ALIBABA CLOUD

Alibaba Cloud

Key Management Service
SDK Reference

Document Version: 20201130

Alibaba Cloud

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 because of product version upgrade, adjustment, or other reasons. Alibaba Cloud reserves the right to modify the content of this document without notice and an updated version of this document will be released through Alibaba Cloud-authorized channels from time to time. You should 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 this document based on the "status quo", "being defective", and "existing functions" of its products and services. 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 take legal responsibility for any errors or lost profits incurred by any organization, company, or individual arising from download, use, or trust in this document. Alibaba Cloud shall not, under any circumstances, take responsibility for any indirect, consequential, punitive, contingent, 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 directly contact Alibaba Cloud for any errors of this document.

Document conventions

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

Table of Contents

1.SDK overview

Alibaba Cloud provides KMS SDKs for Java, Python, PHP, and .NET.

The following table lists the download links and quick start guides of the SDKs for these programming languages. For more information about SDKs, visit Alibaba Cloud Open Platform.

Alibaba Cloud SDK	KMS SDK	Documentation
Alibaba Cloud SDK for Java	Alibaba Cloud KMS SDK for Java	Quick start
Alibaba Cloud SDK for Python	Alibaba Cloud KMS SDK for Python	Quick start
Alibaba Cloud SDK for PHP	Alibaba Cloud KMS SDK for PHP	Quick start
Alibaba Cloud SDK for .NET	Alibaba Cloud KMS SDK for .NET	Quick start

2.Code samples of SDK for Java

This topic provides some code samples of KMS SDK for Java.

Background information

- You can visit the open source code repository to view code samples for more programming languages and scenarios. Your valuable comments or code examples would be greatly appreciated.
- For more information about KMS API operations, see List of operations by function.

Preparations

1. Obtain the dependency declaration of KMS SDK for Java. For information about the required SDK version, see SDK overview. Sample code:

```
<dependency>
  <groupId>com.aliyun</groupId>
  <artifactId>aliyun-java-sdk-core</artifactId>
  <version>4.5.2</version>
</dependency>
<dependency>
  <groupId>com.aliyun</groupId>
  <artifactId>aliyun-java-sdk-kms</artifactId>
  <version>2.10.1</version>
</dependency>
```

2. Obtain the endpoints to access KMS based on the region where you use KMS. For more information, see Endpoints.

> **Note** You can access KMS over the Internet or a VPC endpoint. When you use SDK for Java, you must specify parameters based on your business requirements:
>
> - If you only specify a region ID, SDK for Java uses the public endpoint of the specified region by default.
> - If you want to access KMS over a VPC endpoint, you must specify this endpoint.

Code sample: KmsSample

1. Create encapsulated class KmsClient:

```
package com.aliyun.kms.samples;
import java.util.*;
import com.aliyuncs.DefaultAcsClient;
import com.aliyuncs.exceptions.ClientException;
import com.aliyuncs.http. *;
//Current KMS SDK version:2016-01-20
import com.aliyuncs.kms.model.v20160120.*;
```

```
import com.aliyuncs.kms.model.v20160120.ListKeysResponse.Key;
import com.aliyuncs.profile.*;
public class KmsClient
{
  private DefaultAcsClient kmsClient;
  /**
   * Create KmsClient. You need only to specify a region ID. SDK for Java automatically configures the public endpoint of the region.
   */
  public static KmsClient getClientForPublicEndpoint(String regionId, String accessKeyId, String accessKeySecret) {
    /**
     * Construct an Aliyun Client:
     * Set RegionId, AccessKeyId and AccessKeySecret
     */
    IClientProfile profile = DefaultProfile.getProfile(regionId, accessKeyId, accessKeySecret);
    DefaultAcsClient client = new DefaultAcsClient(profile);
    return new KmsClient(client);
  }
  /**
   * Create KmsClient. You can specify a custom endpoint. In most cases, a VPC endpoint is specified.
   */
  public static KmsClient getClientForVpcEndpoint(String regionId, String accessKeyId, String accessKeySecret, String endpoint) {
    //Specify a custom endpoint.
    DefaultProfile.addEndpoint(regionId, "kms", endpoint);
    IClientProfile profile = DefaultProfile.getProfile(regionId, accessKeyId, accessKeySecret);
    HttpClientConfig clientConfig = HttpClientConfig.getDefault();
    profile.setHttpClientConfig(clientConfig);
    DefaultAcsClient client = new DefaultAcsClient(profile);
    return new KmsClient(client);
  }
  private KmsClient(DefaultAcsClient acsClient) {
    this.kmsClient = acsClient;
  }
  public CreateKeyResponse CreateKey(String keyDesc, String keyUsage) throws ClientException {
    final CreateKeyRequest ckReq = new CreateKeyRequest();
    ckReq.setProtocol(ProtocolType.HTTPS);
    ckReq.setAcceptFormat(FormatType.JSON);
    ckReq.setMethod(MethodType.POST);
    ckReq.setDescription(keyDesc);
```

```
    ckReq.setKeyUsage(keyUsage);
    final CreateKeyResponse response = kmsClient.getAcsResponse(ckReq);
    return response;
  }
  public DescribeKeyResponse DescribeKey(String keyId) throws ClientException {
    final DescribeKeyRequest decKeyReq = new DescribeKeyRequest();
    decKeyReq.setProtocol(ProtocolType.HTTPS);
    decKeyReq.setAcceptFormat(FormatType.JSON);
    decKeyReq.setMethod(MethodType.POST);
    decKeyReq.setKeyId(keyId);
    final DescribeKeyResponse decKeyRes = kmsClient.getAcsResponse(decKeyReq);
    return decKeyRes;
  }
  public ListKeysResponse ListKey(int pageNumber, int pageSize) throws ClientException {
    final ListKeysRequest listKeysReq = new ListKeysRequest();
    listKeysReq.setProtocol(ProtocolType.HTTPS);
    listKeysReq.setAcceptFormat(FormatType.JSON);
    listKeysReq.setMethod(MethodType.POST);
    listKeysReq.setPageNumber(pageNumber);
    listKeysReq.setPageSize(pageSize);
    final ListKeysResponse listKeysRes = kmsClient.getAcsResponse(listKeysReq);
    return listKeysRes;
  }
  public GenerateDataKeyResponse GenerateDataKey(String keyId, String keyDesc, int numOfBytes) th
rows ClientException {
    final  GenerateDataKeyRequest genDKReq = new GenerateDataKeyRequest();
    genDKReq.setProtocol(ProtocolType.HTTPS);
    genDKReq.setAcceptFormat(FormatType.JSON);
    genDKReq.setMethod(MethodType.POST);
    /**
     * Set parameter according to KMS openAPI document:
     * 1.KeyId
     * 2.KeyDescription
     * 3.NumberOfBytes
     */
    genDKReq.setKeySpec(keyDesc);
    genDKReq.setKeyId(keyId);
    genDKReq.setNumberOfBytes(numOfBytes);
    final GenerateDataKeyResponse genDKRes = kmsClient.getAcsResponse(genDKReq);
    return genDKRes;
  }
```

```
  public EncryptResponse Encrypt(String keyId, String plainText) throws ClientException {
    final EncryptRequest encReq = new EncryptRequest();
    encReq.setProtocol(ProtocolType.HTTPS);
    encReq.setAcceptFormat(FormatType.JSON);
    encReq.setMethod(MethodType.POST);
    encReq.setKeyId(keyId);
    encReq.setPlaintext(plainText);
    final EncryptResponse encResponse = kmsClient.getAcsResponse(encReq);
    return encResponse;
  }
  public DecryptResponse Decrypt(String cipherBlob) throws ClientException {
    final DecryptRequest decReq = new DecryptRequest();
    decReq.setProtocol(ProtocolType.HTTPS);
    decReq.setAcceptFormat(FormatType.JSON);
    decReq.setMethod(MethodType.POST);
    decReq.setCiphertextBlob(cipherBlob);
    final DecryptResponse decResponse = kmsClient.getAcsResponse(decReq);
    return decResponse;
  }
}
```

2. Use KmsClient to call KMS API operations to enumerate keys, and encrypt and decrypt data.

> **Note**
>
> - In this code sample, your Alibaba Cloud account has at least one KMS CMK in the China (Hangzhou) region.
> - The `KmsClient.getClientForPublicEndpoint` method is used to initialize KmsClient to access KMS over the public endpoint.
> - The `KmsClient.getClientForVpcEndpoint` method is used to initialize KmsClient to access KMS over the VPC endpoint.

```
package com.aliyun.kms.samples;
import com.aliyuncs.exceptions.ClientException;
import com.aliyuncs.exceptions.ServerException;
import com.google.gson.Gson;
import java.util.*;
import com.aliyuncs.kms.model.v20160120.*;
import com.aliyuncs.kms.model.v20160120.ListKeysResponse.Key;
public class KmsSample {
 public static void main(String[] args) {
    String accessKeyId = System.getenv("ACCESS_KEY_ID");
```

```
    String accessKeySecret = System.getenv("ACCESS_KEY_SECRET");
    KmsClient kmsClient = KmsClient.getClientForPublicEndpoint("cn-hangzhou", accessKeyId, accessKeySecret);
    //KmsClient kmsClient = KmsClient.getClientForVpcEndpoint("cn-hangzhou-vpc", accessKeyId, accessKeySecret, "kms-vpc.cn-hangzhou.aliyuncs.com");
    String keyId = null;
    String plainText = "hello world";
    String cipherBlob = null;
  /*List all MasterKeys in your account*/
    try {
      final ListKeysResponse listKeysRes = kmsClient.ListKey(1, 100);
      /**
       * Parse response and do more further
       */
      System.out.println("TotalCount: " + listKeysRes.getTotalCount());
      System.out.println("PageNumber: " + listKeysRes.getPageNumber());
      System.out.println("PageSize: " + listKeysRes.getPageSize());
      List<Key> keys = listKeysRes.getKeys();
      Iterator<Key> iterator = keys.iterator();
      while (iterator.hasNext()) {
        keyId = iterator.next().getKeyId();
        System.out.println("KeyId: " + keyId);
      }
      System.out.println("List All MasterKeys success! \n");
    } catch (ClientException eResponse) {
      System.out.println("Failed.") ;
      System.out.println("Error code: " + eResponse.getErrCode());
      System.out.println("Error message: " + eResponse.getErrMsg());
    }
    /*Describe the Key */
    try {
      final DescribeKeyResponse decKeyRes = kmsClient.DescribeKey(keyId);
      /**
       * Parse response and do more further
       */
      System.out.println("DescribeKey Response: ");
      DescribeKeyResponse.KeyMetadata meta = decKeyRes.getKeyMetadata();
      System.out.println("KeyId: " + meta.getKeyId());
      System.out.println("Description: " + meta.getDescription());
      System.out.println("KeyState: " + meta.getKeyState());
```

```
        System.out.println("KeyUsage: " + meta.getKeyUsage());
        System.out.println("========================================");
        System.out.println("Describe the MasterKey success!") ;
        System.out.println("========================================\n");
    } catch (ClientException eResponse) {
        System.out.println("Failed.") ;
        System.out.println("Error code: " + eResponse.getErrCode());
        System.out.println("Error message: " + eResponse.getErrMsg());
    }
    /*Generate DataKey*/
    /**
     * Request and got response
     */
    try {
        final GenerateDataKeyResponse genDKResponse = kmsClient.GenerateDataKey(keyId, "AES_256",
64);
        /**
         * Parse response and do more further
         */
        System.out.println("CiphertextBlob: " + genDKResponse.getCiphertextBlob());
        System.out.println("KeyId: " + genDKResponse.getKeyId());
        System.out.println("Plaintext: " + genDKResponse.getPlaintext());
        System.out.println("========================================");
        System.out.println("Generate DataKey success!") ;
        System.out.println("========================================\n");
    } catch (ClientException eResponse) {
        System.out.println("Failed.") ;
        System.out.println("Error code: " + eResponse.getErrCode());
        System.out.println("Error message: " + eResponse.getErrMsg());
    }
    /**
     * Encrypt the plain text and got a cipher one
     */
    try {
        EncryptResponse encResponse = kmsClient.Encrypt(keyId, plainText);
        cipherBlob = encResponse.getCiphertextBlob();
        System.out.println("CiphertextBlob: " + cipherBlob);
        System.out.println("KeyId: " + encResponse.getKeyId());
        System.out.println("========================================");
        System.out.println("Encrypt the plain text success!") ;
        System.out.println("========================================\n");
```

```
    } catch (ClientException eResponse) {
      System.out.println("Failed.") ;
      System.out.println("Error code: " + eResponse.getErrCode());
      System.out.println("Error message: " + eResponse.getErrMsg());
    }
    /**
     * Decrypt the cipher text and verify result with original plain text.
     */
    try {
      DecryptResponse decResponse = kmsClient.Decrypt(cipherBlob);
      System.out.println("Plaintext: " + decResponse.getPlaintext());
      String verifyPlainText = decResponse.getPlaintext();
      int isMatch = verifyPlainText.compareTo(plainText);
      System.out.println("KeyId: " + decResponse.getKeyId());
      System.out.println("========================================");
      System.out.printf("Decrypt the cipher text success, result " + (isMatch == 0 ? "match" : "mismatch"
+ "\n"));
      System.out.println("========================================\n");
    } catch (ClientException eResponse) {
      System.out.println("Failed.") ;
      System.out.println("Error code: " + eResponse.getErrCode());
      System.out.println("Error message: " + eResponse.getErrMsg());
    }
  }
}
```