Key Management Service

SDK Reference

SDK Reference

Create an access key

You can create an access key on Access key management console.

SDK

Currently, Alibaba Cloud provides SDKs in four languages Java, Python, PHP, and C#, of which the links are listed below:

- Java
- Python
- PHP
- C#

Java SDK sample codes

Sample codes

```
package com.alibaba.samples;

import java.util.*;
import java.util.List;

import com.aliyuncs.DefaultAcsClient;
import com.aliyuncs.exceptions.ClientException;
import com.aliyuncs.http.FormatType;
import com.aliyuncs.http.MethodType;
import com.aliyuncs.http.ProtocolType;

//Current KMS SDK version:2016-01-20
import com.aliyuncs.kms.model.v20160120.CreateKeyRequest;
import com.aliyuncs.kms.model.v20160120.CreateKeyResponse;
import com.aliyuncs.kms.model.v20160120.DecryptRequest;
import com.aliyuncs.kms.model.v20160120.DecryptResponse;
```

```
import com.aliyuncs.kms.model.v20160120.DescribeKeyRequest;
import com.aliyuncs.kms.model.v20160120.DescribeKeyResponse;
import com.aliyuncs.kms.model.v20160120.EncryptRequest;
import com.aliyuncs.kms.model.v20160120.EncryptResponse;
import com.aliyuncs.kms.model.v20160120.GenerateDataKeyRequest;
import com.aliyuncs.kms.model.v20160120.GenerateDataKeyResponse;
import com.aliyuncs.kms.model.v20160120.ListKeysRequest;
import com.aliyuncs.kms.model.v20160120.ListKeysResponse;
import com.aliyuncs.kms.model.v20160120.ListKeysResponse.Key;
import com.aliyuncs.profile.DefaultProfile;
import com.aliyuncs.profile.IClientProfile;

public class kmsSample
{
static DefaultAcsClient kmsClient;

private static DefaultAcsClient kmsClient(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 client;
}

private static 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;
}

private static 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;
}

private static 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;
}

private static GenerateDataKeyResponse GenerateDataKey(String keyId, String keyDesc, int numOfBytes) throws
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;
}

private static 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;
}


private static 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;
}

public static void main(String[] args) {
System.out.println("==========================================");
```

```
System.out.println("Getting Started with KMS Service");
System.out.println("========================================\n");

/**
* RegionId: "cn-hangzhou" and "ap-southeast-1", eg. "cn-hangzhou"
*/
String regionId = "cn-hangzhou";
String accessKeyId = "*** Provide your AccessKeyId ***";
String accessKeySecret = "*** Provide your AccessKeySecret ***";

kmsClient = kmsClient(regionId, accessKeyId, accessKeySecret);
String keyId = null;
String plainText = "hello world";
String cipherBlob = null;

// /*Create a Key*/
// try {
// final CreateKeyResponse response = CreateKey("testkey", "ENCRYPT/DECRYPT");
//
// /**
// * Parse response and do more further
// */
// System.out.println(response.getKeyMetadata());
// CreateKeyResponse.KeyMetadata meta = response.getKeyMetadata();
//
// System.out.println("CreateTime: " + meta.getCreationDate());
// System.out.println("Description: " + meta.getDescription());
// System.out.println("KeyId: " + meta.getKeyId());
// keyId = meta.getKeyId();
// System.out.println("KeyState: " + meta.getKeyState());
// System.out.println("KeyUsage: " + meta.getKeyUsage());
//
// System.out.println("==========================================");
// System.out.println("Create 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());
// }
//

/*List all MasterKeys in your account*/
try {
final ListKeysResponse listKeysRes = 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("==========================================");
System.out.println("List All MasterKeys success!\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());
}

/*Describe the Key */
try {
final DescribeKeyResponse decKeyRes = 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 = 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 = 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 = 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());
}
}
}
```

Access KMS in the VPC environment

You need to add a custom endpoint directing to the domain name in the VPC. You also need to specify to use this endpoint in later access to KMS.

```
DefaultProfile.addEndpoint("cn-hangzhou-vpc", "cn-hangzhou-vpc", "Kms", "kms-vpc.cn-hangzhou.aliyuncs.com");
//Add a custom endpoint.
```

For details about the KMS endpoint list, refer to KMS locations.