Alibaba Cloud
Key Management Service

User Guide

Issue: 20191024

Legal disclaimer

Alibaba Cloud reminds you to carefully read and fully understand the terms and conditions of this legal disclaimer before you read or use this document. If you have read or used this document, it shall be deemed as your total acceptance of this legal disclaimer.

1. **You shall download and obtain this document from the Alibaba Cloud website or other Alibaba Cloud-authorized channels, and use this document for your own legal business activities only. The content of this document is considered confidential information of Alibaba Cloud. You shall strictly abide by the confidentiality obligations. No part of this document shall be disclosed or provided to any third party for use without the prior written consent of Alibaba Cloud.**
2. **No part of this document shall be excerpted, translated, reproduced, transmitted, or disseminated by any organization, company, or individual in any form or by any means without the prior written consent of Alibaba Cloud.**
3. **The content of this document may be changed due to product version upgrades, adjustments, or other reasons. Alibaba Cloud reserves the right to modify the content of this document without notice and the updated versions of this document will be occasionally released through Alibaba Cloud-authorized channels. You shall pay attention to the version changes of this document as they occur and download and obtain the most up-to-date version of this document from Alibaba Cloud-authorized channels.**
4. **This document serves only as a reference guide for your use of Alibaba Cloud products and services. Alibaba Cloud provides the document in the context that Alibaba Cloud products and services are provided on an "as is", "with all faults" and "as available" basis. Alibaba Cloud makes every effort to provide relevant operational guidance based on existing technologies. However, Alibaba Cloud hereby makes a clear statement that it in no way guarantees the accuracy, integrity, applicability, and reliability of the content of this document, either explicitly or implicitly. Alibaba Cloud shall not bear any liability for any errors or financial losses incurred by any organizations, companies, or individuals arising from their download, use, or trust in this document. Alibaba Cloud shall not, under any circumstances, bear responsibility for any indirect, consequent**

ial, exemplary, incidental, special, or punitive damages, including lost profits arising from the use or trust in this document, even if Alibaba Cloud has been notified of the possibility of such a loss.

5. **By law, all the contents in Alibaba Cloud documents, including but not limited to pictures, architecture design, page layout, and text description, are intellectual property of Alibaba Cloud and/or its affiliates. This intellectual property includes, but is not limited to, trademark rights, patent rights, copyrights, and trade secrets. No part of this document shall be used, modified, reproduced, publicly transmitted, changed, disseminated, distributed, or published without the prior written consent of Alibaba Cloud and/or its affiliates. The names owned by Alibaba Cloud shall not be used, published, or reproduced for marketing, advertising, promotion, or other purposes without the prior written consent of Alibaba Cloud. The names owned by Alibaba Cloud include, but are not limited to, "Alibaba Cloud", "Aliyun", "HiChina", and other brands of Alibaba Cloud and/or its affiliates, which appear separately or in combination, as well as the auxiliary signs and patterns of the preceding brands, or anything similar to the company names, trade names, trademarks, product or service names, domain names, patterns, logos, marks, signs, or special descriptions that third parties identify as Alibaba Cloud and/or its affiliates.**
6. **Please contact Alibaba Cloud directly if you discover any errors in this document.**

Document conventions

Style	Description	Example
	A danger notice indicates a situation that will cause major system changes, faults, physical injuries, and other adverse results.	 **Danger:** Resetting will result in the loss of user configuration data.
	A warning notice indicates a situation that may cause major system changes, faults, physical injuries, and other adverse results.	 **Warning:** Restarting will cause business interruption. About 10 minutes are required to restart an instance.
	A caution notice indicates warning information, supplementary instructions, and other content that the user must understand.	 **Notice:** If the weight is set to 0, the server no longer receives new requests.
	A note indicates supplemental instructions, best practices, tips, and other content.	 **Note:** You can use Ctrl + A to select all files.
>	Closing angle brackets are used to indicate a multi-level menu cascade.	Click Settings > Network > Set network type.
Bold	Bold formatting is used for buttons, menus, page names, and other UI elements.	Click **OK**.
`Courier font`	Courier font is used for commands.	Run the `cd /d C:/window` command to enter the Windows system folder.
Italic	Italic formatting is used for parameters and variables.	`bae log list --instanceid` *`Instance_ID`*
[] or [a|b]	This format is used for an optional value, where only one item can be selected.	`ipconfig [-all|-t]`

Style	Description	Example
{} or {a|b}	**This format is used for a required value, where only one item can be selected.**	`switch {active|stand}`

Contents

1 Use RAM to authorize KMS resources

KMS uses RAM to control access to resources. This topic describes the resource types, actions, and policy conditions in KMS.

An Alibaba Cloud account has full operation permissions on its own resources. RAM users and roles are granted varying operation permissions on resources through RAM authorization. Before you use RAM to authorize and access CMKs, make sure that you have read *What is RAM* and *#unique_5*.

Resource types in KMS

The following table lists all resource types and corresponding Alibaba Cloud Resource Names (ARNs) in KMS. They can be used in the `Resource` parameter of a RAM policy.

Resource type	ARN
Key container	acs:kms:${region}:${account}:key
Alias container	acs:kms:${region}:${account}:alias
Key	acs:kms:${region}:${account}:key/${key-id}
Alias	acs:kms:${region}:${account}:alias/${alias-name}

Actions in KMS

KMS defines actions used in RAM policies and these actions correspond to different API operations that require access control. Actions must be in the `kms:${api-name}` format.

> **Note:**
> **The `DescribeRegions` API operation requires no access control. The DescribeRegions API operation can be called by Alibaba Cloud accounts, RAM users, or RAM roles when they pass RAM authentication.**

The following table lists the relationship between KMS API operations, RAM actions, and resource types.

Operation	Action	Resource type
ListKeys	kms:ListKeys	Key container
CreateKey	kms:CreateKey	Key container
DescribeKey	kms:DescribeKey	Key
UpdateKeyDescription	kms:UpdateKeyDescription	Key
EnableKey	kms:EnableKey	Key
DisableKey	kms:DisableKey	Key
ScheduleKeyDeletion	kms:ScheduleKeyDeletion	Key
CancelKeyDeletion	kms:CancelKeyDeletion	Key
GetParametersForImport	kms:GetParametersForImport	Key
ImportKeyMaterial	kms:ImportKeyMaterial	Key
DeleteKeyMaterial	kms:DeleteKeyMaterial	Key
Encrypt	kms:Encrypt	Key
GenerateDataKey	kms:GenerateDataKey	Key
GenerateDataKeyWithoutPlaintext	kms:GenerateDataKeyWithoutPlaintext	Key
Decrypt	kms:Decrypt	Key
ListAliases	kms:ListAliases	Alias container
CreateAlias	kms:CreateAlias	Alias and key
UpdateAlias	kms:UpdateAlias	Alias and key
DeleteAlias	kms:DeleteAlias	Alias and key
ListAliasesByKeyId	kms:ListAliasesByKeyId	Key
TagResource	kms:TagResource	Key
UntagResource	kms:UntagResource	Key
ListResourceTags	kms:ListResourceTags	Key
DescribeKeyVersion	kms:DescribeKeyVersion	Key
ListKeyVersions	kms:ListKeyVersions	Key
UpdateRotationPolicy	kms:UpdateRotationPolicy	Key

Policy conditions in KMS

You can add conditions to RAM policies to control access to KMS. RAM authentication will only be successful when the specified conditions are met. For example, you can use `acs:CurrentTime` to control the time period when a RAM policy is valid.

In addition to global conditions, you can use tags as filters to restrict the use of key-related API operations such as `Encrypt`, `Decrypt`, and `GenerateDataKey`. Filters must be in the `kms:tag/${tag-key}` format.

For more information, see *#unique_6*.

RAM policy examples

- **A RAM policy allowing users to access all KMS resources**

```
{
  "Version": "1",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "kms:*"
      ],
      "Resource": [
        "*"
      ]
    }
  ]
}
```

- **A RAM policy allowing users to view keys, aliases, and key usage permissions**

```
{
  "Version": "1",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "kms:List*", "kms:Describe*",
        "kms:Encrypt", "kms:Decrypt", "kms:GenerateDataKey"
      ],
      "Resource": [
        "*"
      ]
    }
  ]
```

```
}
```

- **A RAM policy allowing users to perform operations on keys that contain the following tag:**
  - **Tag key:** `Project`
  - **Tag value:** `Apollo`

```
{
    "Version": "1",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "kms:Encrypt", "kms:Decrypt", "kms:GenerateDataKey"
            ],
            "Resource": [
                "*"
            ],
            "Condition": {
                "StringEqualsIgnoreCase": {
                    "kms:tag/Project": [
                        "Apollo"
                    ]
                }
            }
        }
    ]
}
```

2 Use ActionTrail to record KMS events

KMS has been integrated with the *ActionTrail* service. You can view the events performed by all users (including the primary account and RAM users) on your resource instances in ActionTrail.

The KMS information recorded by ActionTrail includes all APIs except DescribeRegions. For more information, see *#unique_8*.

For event details, see *#unique_9*.

3 Use aliases

Aliases are an optional parameter used to identify Customer Master Keys (CMKs).

Aliases must be unique within a region for each Alibaba Cloud account. Different regions can contain identical aliases. An alias can be bound to a single CMK within a region, but a CMK can have multiple bound aliases.

Although aliases are bound to CMKs, aliases are resources independent from the CMKs to which they are bound. Aliases have the following characteristics:

- **You can call the *#unique_11* API operation to bind an alias to a different CMK. This operation will not affect the CMK.**
- **Deleting an alias will not delete the CMK that it is bound to.**
- **A RAM user must be authorized before it can perform operations on an alias. For more information, see *Use RAM to authorize KMS resources*.**
- **Aliases cannot be modified. To change the alias of a CMK, you must delete the old alias and create a new one for the CMK.**

You can replace the CMK ID with a bound alias in the following API operations:

- **DescribeKey**
- **Encrypt**
- **GenerateDataKey**
- **GenerateDataKeyWithoutPlaintext**

To call the preceding API operations, the RAM user must have the relevant permissions on the CMK. The RAM user does not need to have permission on the aliases.

You can perform the following alias-related operations:

- *Create an alias*
- *Update an alias*
- *Delete an alias*
- *List aliases*
- *List aliases bound to a specified CMK*

For all of the preceding operations, you must specify a complete alias with the `alias/` prefix.

```
//The example alias with the alias/ prefix
alias/example
```

Create an alias

- **Aliases must contain the `alias/` prefix. The alias can contain letters, digits, underscores (_), hyphens (-), and forward slashes (/). Excluding the prefix, the alias must be 1 to 255 characters in length.**
- **To create an alias, a RAM user must have permissions on both the alias and the CMK that the alias is bound to.**
- **Creating a new alias for a CMK will not affect the existing aliases of the CMK.**
- **You can call the *#unique_17* API operation to create an alias.**

```
//A sample RAM policy used to create an alias: The 123456 RAM user
can create the alias/example alias for the 08ec3bb9-034f-485b-b1cd-
3459baa889c7 CMK
{
  "Version": "1",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "kms:CreateAlias"
      ],
      "Resource": [
        "acs:kms:cn-hangzhou:123456:key/08ec3bb9-034f-485b-b1cd-
3459baa889c7",
        "acs:kms:cn-hangzhou:123456:alias/example"
      ]
    }
  ]
}
//Create an alias
aliyuncli kms CreateAlias --KeyId 08ec3bb9-034f-485b-b1cd-3459baa889c7
 --AliasName alias/example
```

Update an alias

- **This operation changes the bound CMK of an alias to a different CMK. You can call the *#unique_11*API operation to create an alias.**
- **To update an alias, a RAM user must have permissions on the source and destination CMKs as well as the alias.**

```
//A sample RAM policy used to update an alias: The 123456 RAM user can
 bind the alias/example alias that has been bound to the 08ec3bb9-034f
-485b-b1cd-3459baa889c7 CMK to the 127d2f84-ee5f-4f4d-9d41-dbc1aca287
88 CMK
{
```

```
  "Version": "1",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "kms:UpdateAlias"
      ],
      "Resource": [
        "acs:kms:cn-hangzhou:123456:key/08ec3bb9-034f-485b-b1cd-
3459baa889c7",
        "acs:kms:cn-hangzhou:123456:key/127d2f84-ee5f-4f4d-9d41-
dbc1aca28788",
        "acs:kms:cn-hangzhou:123456:alias/example"
      ]
    }
  ]
}
//Update an alias
aliyuncli kms UpdateAlias --AliasName alias/example --KeyId 127d2f84-
ee5f-4f4d-9d41-dbc1aca28788
```

Delete an alias

- **Deleting an alias will not affect the CMK that alias is bound to. You can call the *#unique_18* API operation to delete an alias.**
- **To delete an alias, a RAM user must have permissions on both the alias and the CMK that the alias is bound to.**

```
//A sample RAM policy used to delete an alias: The 123456 RAM user
 can delete the alias/example alias of the 127d2f84-ee5f-4f4d-9d41-
dbc1aca28788 CMK
{
  "Version": "1",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "kms:DeleteAlias"
      ],
      "Resource": [
        "acs:kms:cn-hangzhou:123456:key/127d2f84-ee5f-4f4d-9d41-
dbc1aca28788",
        "acs:kms:cn-hangzhou:123456:alias/example"
      ]
    }
  ]
}
//Delete an alias
aliyuncli kms DeleteAlias --AliasName alias/example
```

List aliases

- **This operation lists information of all aliases. You can call the *#unique_19* API operation to list aliases.**
- **To list all aliases, a RAM user must have the permissions on alias resources.**

```
//A sample RAM policy used to list aliases
```

```
{
  "Version": "1",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "kms:ListeAliases"
      ],
      "Resource": [
        "acs:kms:cn-hangzhou:123456:alias"
      ]
    }
  ]
}
//List aliases
aliyuncli kms ListAliases
```

List aliases bound to a specified CMK

- **This operation lists information of the aliases bound to a specified CMK. You can call the *#unique_20* API operation to list the aliases bound to a specified CMK.**
- **To list the aliases bound to a specified CMK, a RAM user must have the permissions on the specified CMK.**

```
//A sample RAM policy used to list the aliases bound to a specified
CMK: List all aliases bound to the 127d2f84-ee5f-4f4d-9d41-dbc1aca287
88 CMK
{
  "Version": "1",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "kms:DeleteAlias"
      ],
      "Resource": [
        "acs:kms:cn-hangzhou:123456:key/127d2f84-ee5f-4f4d-9d41-
dbc1aca28788"
      ]
    }
  ]
}
//List aliases bound to a specified CMK
aliyuncli kms ListAliasesByKeyId --KeyId 127d2f84-ee5f-4f4d-9d41-
dbc1aca28788
```

4 Import key materials

This topic describes how to import external key materials and delete the external key materials imported into a CMK.

CMK overview

Customer master keys (CMKs) are basic resources of KMS. CMKs are composed of key IDs, basic metadata (such as key state) and key materials used to encrypt and decrypt data. KMS generates key materials when you call the *#unique_22* operation. You can choose to create a key from external key materials. In this case, KMS does not generate key materials for the CMK you create and you can import your own key materials to the CMK. You can call the *#unique_23* operation to view the source of key materials. When the Origin value in the KeyMetadata is Aliyun_KMS, the key material was generated by KMS and the corresponding key is a normal key. If the Origin value is EXTERNAL, the key material was imported from an external source and the corresponding key is an external key.

Precautions

When you select an external key material source and use the key material you imported, take note of the following points:

- **You must make sure that qualified random sources are used to generate key materials.**
- **You must ensure the reliability of the key materials.**
  - **KMS ensures the high availability of imported key materials, but cannot ensure that the imported key material has the same reliability as the key material generated by KMS.**
  - **You can directly call the *#unique_24* operation to delete the imported key material. Alternatively, you can set an expiration time to automatically delete the imported key material after it expires (without deleting CMKs). The key material generated by KMS cannot be directly deleted. Instead, you can call**

the #unique_25 operation to delete the key material along with CMK after 7 to 30 days.

- After you delete the imported key material, you can re-import the same key material to make the relevant CMK available again. Therefore, you need to save a copy of the key material.

- Each CMK can only have one imported key material. After you import the key material to a CMK, this CMK is bound to this key material. Even if the key material expires or is deleted, you cannot import any other key material into the CMK. If you need to rotate a CMK that uses the external key material, you must create a new CMK and then import new key material.
- CMKs are independent. When you use one CMK to encrypt data, you cannot use another CMK to decrypt the data, even if these CMKs use the same key material.
- You can only import 256-bit symmetric keys as key materials.

Import key materials

1. Create an external key.

   First, you must create an external key.

   To do this, set Key Material Source to External on the Create Key page of the KMS console, or call the CreateKey operation and set Origin to EXTERNAL. By

choosing to create an external key, you acknowledge that you have read and understood the *Precautions* and *Import key materials* sections of this topic.

Example

```
aliyuncli kms CreateKey --Origin EXTERNAL --Description "External
key"
```

2. Obtain key material import parameters.

   After you create an external key and before you import the key material, you must obtain the key material import parameters.

   You can obtain the key material import parameters through the console or by calling the *#unique_27* operation. The key material import parameters include a public key used to encrypt the key material and an import token.

   Example

```
aliyuncli kms GetParametersForImport --KeyId 1339cb7d-54d3-47e0-b595
-c7d3dba82b6f --WrappingAlgorithm RSAES_OAEP_SHA_1 --WrappingKeySpec
 RSA_2048
```

3. Import key materials.

   You can import key materials for external keys that do not yet have key materials , re-import key materials that have expired or been deleted, or reset the key material expiration time. The import token is bound to the public key used to encrypt key material. A single token can only be used to import the key material for the CMK specified at the time of generation. An import token is valid for 24 hours and can be used multiple times during this period. After the token expires , you must obtain a new import token and public encryption key.

   - First, use the public encryption key to encrypt the key material. The public encryption key is a 2048-bit RSA public key. The encryption algorithm used must be consistent with that specified when obtaining the key material import parameters. Because the public encryption key returned when you call the operation is Base64 encoded, you must first perform Base64 decoding before using the public encryption key. KMS supports the following encryption

algorithms: RSAES_OAEP_SHA_1, RSAES_OAEP_SHA_256, and RSAES_PKCS 1_V1_5.

- After encryption, you must perform Base64 encoding on the encrypted key material and then import the key material along with the import token to KMS as *#unique_28* parameters.

Example

```
aliyuncli kms ImportKeyMaterial --KeyId 1339cb7d-54d3-47e0-b595-
c7d3dba82b6f --EncryptedKeyMaterial xxx --ImportToken xxxx
```

Delete key materials

- After importing key materials, you can use external keys just like normal keys . External keys differ from normal keys because their key material can expire or be manually deleted. After the key material expires or is deleted, the key will no longer function and ciphertext data encrypted using this key cannot be decrypted before you re-import the same key material.
- If a key enters the PendingDeletion state after its key material expires or is deleted, the key state does not change. Otherwise, the key state changes to PendingImport.
- You can use the console or call the *#unique_24* operation to delete the key material.

Example

```
aliyuncli kms DeleteKeyMaterial --KeyId xxxx
```

Examples

Use OPENSSL to encrypt and upload key material

- Create an external key.
- Generate the key material. The key material must be a 256-bit symmetric key. In this example, OPENSSL is used to generate a 32-byte random number.

```
1.openssl rand -out KeyMaterial.bin 32
```

- Obtain key material import parameters.

- **Encrypt key materials.**
  - **First, you must perform Base64 decoding on the public encryption key.**
  - **Then, use the encryption algorithm (RSAES_OAEP_SHA_1 here) to encrypt the key material.**
  - **Finally, perform Base64 encoding on the encrypted key material and save it as a text file.**

```
openssl rand -out KeyMaterial.bin 32
openssl enc -d -base64 -A -in PublicKey_base64.txt -out PublicKey.
bin
openssl rsautl -encrypt -in KeyMaterial.bin -oaep -inkey PublicKey
.bin  -keyform DER  -pubin -out EncryptedKeyMaterial.bin
openssl enc -e -base64 -A -in EncryptedKeyMaterial.bin -out
EncryptedKeyMaterial_base64.txt
```

- **Upload the encrypted key material and import token.**

Use JAVA SDK to encrypt and upload the key material

```
//Use the latest KMS JAVA SDK
//KmsClient.java

import com.aliyuncs.kms.model.v20160120.*;
import com.aliyuncs.profile.DefaultProfile;

//KMS API encapsulation
public class KmsClient {
        DefaultAcsClient client;

        public KmsClient( String region_id, String ak, String secret)
 {
                DefaultProfile profile = DefaultProfile.getProfile(
region_id, ak, secret);
                this.client = new DefaultAcsClient(profile);
        }

        public CreateKeyResponse createKey() throws Exception {
                CreateKeyRequest request = new CreateKeyRequest();
                request.setOrigin("EXTERNAL"); //Create an external
key
                return this.client.getAcsResponse(request);
        }
        //... Omitted. The remaining operations are the same as those
in the API method.
}
//example.java
import com.aliyuncs.kms.model.v20160120.*;
import KmsClient
import java.security.KeyFactory;
import java.security.PublicKey;
import java.security.spec.MGF1ParameterSpec;
import javax.crypto.Cipher;
import javax.crypto.spec.OAEPParameterSpec;
import javax.crypto.spec.PSource.PSpecified;
import java.security.spec.X509EncodedKeySpec;
import java.util.Random;
import javax.xml.bind.DatatypeConverter;
```

```
public class CreateAndImportExample {
        public static void main(String[] args) {
                String regionId = "cn-hangzhou";
        String accessKeyId = "*** Provide your AccessKeyId ***";
        String accessKeySecret = "*** Provide your AccessKeySecret
 ***";
        KmsClient kmsclient = new KmsClient(regionId,accessKeyId,
accessKeySecret);
        //Create External Key
        try {
                CreateKeyResponse keyResponse = kmsclient.createKey();
                String keyId = keyResponse.KeyMetadata.getKeyId();
                //Generate a 32-bit random number
                byte[] keyMaterial = new byte[32];
                new Random().nextBytes(keyMaterial);
                //Obtain key material import parameters
                GetParametersForImportResponse paramResponse =
kmsclient.getParametersForImport(keyId,"RSAES_OAEP_SHA_256");
                String importToekn = paramResponse.getImportToken();
                String encryptPublicKey = paramResponse.getPublicKey
();
                //Perform Base64 decoding on the public encryption key
                byte[] publicKeyDer = DatatypeConverter.parseBase6
4Binary(encryptPublicKey);
                //Parse the RSA public key
                KeyFactory keyFact = KeyFactory.getInstance("RSA");
                X509EncodedKeySpec spec = new X509EncodedKeySpec(
publicKeyDer);
                PublicKey publicKey = keyFact.generatePublic(spec);
                //Encrypt the key material
                Cipher oaepFromAlgo = Cipher.getInstance("RSA/ECB/
OAEPWithSHA-1AndMGF1Padding");
                String hashFunc = "SHA-256";
                OAEPParameterSpec oaepParams = new OAEPParameterSpec(
hashFunc, "MGF1", new MGF1ParameterSpec(hashFunc), PSpecified.DEFAULT
);
                oaepFromAlgo.init(Cipher.ENCRYPT_MODE, publicKey,
oaepParams);
                byte[] cipherDer = oaepFromAlgo.doFinal(keyMaterial);
                //You must perform Base64 encoding on the encrypted
key material
                String encryptedKeyMaterial = DatatypeConverter.
printBase64Binary(cipherDer);
                //Import the key material
                Long expireTimestamp = 1546272000L; //Unix timestamp,
precise to the second, 0 indicates no expiration
                        kmsClient.importKeyMaterial(keyId,encryptedK
eyMaterial, expireTimestamp);
        } catch(Exception e) {
                //... Omitted
        }
        }
}
```

5 Managed HSM (preview)

5.1 Overview

Managed HSM is an important feature of Key Management Service (KMS) to enable easy access to certified Hardware Security Modules (HSMs) provided by Alibaba Cloud.

An HSM is a hardware device that performs cryptographic operations, and generates and stores keys. You can protect your most sensitive workloads and assets provided by Alibaba Cloud, by hosting keys in these highly secure hardware devices.

Supported regions

You can use Managed HSM in the following regions. This feature will be provided in more regions later.

Region	City	Region ID
China（Hong Kong）	Hong Kong	cn-hongkong
Singapore	Singapore	ap-southeast-1

Regulatory compliance

Managed HSM can help you meet stringent regulatory requirements. Based on different regulatory requirements in each local market, Alibaba Cloud offers HSMs certified by different third-party organizations to meet your localization and internationalization requirements.

For regions outside mainland China,

- **FIPS validation for hardware: Alibaba Cloud HSMs, including their hardware and firmware, have passed FIPS 140-2 Level 3 validation. For more information, see *Certificate #3254*.**
- **FIPS 140-2 Level 3 Compliance: Alibaba Cloud Managed HSM runs under FIPS Approved Level 3 mode of operation.**
- **PCI DSS: Alibaba Cloud Managed HSM complies with PCI DSS requirements.**

High security assurance

- **Hardware protection**

  Managed HSM helps you protect keys in KMS through hardware mechanisms . The plaintext key material of CMKs is only processed inside HSMs for key operations. It is kept within the hardware security boundary of HSMs.
- **Secure key generation**

  Randomness is crucial to the encryption strength of keys. Managed HSM uses a random number generation algorithm that is secure and licensed and has high system entropy seeds to generate key material. This protects keys from being recovered or predicted by attackers.

Ease of operation

Alibaba Cloud fully manages HSM hardware. This eliminates the costs otherwise incurred by the following hardware management operations:

- **Hardware lifecycle management**
- **HSM cluster management**
- **High availability and scalability management**
- **System patching**
- **Most disaster recovery operations**

Ease of integration

Native key management capabilities allow you to use the following features:

- **Key version management**
- **Automatic key rotation**
- **Resource tag management**
- **Controlled authorization**

These features enable rapid integration of your applications with HSMs, as well as integration of ECS, RDS, and other cloud services with Managed HSM. You can implement static encryption of cloud data without paying any R&D costs.

Key control

Managed HSM allows you to better control encryption keys on the cloud and move the most sensitive computing tasks and assets to the cloud.

When using both Managed HSM and *Bring Your Own Key (BYOK)*, you can have full control over the following items:

- **How key material is generated**
- **The key material that you import to the managed HSM cannot be exported, but can be destroyed.**
- **Key lifecycle**
- **Key persistence**

Low cost

You can benefit from the pay-as-you-go billing method of cloud computing. Compared with user-created key infrastructure by using local HSMs, Managed HSM eliminates hardware procurement costs, as well as subsequent R&D and O&M costs .

5.2 Using Managed HSM

This topic describes how to create and use keys through Managed HSM.

Enable free-trial version of Managed HSM

You can contact your pre-sales or after-sales consultant to enable the free-trial version of Managed HSM.

Create a key in Managed HSM

You can only use Managed HSM in some regions. For more information about supported regions, see *Supported regions*.

Create a key in the console

1. **Log on to the *KMS console*.**
2. **Select a region in the upper-left corner of the page. Click Create Key.**
3. **Select HSM from the Protection Level drop-down list.**
4. **Enter a description and click OK.**

After a key is created, its Protection Level is displayed on the Key Details and Keys pages.

Create a key by using Alibaba Cloud CLI

```
aliyun kms CreateKey --ProtectionLevel HSM --Description "Key1 in
Managed HSM"
```

The protection level of the key is displayed in the output after the key is created or in the output of DescribeKey called by using Alibaba Cloud CLI. Example:

```
{
  "KeyMetadata": {
    "CreationDate": "2019-07-04T13:14:15Z",
    "Description": "Key1 in Managed HSM",
    "KeyId": "1234abcd-12ab-34cd-56ef-12345678****",
    "KeyState": "Enabled",
    "KeyUsage": "ENCRYPT/DECRYPT",
    "DeleteDate": "",
    "Creator": "111122223333",
    "Arn": "acs:kms:cn-hongkong:111122223333:key/1234abcd-12ab-34cd-
56ef-12345678****",
    "Origin": "Aliyun_KMS",
    "MaterialExpireTime": "",
    "ProtectionLevel": "HSM"
  },
  "RequestId": "8eaeaa8b-4491-4f1e-a51e-f95a4e54620c"
}
```

Import external keys to Managed HSM

You can also import keys from user-created key infrastructure to Managed HSM. You only need to set Protection Level to HSM in the first step of the *import key material* process. This step is to create external keys. Keep the remaining steps unchanged.

The actions on the Alibaba Cloud side:

- **When you call `GetParametersForImport`, Alibaba Cloud generates a key pair for importing external keys in Managed HSM based on the HSM protection level, and returns the public key of the key pair.**
- **When you call `ImportKeyMaterial`, Alibaba Cloud imports the encrypted external key material to Managed HSM, and obtains the key material by unwrapping the HSM key. The imported plaintext key material will never be exported.**

Manage and use keys

All management and cryptographic features supported by KMS are applicable to keys created in Managed HSM. Specifically, you can:

- **Enable and disable keys.**
- **Manage key lifecycle.**
- **Manage key aliases.**

- **Manage key tags.**
- **Call key operations.**

Integration with other cloud products

Keys created in Managed HSM can be used in other cloud products such as ECS, RDS , and OSS through standard APIs of KMS, to protect your native Alibaba Cloud data . In this case, cloud products must support server-side encryption by using custom keys. You only need to select keys created in Managed HSM when configuring CMKs for server-side encryption on cloud products.

6 Key rotation

6.1 Overview

Keys are often used to protect data. The security of data is dependent on the security of its corresponding keys. You can use key versions and the periodic rotation mechanism to improve key security and implement security policies and best practices for data protection.

Security goals

You can use the periodic key rotation mechanism to:

- **Reduce the amount of data encrypted by each key**

  The security of a key is inversely proportional to the amount of data encrypted by it. This amount is usually defined by the total bytes of data or the total number of messages that are encrypted by the same key. For example, National Institute of Standards and Technology (NIST) defines the secure lifecycle of a key in GCM mode as the total number of messages encrypted based on the key . The periodic key rotation mechanism enables each key to remain secure and minimize vulnerability to cryptanalytic attacks.

- **Respond in advance to security events**

  In the early days of system design, key rotation was introduced as a routine O&M method. This provides the system with a method to handle security events when they occur, and complies with the fail early, fail often principle of software engineering. If key rotation is not executed until an emergency event has already occurred, the probability of system failure increases exponentially.

- **Provide logical isolation of data**

  Encrypted data is isolated with each key rotation from other data encrypted using different keys. The impact of key-related security events can be identified quickly and preventive measures can be taken.

- **Reduce the window of time to crack keys**

  Periodic rotation of encryption keys ensures that you can control and reduce the window of time for which the key and its encrypted data are vulnerable to being cracked. Attackers only have a limited period of time between rotation tasks

during which they are able to crack the key. This practice greatly increases the security of your data against cryptanalytic attacks.

Regulatory compliance

The periodic key rotation mechanism facilitates compliance with various regulations, which include but are not limited to:

- **Payment Card Industry Data Security Standard (PCI DSS)**
- **Cryptography-related industrial standards issued by State Cryptography Administration, such as GM/T 0051-2016**
- **Cryptography-related standards issued by NIST, such as NIST Publication 800-38D**

6.2 Automatic key rotation

This topic describes how to use automatic rotation of Customer Master Keys (CMKs) in KMS.

Key versions

CMKs support multiple key versions. Each key version represents an independen tly generated key. Multiple key versions of a CMK do not have any cryptographic relation to each other. KMS rotates CMKs automatically by generating new key versions.

There are two types of key versions:

- **Primary key versions**
  - **The primary key version of a CMK is an active encryption key. Each CMK only has a single primary key version at any time.**
  - **When you call API operations such as** `GenerateDataKey` **and** `Encrypt`**, KMS uses the primary key version of a specified CMK to encrypt the target plaintext.**
  - **You can call the** `DescribeKey` **API operation to view** `PrimaryKeyVersion` **attributes.**

- **Non-primary key versions**
  - **The non-primary key version of a CMK is an inactive encryption key. Each CMK can have any number of non-primary key versions.**
  - **Each non-primary key version was a primary key version in the past and used to act as the active encryption key.**
  - **When a new primary key version is created, KMS does not delete or disable non-primary key versions, which can be used to decrypt data.**

Note:
When you call the Encrypt API operation, the primary key version of a specified CMK is used. When you call the Decrypt API operation, the key version that was used to encrypt the ciphertext is selected.

You can generate a key version in either of the following ways:

- **Create a CMK**

  You can call the `CreateKey` API operation to create a CMK. If you set the `Origin` parameter to `Aliyun_KMS`, KMS generates an initial key version and sets it as the primary key version.
- **Execute an automatic rotation policy**

  After you configure an automatic rotation policy, KMS executes the policy on a periodic basis to generate new key versions.

Automatic key rotation

Configure a key rotation policy

When you call the `CreateKey` API operation to create a CMK, you can specify an automatic rotation policy for the CMK. You can call the `UpdateRotationPolicy` API operation to update the currently used automatic rotation policy. When you call the UpdateRotationPolicy API operation, you must configure the following parameters:

- `EnableAutomaticRotation`**: specifies whether to enable automatic rotation.**
- `RotationInterval`**: specifies the period for automatic rotation.**

You can call the `DescribeKey` API operation to view the automatic rotation policy information. The following policy-related parameters are displayed:

- `AutomaticRotation`: indicates whether automatic rotation is enabled. `Disabled` indicates that automatic rotation is not enabled. `Enabled` indicates that automatic rotation is enabled. `Suspended` indicates that KMS suspends execution of automatic rotation although automatic rotation is enabled. For more information, see *Effects of CMK status on automatic rotation*.
- `RotationInterval`: indicates the period for automatic rotation.

Execute an automatic rotation policy

When automatic rotation is enabled, KMS calculates the time of the next rotation using the following formula:

```
${NextRotationTime} = ${LastRotationTime} + ${RotationInterval}
```

The parameters are as follows:

- `LastRotationTime`: specifies the time that the last key version was created. You can call the `DescribeKey` API operation and check the `LastRotationDate` parameter to query this value.
- `NextRotationTime`: specifies the time that KMS will perform the next rotation task to create a new key version. You can call the `DescribeKey` API operation and check the `NextRotationDate` parameter to query this value.

Notice:
When you update the RotationInterval parameter of an automatic rotation policy, the calculated value of NextRotationTime may be a point in time in the past. This will not affect the execution of the automatic rotation policy. KMS will create a new key version as scheduled. If the calculated value of NextRotationTime is before the current time, KMS will execute the automatic rotation policy immediately.

Effects of CMK status on automatic rotation

A new key version can be created only when a CMK is in the Enabled state (the `KeyState` parameter is displayed as `Enabled`). You must take note of the following points:

- If a CMK is in the `Disabled` or `PendingDeletion` state, do not call the `UpdateRotationPolicy` API operation to update its automatic rotation policy.
- If a CMK enters the `Disabled` or `PendingDeletion` state after automatic rotation is enabled, KMS suspends execution of automatic rotation. When you call the

`DescribeKey` **API operation, the** `AutomaticRotation` **parameter will be displayed as** `Suspended`**. When the CMK enters the Enabled state, its automatic rotation policy becomes active again.**

Limits

The following keys do not support multiple key versions:

- **Service managed keys: the default keys managed by KMS for specific cloud services. These keys belong to users of specific cloud services and are used to provide basic encryption protection for user data.**
- **Bring Your Own Keys (BYOKs): the keys that you have imported to KMS. For more information, see *Import key materials*. The** `Origin` **value of these keys is** `External`**, so KMS does not generate key materials or initiate rotation tasks for these keys.**

The preceding key types do not support version-based key rotation. KMS does not support multiple BYOK versions. First, users have strong control over BYOK persistence and lifecycle, which makes management difficult and error-prone. For example, you must have on-premise key management facilities, data must be synchronized between on-premise and cloud facilities, and no grace period is provided when you delete key materials. The complexity of maintaining multiple versions of BYOKs makes it much more risky. Second, both primary and non-primary key versions may become unavailable at different times. For example, if key versions are deleted by KMS or imported again when they expire, it would be impossible to synchronize the CMKs and protected data or guarantee system integrity.

6.3 Manual key rotation

If your Customer Master Keys (CMKs) do not support version-based automatic rotation, you can manually rotate the CMKs. This is an alternative solution that does not depend on whether automatic key rotation is supported.

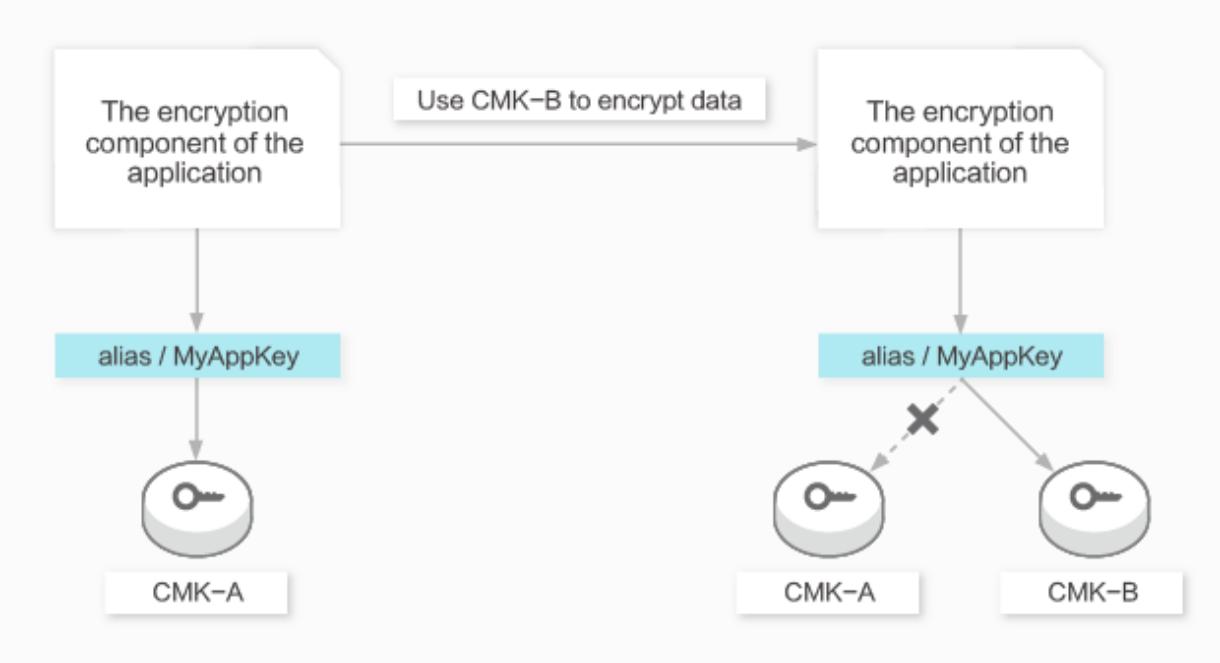
Custom data encryption scenario

On-premise or cloud applications can call the API operation to implement custom data encryption. Examples:

- **Encrypt sensitive data such as ID card numbers, credit card information, and home addresses before writing it to databases**

- Encrypt data at the client side before uploading it to OSS
- Encrypt service profiles that contain sensitive data and SSL key certificates such as application profiles

You can use the key alias feature to rotate encryption keys within applications. The ID and alias of the key are not required when you call the Decrypt API operation.



In this scenario, you must perform the following steps:

1. Initial configuration
   a. The administrator creates a CMK, whose ID is CMK-A.
   b. The administrator binds the alias/MyAppKey alias to CMK-A.
   c. When the application encryption module calls the `Encrypt` API operation, the value of the `KeyId` parameter is `alias/MyAppKey`. KMS finds that alias/MyAppKey is bound to CMK-A and then uses CMK-A to encrypt data.
   d. When the application decryption module calls the `Decrypt` API operation, the `KeyId` parameter is not used. KMS uses the CMK used to encrypt the data to decrypt the data.

2. **Manual rotation**
   a. **The administrator creates a CMK, whose ID is CMK-B.**
   b. **The administrator calls the `UpdateAlias` API operation to bind the alias/MyAppKey alias to CMK-B.**
   c. **When the application decryption module calls the `Encrypt` API operation, the value of the `KeyId` parameter is `alias/MyAppKey`. KMS finds that alias/MyAppKey is bound to CMK-B and uses CMK-B to encrypt data.**
   d. **When the application decryption module calls the `Decrypt` API operation, the `KeyId` parameter is not used. KMS uses the CMK used to encrypt the data to decrypt the data.**

Server encryption scenario

Other cloud services can encrypt their data by integrating KMS API operations. The following situations may occur in key rotation scenarios:

- **Automatic rotation policies configured on KMS affect the server encryption of other cloud services**

  Cause: After CMK encryption is configured for cloud services, they will call the `GenerateDataKey` API operation of KMS to generate data keys. When KMS generates a new primary key version, cloud services use the new version to generate new data key. A typical example of such cloud services is OSS. In this situation, if you want to enable automatic key rotation, you cannot use service managed keys or BYOKs imported to KMS because they do not support automatic rotation.
- **Automatic rotation policies configured on KMS do not affect server encryption of other cloud services**

  Cause: After CMK encryption is configured for cloud services, the services only call the GenerateDataKey API operation of KMS once to generate keys to encrypt specific resources. When KMS generates a new primary key version, cloud services will not use it. For example, when encrypting a cloud disk, ECS calls the `GenerateDataKey` API operation of KMS once to generate a volume encryption key. This key will not be updated again after it is created.

If automatic rotation policies configured on KMS do not affect server encryption of other cloud services or you want to rotate data keys when BYOKs are used, you

can change configurations and copy data to achieve the same effect as key rotation . These methods depend on the features of different cloud services. For more information, see documentation of respective cloud services.

7 Integration of Alibaba Cloud services with KMS

7.1 Integration with KMS

Alibaba Cloud can encrypt your data stored in Alibaba Cloud services by using keys from Key Management Service (KMS). Alibaba Cloud supports the Advanced Encryption Standard (AES) 256-bit algorithm for encryption, which meets the encryption requirements of sensitive data.

The integration of Alibaba Cloud services with KMS brings the following benefits:

- **Enhanced security and privacy protection for data stored in Alibaba Cloud services**

  Alibaba Cloud services can use KMS keys to encrypt any of your data, including the data that you can directly access or internal data of Alibaba Cloud services that you can only indirectly access, such as files generated by database engines. This ensures the security and privacy of your data stored in Alibaba Cloud services.
- **No need to develop your own data encryption system**

  To develop your own data encryption system, you must:

  - **Design a proper key hierarchy and data distribution mode to balance between encryption performance and security.**
  - **Design the key rotation and data re-encryption mechanisms.**
  - **Master cryptography technologies to ensure that your encryption algorithm is robust, secure, and tamper-proofing.**
  - **Improve the engineering robustness and reliability of your system to ensure data persistence.**

  The integration of Alibaba Cloud services with KMS resolves all these complex engineering and security issues for you and reduces R&D costs.

Select appropriate keys

You can select different types of keys stored in KMS for encryption based on your data protection requirements.

- **Service-managed keys**

  Each Alibaba Cloud service can create a default key, which is also called a service key, for you in KMS. Yo do not need to manage this service key. It is managed by the Alibaba Cloud service. In addition, you do not need to explicitly authorize the Alibaba Cloud service to use this service key. By using ActionTrail, you can audit the use of the service key by the Alibaba Cloud service.

  To allow you to easily identify the service key, KMS sets the Creator attribute of the service key to the code of the Alibaba Cloud service, and associates the service key with a special alias in the format of `acs/<Alibaba Cloud service code>`. For example, the Creator attribute of the service key created by Object Storage Service (OSS) is set to OSS, and the service key is associated with the alias acs/oss.

- **User-managed keys**

  You can use keys created by yourself to encrypt data stored in Alibaba Cloud services. This gives you more control over how data is encrypted. You must explicitly authorize Alibaba Cloud services to use your keys. You can use Resource Access Management (RAM) to perform the authorization. By configuring a permission policy and granting the policy to an Alibaba Cloud service in RAM, you can allow or deny the Alibaba Cloud service to use a specific customer master key (CMK) stored in KMS. When the Alibaba Cloud service requests the CMK from KMS, KMS checks the permission of the Alibaba Cloud service through RAM.

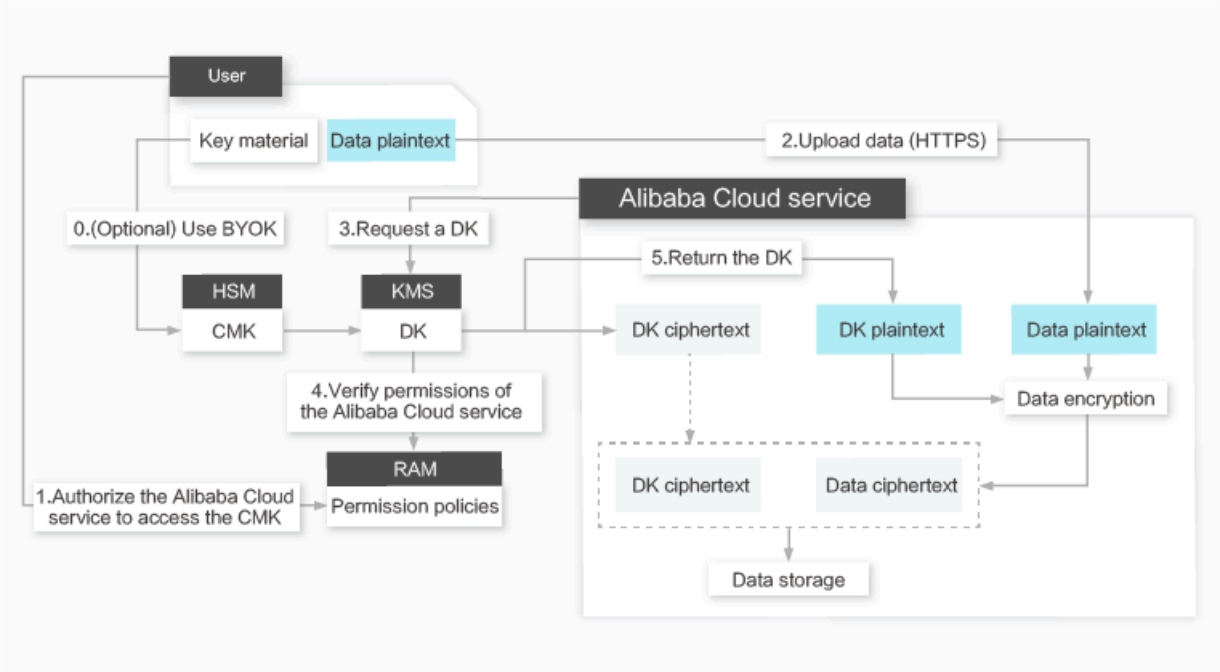
  Besides keys generated by KMS, you can securely import offline key materials to CMKs in KMS through the Bring Your Own Key (BYOK) feature and use these CMKs as your keys. In this way, you can gain more control over the keys. For example, you cannot immediately delete the key materials generated by KMS, but you can immediately delete the key materials imported to KMS. Exercise caution when using the BYOK feature because it incurs extra management costs. For more information, see *Import key materials*.

Encrypt data in Alibaba Cloud services

The encryption design varies with Alibaba Cloud services based on their business forms and customer needs. Generally, a key hierarchy consisting of at least two

layers is used, and business data is encrypted by using the envelope encryption mechanism.

The first layer is the CMK in KMS, and the second layer is the data key (DK). The CMK is used to encrypt and decrypt the DK, while the DK is used to encrypt and decrypt your business data. When storing your business data to a persistent storage medium, an Alibaba Cloud service writes both the ciphertext of the DK (encrypted by KMS using the CMK) and the ciphertext of the business data (encrypted by the Alibaba Cloud service using the DK) to this medium. This mechanism is known as envelope encryption. The ciphertext of the DK and the ciphertext of the business data are packaged together in an "envelope." When reading the encrypted data, the Alibaba Cloud service reads both the ciphertext of the DK and the ciphertext of the business data. The Alibaba Cloud service must first decrypt the ciphertext of the DK before using the decrypted DK to decrypt the ciphertext of the business data.



In envelope encryption, the CMK is protected by the key management infrastructure of KMS. The Alibaba Cloud service must be authorized to use the CMK to generate the DK for encrypting business data or decrypt the ciphertext of the DK for decrypting business data. The plaintext of the DK never leaves the memory of the host where the Alibaba Cloud service instance resides. That is, the DK will not be stored in plaintext in any persistent storage medium.

7.2 Alibaba Cloud services that support integration with KMS

This topic describes the Alibaba Cloud services that support integration with Key Management Service (KMS). These Alibaba Cloud services can use server-managed keys or user-managed keys, including the keys uploaded through the Bring Your Own Key (BYOK) feature, to encrypt your data.

The integration with KMS allows you to protect your data stored in Alibaba Cloud services at low costs, provides security boundaries for business data, and enhances Alibaba Cloud's capabilities in protecting business security. Alibaba Cloud services can encrypt not only the business data that you can directly access, but also the business data that you can only indirectly access.

ECS

By default, the *disk encryption* feature of Elastic Compute Service (ECS) uses the service key to encrypt your data. This feature also supports user-managed keys. Each disk has its own customer master key (CMK) and data key (DK), and uses the envelope encryption mechanism to encrypt your data.

An ECS instance automatically encrypts the data transmitted to an encrypted disk and decrypts the data read from the disk. Data is encrypted and decrypted on the host where the ECS instance resides. During encryption and decryption, the performance of the disk hardly degrades.

After an encrypted disk is created and attached to an ECS instance, the ECS instance encrypts the following data:

- **Static data stored on the disk.**
- **Data transmitted between the disk and the ECS instance. Data in the operating system of the ECS instance is not encrypted.**
- **All snapshots created from the encrypted disk. These snapshots are called encrypted snapshots.**

Note:
Container Service can also use the disk encryption feature to encrypt your data.

OSS

Object Storage Service (OSS) uses the server-side encryption (SSE) feature to encrypt uploaded data. When you upload data to OSS, OSS encrypts the data and

stores the encrypted data in persistent storage. When you download data from OSS , OSS automatically decrypts the encrypted data and returns the decrypted data to you. In addition, OSS declares that the data has been encrypted on the server through a header in the returned HTTP response.

OSS can use an encryption system dedicated to OSS to implement the SSE feature. In this case, this feature is called SSE-OSS. The keys used in this encryption system are managed by OSS. Therefore, you cannot use ActionTrail to audit the use of these keys.

OSS can also use KMS to implement the SSE feature. In this case, this feature is called SSE-KMS. OSS uses the service key or user-managed keys to encrypt your data. OSS allows you to configure a default CMK for each bucket or specify the CMK to use when uploading an object.

For more information, see *#unique_40* and *SDK reference* of OSS.

ApsaraDB

- ApsaraDB for RDS

  ApsaraDB for Relational Database Service (RDS) supports the following methods for encrypting data:

  - Disk encryption

    For disks used by RDS instances, Alibaba Cloud provides the disk encryption feature for free, which encrypts the disks based on block storage. The keys used for disk encryption are encrypted and stored in KMS. RDS reads the keys only when starting or migrating instances.

  - TDE

    RDS for MySQL and RDS for SQL Server support transparent data encryption (TDE). The keys used for TDE are encrypted and stored in KMS. RDS reads the keys only when starting or migrating instances. After TDE is enabled for an RDS instance, you can specify the database or table to be encrypted. The data of the specified database or table is first encrypted and then written to the destination device such as a hard disk drive (HDD), solid-state drive (SSD), or Peripheral Component Interconnect Express (PCIe) card, or to any service

such as OSS or Archive Storage. All data files and backups of the RDS instance are stored in ciphertext.

For more information, see the following topics:

- RDS for MySQL: *#unique_42*
- RDS for SQL Server: *#unique_43*

- ApsaraDB for MongoDB

  The encryption method for ApsaraDB for MongoDB is similar to that for ApsaraDB for RDS. For more information, see *#unique_44*.

ACM

Application Configuration Management (ACM) integrates KMS to encrypt application configurations. This ensures the security of sensitive configurations, such as data sources, tokens, usernames, and passwords, and reduces the risk of configuration leakage. ACM can use KMS in either of the following ways:

- Encrypt data in KMS

  ACM calls KMS API to transmit configurations to KMS and encrypt the configurations with the specified CMK.
- Encrypt data in ACM by using the envelope encryption mechanism

  ACM uses a DK to encrypt configurations in ACM and calls KMS API to encrypt the DK with the specified CMK.

For more information, see *#unique_45*.

NAS

By default, Network Attached Storage (NAS) uses the service key to encrypt your data. Each volume has its own CMK and DK, and uses the envelope encryption mechanism to encrypt your data. Currently, only the NAS service key can be used as the CMK. User-managed keys will be supported in the future.

Table Store

By default, Table Store uses the service key to encrypt your data. Table Store also supports user-managed keys. Each table has its own CMK and DK, and uses the envelope encryption mechanism to encrypt your data.

MaxCompute

MaxCompute uses the service key as the CMK to encrypt your data.

CSG

Cloud Storage Gateway (CSG) supports OSS-based data encryption. For more information, see *#unique_46*.

ApsaraVideo for Media Processing

ApsaraVideo for Media Processing supports both Alibaba Cloud proprietary cryptography and HTTP Live Streaming (HLS) encryption. Both methods can integrate KMS to protect video content.

ApsaraVideo VOD

ApsaraVideo VOD supports *Alibaba Cloud video encryption* **and** *HLS encryption*. **Both methods can integrate KMS to protect video content.**

Web+

Web App Service (Web+) integrates KMS to encrypt sensitive configuration data.