

Alibaba Cloud KeyManagementService

User Guide

Issue: 20190916

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, consequential, 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 content of the Alibaba Cloud website, including but not limited to works, products, images, archives, information, materials, website architecture, website graphic layout, and webpage design, 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 the Alibaba Cloud website, product programs, or content 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.

Generic conventions

Table -1: Style conventions

Style	Description	Example
	This warning information 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.
	This warning information 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 restore business.
	This indicates warning information, supplementary instructions, and other content that the user must understand.	 Notice: Take the necessary precautions to save exported data containing sensitive information.
	This indicates supplemental instructions, best practices, tips, and other content that is good to know for the user.	 Note: You can use Ctrl + A to select all files.
>	Multi-level menu cascade.	Settings > Network > Set network type
Bold	It is used for buttons, menus, page names, and other UI elements.	Click OK.
<code>Courier</code> font	It is used for commands.	Run the <code>cd / d C :/ windows</code> command to enter the Windows system folder.
<i>Italics</i>	It is used for parameters and variables.	<code>bae log list --instanceid Instance_ID</code>
[] or [a b]	It indicates that it is an optional value, and only one item can be selected.	<code>ipconfig [-all -t]</code>

Style	Description	Example
<code>{}</code> or <code>{a b}</code>	It indicates that it is a required value, and only one item can be selected.	<code>switch {stand slave}</code>

Contents

Legal disclaimer.....	I
Generic conventions.....	I
1 Use RAM to authorize KMS resources.....	1
2 Use ActionTrail to record KMS events.....	5
3 Import key materials.....	6
4 Managed HSM (preview).....	13
4.1 Overview.....	13
4.2 Using Managed HSM.....	15

1 Use RAM to authorize KMS resources

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

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 [../SP_65/DNRAM11820297/EN-US_TP_12331.dita#concept_oyr_zzv_tdb](#) and [#unique_4](#).

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 as access control for different API operations. They must be in the `kms : ${ api - name }` format.



Note:

Access control is not required for the `DescribeRegions` operation. The `DescribeRegions` operation can be called by an Alibaba Cloud account, a RAM user, or a RAM role and authenticated directly.

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

KMS API operation	Action	Resource type
ListKeys	kms:ListKeys	Key container
CreateKey	kms:CreateKey	Key container
DescribeKey	kms:DescribeKey	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
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

Policy conditions in KMS

You can add conditions to RAM policies to control their 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_5](#).

Common RAM policy examples

- The RAM policy allowing users to access all KMS resources

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

- The 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": [
        "*"
      ]
    }
  ]
}
```

- The 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_7](#).

For event details, see [#unique_8](#).

3 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_10](#) 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_11](#) 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_12](#) 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_13 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 -- Descriptio n  
" 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_15](#) operation. The key material import parameters include a public key used to encrypt the key material and an import token.

Example

```
aliyuncli kms GetParamet ersForImpo rt -- KeyId 1339cb7d  
- 54d3 - 47e0 - b595 - c7d3dba82b 6f -- WrappingAl gorithm  
RSAES_OAEP _SHA_1 -- WrappingKe ySpec 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_PKCS1_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_16](#) parameters.

Example

```
aliyuncli kms ImportKeyMaterial -- KeyId 1339cb7d - 54d3 - 47e0 - b595 - c7d3dba82b 6f -- 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_12](#) 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.l . bin  32
openssl enc - d - base64 - A - in    PublicKey_ base64 .
txt - out    PublicKey . bin
openssl rsautl - encrypt - in    KeyMaterial.l . bin - oaep
- inkey    PublicKey . bin - keyform    DER - pubin - out
EncryptedKey eyMaterial . bin
openssl enc - e - base64 - A - in    EncryptedKey eyMaterial
. bin - out    EncryptedKey eyMaterial _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 CreateKeyR
equest () ;
        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 importToken = paramResponse .
getImportToken ();
            String encryptPublicKey = paramResponse .
getPublicKey ();
            // Perform Base64 decoding on the public
            encryption key
            byte [] publicKeyDer = DatatypeConverter .
parseBase64Binary ( encryptPublicKey );
            // Parse the RSA public key
            KeyFactory keyFact = KeyFactory . getInstanc e
(" RSA ");
            X509EncodedKeySpec spec = new X509Encode
dKeySpec ( publicKeyDer );
            PublicKey publicKey = keyFact . generatePu blic
( spec );
            // Encrypt the key material
            Cipher oaepFromAlgo = Cipher . getInstanc e ("
RSA / ECB / OAEPWithSHA-1AndMGF1Padding ");
            String hashFunc = " SHA - 256 ";
            OAEPParameterSpec oaepParams = new
OAEPParameterSpec ( hashFunc , " MGF1 " , new MGF1Parame terSpec
( 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 = DatatypeCo
nverter . printBase64Binary ( cipherDer );
            // Import the key material
            Long expireTime stamp = 1546272000 L ; // Unix
timestamp , precise to the second , 0 indicates no
expiration
            kmsclient . importKeyMaterial ( keyId ,
encryptedKeyMaterial , expireTime stamp );

```

```
    } catch ( Exception e ) {  
        //... Omitted  
    }  
}
```

4 Managed HSM (preview)

4.1 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 SS 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.

4.2 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 [#unique_20/unique_20_Connect_42_section_9br_g7q_yb4](#).

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 -- Protection Level HSM -- Descriptio n  
" 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": "1111222233 33",
    "Arn": "acs:kms:cn-hongkong:1111222233 33:key/1234abcd-12ab-34cd-56ef-12345678 ****",
    "Origin": "Aliyun_KMS",
    "MaterialExpirationTime": "",
    "ProtectionLevel": "HSM"
  },
  "RequestId": "8eaeaa8b-4491-4f1e-a51e-f95a4e5462 0c"
}
```

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.