in JSON format.

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

KeyId (p. 197)

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 (p. 56) 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 (p. 151) or DescribeKey (p. 56). To get the alias name and alias ARN, use ListAliases (p. 136).

Type: String

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

Required: Yes

Message (p. 197)

Specifies the message or message digest to sign. Messages can be 0-4096 bytes. To sign a larger message, provide the message digest.

If you provide a message, AWS KMS generates a hash digest of the message and then signs it.

Type: Base64-encoded binary data object

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

Required: Yes

SigningAlgorithm (p. 197)

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.

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`

Required: Yes

GrantTokens (p. 197)

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 (p. 197)

Tells AWS KMS whether the value of the `Message` parameter is a message or message digest. The default value, RAW, indicates a message. To indicate a message digest, enter `DIGEST`.

Type: String

Valid Values: `RAW | DIGEST`

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 (p. 199)

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 (p. 199)

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 ANS 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 (p. 199)

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`

Errors

For information about the errors that are common to all actions, see Common Errors (p. 254).

DependencyTimeoutException

The system timed out while trying to fulfill the request. The request can be retried.

HTTP Status Code: 500

DisabledException

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

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`. To find the `KeyUsage` of a KMS key, use the DescribeKey (p. 56) operation.

To find the encryption or signing algorithms supported for a particular KMS key, use the DescribeKey (p. 56) 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.

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 .

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
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java V2
- AWS SDK for JavaScript
- AWS SDK for PHP V3
- AWS SDK for Python
- 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 in 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 (p. 26)) or replicating it (ReplicateKey (p. 178)).

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 (p. 26)
- ListResourceTags (p. 155)
- ReplicateKey (p. 178)
- UntagResource (p. 206)

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 (p. 252).

The request accepts the following data in JSON format.

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

KeyId (p. 202)

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 (p. 151) or DescribeKey (p. 56).

Type: String

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

Required: Yes

Tags (p. 202)

One or more tags.

Each tag consists of a tag key and a tag value. 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.

Type: Array of Tag (p. 251) 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 Errors (p. 254).

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.

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 .

HTTP Status Code: 400

LimitExceededException

The request was rejected because a 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
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java V2
- AWS SDK for JavaScript
- AWS SDK for PHP V3
- AWS SDK for Python
- 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 in 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 (p. 155) 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 (p. 26)
- ListResourceTags (p. 155)
- ReplicateKey (p. 178)
- TagResource (p. 202)

Request Syntax

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

Request Parameters

For information about the parameters that are common to all actions, see Common Parameters (p. 252).

The request accepts the following data in JSON format.

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

KeyId (p. 206)

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 (p. 151) or DescribeKey (p. 56).

Type: String

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

Required: Yes

TagKeys (p. 206)

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 Errors (p. 254).

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.

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 .

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
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java V2
- AWS SDK for JavaScript
- AWS SDK for PHP V3
- AWS SDK for Python
- 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 in 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), and they must have the same key usage (`ENCRYPT_DECRYPT` or `SIGN_VERIFY`). This restriction prevents errors in code that uses aliases. If you must assign an alias to a different type of KMS key, use DeleteAlias (p. 43) to delete the old alias and CreateAlias (p. 11) to create a new alias.

You cannot use `UpdateAlias` to change an alias name. To change an alias name, use DeleteAlias (p. 43) to delete the old alias and CreateAlias (p. 11) 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 (p. 56) operation. To get the aliases of all KMS keys in the account, use the ListAliases (p. 136) 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 (p. 11)
- DeleteAlias (p. 43)
- ListAliases (p. 136)

Request Syntax

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

Request Parameters

For information about the parameters that are common to all actions, see Common Parameters (p. 252).

The request accepts the following data in JSON format.

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

AliasName (p. 209)

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.

Type: String

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

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

Required: Yes

TargetKeyId (p. 209)

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) 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 (p. 151) or DescribeKey (p. 56).

To verify that the alias is mapped to the correct KMS key, use ListAliases (p. 136).

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 Errors (p. 254).

DependencyTimeoutException

The system timed out while trying to fulfill the request. The request can be retried.

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.

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 .

HTTP Status Code: 400

LimitExceededException

The request was rejected because a 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
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java V2
- AWS SDK for JavaScript
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

UpdateCustomKeyStore

Changes the properties of a custom key store. Use the `CustomKeyStoreId` parameter to identify the custom key store you want to edit. Use the remaining parameters to change the properties of the custom key store.

You can only update a custom key store that is disconnected. To disconnect the custom key store, use DisconnectCustomKeyStore (p. 67). To reconnect the custom key store after the update completes, use ConnectCustomKeyStore (p. 8). To find the connection state of a custom key store, use the DescribeCustomKeyStores (p. 52) operation.

The `CustomKeyStoreId` parameter is required in all commands. Use the other parameters of `UpdateCustomKeyStore` to edit your key store settings.

- Use the `NewCustomKeyStoreName` parameter to change the friendly name of the custom key store to the value that you specify.
- Use the `KeyStorePassword` parameter tell AWS KMS the current password of the `kmsuser` crypto user (CU) in the associated AWS CloudHSM cluster. You can use this parameter to fix connection failures that occur when AWS KMS cannot log into the associated cluster because the `kmsuser` password has changed. This value does not change the password in the AWS CloudHSM cluster.
- Use the `CloudHsmClusterId` parameter to associate the custom key store with a different, but related, AWS CloudHSM cluster. You can use this parameter to repair a custom key store if its AWS CloudHSM cluster becomes corrupted or is deleted, or when you need to create or restore a cluster from a backup.

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

This operation is part of the custom key store feature feature in AWS KMS, which combines the convenience and extensive integration of AWS KMS with the isolation and control of a single-tenant key store.

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 (p. 8)
- CreateCustomKeyStore (p. 15)
- DeleteCustomKeyStore (p. 46)
- DescribeCustomKeyStores (p. 52)
- DisconnectCustomKeyStore (p. 67)

Request Syntax

```
{
   "CloudHsmClusterId": "string",
   "CustomKeyStoreId": "string",
   "KeyStorePassword": "string",
   "NewCustomKeyStoreName": "string"
}
```

Request Parameters

For information about the parameters that are common to all actions, see Common Parameters (p. 252).

The request accepts the following data in JSON format.

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

CustomKeyStoreId (p. 213)

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 (p. 52) operation.

Type: String

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

Required: Yes

CloudHsmClusterId (p. 213)

Associates the custom key store with a related AWS CloudHSM cluster.

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.

Type: String

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

Required: No

KeyStorePassword (p. 213)

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 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.

Type: String

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

Required: No

NewCustomKeyStoreName (p. 213)

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.

Type: String

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

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 Errors (p. 254).

CloudHsmClusterInvalidConfigurationException

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

- The 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 cluster. Do not delete or change them. To get information about a particular security group, use the DescribeSecurityGroups operation.
- The cluster must contain at least as many HSMs as the operation requires. To add HSMs, use the AWS CloudHSM CreateHsm operation.

  For the CreateCustomKeyStore (p. 15), UpdateCustomKeyStore (p. 213), and CreateKey (p. 26) operations, the AWS CloudHSM cluster must have at least two active HSMs, each in a different Availability Zone. For the ConnectCustomKeyStore (p. 8) 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 a custom 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 that is associated with the custom 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.

Specify a 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.

Clusters that share a backup history have the same cluster certificate. To view the cluster certificate of a 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 (p. 52) operation.

This exception is thrown under the following conditions:

- You requested the CreateKey (p. 26) or GenerateRandom (p. 109) operation in a custom key store that is not connected. These operations are valid only when the custom key store `ConnectionState` is `CONNECTED`.
- You requested the UpdateCustomKeyStore (p. 213) or DeleteCustomKeyStore (p. 46) operation on a custom key store that is not disconnected. This operation is valid only when the custom key store `ConnectionState` is `DISCONNECTED`.
- You requested the ConnectCustomKeyStore (p. 8) operation on a custom key store with a `ConnectionState` of `DISCONNECTING` or `FAILED`. This operation is valid for all other `ConnectionState` values.

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

See Also

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

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java V2
- AWS SDK for JavaScript
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

UpdateKeyDescription

Updates the description of a KMS key. To see the description of a KMS key, use DescribeKey (p. 56).

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 (p. 26)
- DescribeKey (p. 56)

Request Syntax

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

Request Parameters

For information about the parameters that are common to all actions, see Common Parameters (p. 252).

The request accepts the following data in JSON format.

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

Description (p. 218)

New description for the KMS key.

Type: String

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

Required: Yes

KeyId (p. 218)

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 (p. 151) or DescribeKey (p. 56).

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 Errors (p. 254).

DependencyTimeoutException

The system timed out while trying to fulfill the request. The request can be retried.

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.

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 .

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
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java V2
- AWS SDK for JavaScript
- AWS SDK for PHP V3
- AWS SDK for Python
- 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 Updating the primary Region 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 (p. 56) operation on the primary key or any replica key. To create a replica key, use the ReplicateKey (p. 178) 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 (p. 56) 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 (p. 26)
- ReplicateKey (p. 178)

Request Syntax

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

Request Parameters

For information about the parameters that are common to all actions, see Common Parameters (p. 252).

The request accepts the following data in JSON format.

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

KeyId (p. 222)

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 (p. 151) or DescribeKey (p. 56).

Type: String

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

Required: Yes

PrimaryRegion (p. 222)

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 Errors (p. 254).

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.

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 .

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
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java V2
- AWS SDK for JavaScript
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

Verify

Verifies a digital signature that was generated by the Sign (p. 197) 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 verify a digital signature, you can use the `Verify` operation. Specify the same asymmetric KMS key, message, and signing algorithm that were used to produce the signature.

You can also verify the digital signature by using the public key of the KMS key outside of AWS KMS. Use the GetPublicKey (p. 125) 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.

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 (p. 197)

Request Syntax

```
{
   "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 (p. 252).

The request accepts the following data in JSON format.

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

KeyId (p. 224)

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 (p. 151) or DescribeKey (p. 56). To get the alias name and alias ARN, use ListAliases (p. 136).

Type: String

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

Required: Yes

Message (p. 224)

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 (p. 224)

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 (p. 224)

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`

Required: Yes

GrantTokens (p. 224)

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 (p. 224)

Tells AWS KMS whether the value of the `Message` parameter is a message or message digest. The default value, RAW, indicates a message. To indicate a message digest, enter `DIGEST`.

Important
Use the `DIGEST` value only when the value of the `Message` parameter is a message digest. If you use the `DIGEST` value with a raw message, the security of the verification operation can be compromised.

Type: String

Valid Values: `RAW | DIGEST`

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 (p. 226)

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 (p. 226)

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 (p. 226)

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`

Errors

For information about the errors that are common to all actions, see Common Errors (p. 254).

DependencyTimeoutException

The system timed out while trying to fulfill the request. The request can be retried.

HTTP Status Code: 500

DisabledException

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

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`. To find the `KeyUsage` of a KMS key, use the DescribeKey (p. 56) operation.

To find the encryption or signing algorithms supported for a particular KMS key, use the DescribeKey (p. 56) 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.

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 .

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
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java V2
- AWS SDK for JavaScript
- AWS SDK for PHP V3
- AWS SDK for Python
- 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.

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 (p. 105)

Request Syntax

```
{
   "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 (p. 252).

The request accepts the following data in JSON format.

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

KeyId (p. 229)

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 (p. 229)

The HMAC to verify. Enter the HMAC that was generated by the GenerateMac (p. 105) 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 (p. 229)

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 (p. 229)

The message that will be used in the verification. Enter the same message that was used to generate the HMAC.

GenerateMac (p. 105) 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

GrantTokens (p. 229)

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 (p. 230)

The HMAC KMS key used in the verification.

Type: String

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

MacAlgorithm (p. 230)

The MAC algorithm used in the verification.

Type: String

Valid Values: `HMAC_SHA_224 | HMAC_SHA_256 | HMAC_SHA_384 | HMAC_SHA_512`

MacValid (p. 230)

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 Errors (p. 254).

DisabledException

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

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`. To find the `KeyUsage` of a KMS key, use the DescribeKey (p. 56) operation.

To find the encryption or signing algorithms supported for a particular KMS key, use the DescribeKey (p. 56) 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.

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 .

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
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java V2
- AWS SDK for JavaScript
- AWS SDK for PHP V3
- AWS SDK for Python
- 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

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 Go
- 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 the custom key store.

Type: String

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

Required: No

ConnectionErrorCode

Describes the connection error. This field appears in the response only when the `ConnectionState` is `FAILED`. For help resolving these errors, see How to Fix a Connection Failure in *AWS Key Management Service Developer Guide*.

Valid values are:

- `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.
- `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.
- `INVALID_CREDENTIALS` - AWS KMS does not have the correct password for 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 key store password value for the custom key store.
- `NETWORK_ERRORS` - Network errors are preventing AWS KMS from connecting to 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 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.

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`

Required: No

ConnectionState

Indicates whether the custom key store is connected to its AWS CloudHSM cluster.

You can create and use KMS keys in your custom key stores only when its connection state is `CONNECTED`.

The value is `DISCONNECTED` if the key store has never been connected or you use the DisconnectCustomKeyStore (p. 67) operation to disconnect it. If the value is `CONNECTED` but you are having trouble using the custom key store, make sure that its associated AWS CloudHSM cluster is active and contains at least one active HSM.

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

TrustAnchorCertificate

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

Type: String

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

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 Go
- 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 includes the specified encryption context.

AWS KMS applies the grant constraints only to cryptographic operations that support an encryption context, that is, all cryptographic operations with a symmetric encryption KMS key. Grant constraints are not applied to operations that do not support an encryption context, such as cryptographic operations with HMAC KMS keys or asymmetric KMS keys, and management operations, such as DescribeKey (p. 56) or RetireGrant (p. 185).

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: in the AWS Key Management Service Developer Guide .

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

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 Go
- 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

A list of key-value pairs that must be present in the encryption context of certain subsequent operations that the grant allows.

Type: GrantConstraints (p. 239) 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.

The `GranteePrincipal` field in the `ListGrants` response usually contains the user or role designated as the grantee principal in the grant. However, when the grantee principal in the grant is an AWS service, the `GranteePrincipal` field contains the service principal, which might represent several different grantee principals.

Type: String

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

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

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 (p. 19) 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`

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

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 Go
- 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 Go
- 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 (p. 26) and DescribeKey (p. 56) 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 a custom key store, AWS KMS creates the key material for the KMS key in the associated AWS CloudHSM cluster. This value is present only when the KMS key is created in a custom key store.

Type: String

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

Required: No

CreationDate

The date and time when the KMS key was created.

Type: Timestamp

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 will support 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`

Required: No

CustomKeyStoreId

A unique identifier for the custom key store that contains the KMS key. This value 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

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`

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

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`

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`

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 (p. 56) 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 (p. 249) 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`

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`

Required: No

ValidTo

The time at which the imported key material expires. When the 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 whose `ExpirationModel` is `KEY_MATERIAL_EXPIRES`, otherwise this value is omitted.

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 Go
- 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 (p. 56) 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 (p. 250) 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 (p. 250) 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 Go
- 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 Go
- 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.

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 Go
- 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 Signature Version 4 Signing Process in the *Amazon Web Services General Reference*.

Action

The action to be performed.

Type: string

Required: Yes

Version

The API version that the request is written for, expressed in the format YYYY-MM-DD.

Type: string

Required: Yes

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 Task 2: Create a String to Sign for Signature Version 4 in the *Amazon Web Services General Reference*.

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 Handling Dates in Signature Version 4 in the *Amazon Web Services General Reference*.

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 Security Token Service, go to AWS Services That Work with IAM in the *IAM User Guide*.

Condition: If you're using temporary security credentials from the AWS Security Token Service, 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 Task 1: Create a Canonical Request For Signature Version 4 in the *Amazon Web Services General Reference*.

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 Errors

This section lists the errors common to the API actions of all AWS services. For errors specific to an API action for this service, see the topic for that API action.

AccessDeniedException

You do not have sufficient access to perform this action.

HTTP Status Code: 400

IncompleteSignature

The request signature does not conform to AWS standards.

HTTP Status Code: 400

InternalFailure

The request processing has failed because of an unknown error, exception or failure.

HTTP Status Code: 500

InvalidAction

The action or operation requested is invalid. Verify that the action is typed correctly.

HTTP Status Code: 400

InvalidClientTokenId

The X.509 certificate or AWS access key ID provided does not exist in our records.

HTTP Status Code: 403

InvalidParameterCombination

Parameters that must not be used together were used together.

HTTP Status Code: 400

InvalidParameterValue

An invalid or out-of-range value was supplied for the input parameter.

HTTP Status Code: 400

InvalidQueryParameter

The AWS query string is malformed or does not adhere to AWS standards.

HTTP Status Code: 400

MalformedQueryString

The query string contains a syntax error.

HTTP Status Code: 404

MissingAction

The request is missing an action or a required parameter.

HTTP Status Code: 400

MissingAuthenticationToken

The request must contain either a valid (registered) AWS access key ID or X.509 certificate.

HTTP Status Code: 403

MissingParameter

A required parameter for the specified action is not supplied.

HTTP Status Code: 400

NotAuthorized

You do not have permission to perform this action.

HTTP Status Code: 400

OptInRequired

The AWS access key ID needs a subscription for the service.

HTTP Status Code: 403

RequestExpired

The request reached the service more than 15 minutes after the date stamp on the request or more than 15 minutes after the request expiration date (such as for pre-signed URLs), or the date stamp on the request is more than 15 minutes in the future.

HTTP Status Code: 400

ServiceUnavailable

The request has failed due to a temporary failure of the server.

HTTP Status Code: 503

ThrottlingException

The request was denied due to request throttling.

HTTP Status Code: 400

ValidationError

The input fails to satisfy the constraints specified by an AWS service.

HTTP Status Code: 400