AWS Key Management Service

Developer Guide

AWS Key Management Service: Developer Guide

Table of Contents

What is the AWS Key Management Service?

The AWS Key Management Service (AWS KMS) is a managed service that makes it easy for you to create and control the encryption keys used to encrypt your data. AWS KMS is integrated with other AWS services including Amazon Elastic Block Store (Amazon EBS), Amazon Simple Storage Service (Amazon S3), Amazon Redshift, Amazon Elastic Transcoder, Amazon WorkMail, and Amazon Relational Database Service (Amazon RDS) to make it simple to encrypt your data with encryption keys that you manage. AWS KMS is also integrated with AWS CloudTrail to provide you with key usage logs to help meet your regulatory and compliance needs.

AWS KMS lets you create keys that can never be exported from the service and which can be used to encrypt and decrypt data based on policies you define.

You can perform the following management actions on keys by using AWS KMS:

- Create, describe, and list keys
- Enable and disable keys
- Set and retrieve key usage policies
- Create, delete, list, and update key aliases

With AWS KMS you can also perform the following cryptographic functions using keys:

- Encrypt, decrypt, and re-encrypt data
- Generate data keys that can be exported from the service in plaintext or which can be encrypted under a key that doesn't leave the service
- Generate random numbers suitable for cryptographic applications

By using AWS KMS, you gain more control over access to data you encrypt. You can use the key management and cryptographic features directly in your applications or through AWS services that are integrated with AWS KMS. Whether you are writing applications for AWS or using AWS services, AWS KMS enables you to maintain control over who can use your keys and gain access to your encrypted data.

AWS KMS is integrated with AWS CloudTrail, a service that delivers log files to an Amazon S3 bucket that you designate. By using CloudTrail you can monitor and investigate how and when your keys have been used and by whom.

To learn more about how the AWS Key Management Service uses cryptography and secures keys, see the AWS Key Management Service Cryptographic Details whitepaper.

Topics

Concepts

This section introduces the basic terms and concepts in the AWS Key Management Service and how they work together to help protect customer data.

Customer Master Keys

AWS KMS uses a type of key called a *customer master key* (CMK) to encrypt and decrypt data. CMKs are the fundamental resources that AWS KMS manages. CMKs can be either customer-managed keys or AWS-managed keys. They can be used inside of AWS KMS to encrypt or decrypt up to 4 kilobytes of data directly. They can also be used to encrypt generated *data keys* which are then used to encrypt or decrypt larger amounts of data outside of the service. CMKs can never leave AWS KMS unencrypted but data keys can.

There is one *AWS-managed key* for each account for each service that is integrated with AWS KMS. This key is referred to as the *default key* for the service under your account. This key is used to encrypt data keys used by AWS services to protect data when you don't specify a CMK while creating the encrypted resource. If you need more granular control, you can specify a customer-managed key. For example, if you choose to encrypt an Amazon EBS volume, you can specify the AWS-managed default EBS key for the account or a CMK you created within AWS KMS. The key you selected is then used to protect the data key used to encrypt the volume.

You can only create CMKs if you have the appropriate permissions. You can provide an alias (display name) and a description for the key and define which IAM users or roles within an account can manage and use the key. You can also choose to allow AWS accounts other than your own to use the key.

Data Keys

You use data keys to encrypt large data objects within your own application outside AWS KMS. When you call `GenerateDataKey`, AWS KMS returns a plaintext version of the key and ciphertext that contains the key encrypted under the specified CMK. AWS KMS tracks which CMK was used to encrypt the data key. You use the plaintext data key in your application to encrypt data, and you typically store the encrypted key alongside your encrypted data. Security best practices suggest that you should remove the plaintext key from memory as soon as is practical after use. To decrypt data in your application, pass the encrypted data key to the `Decrypt` function. AWS KMS uses the associated CMK to decrypt and retrieve your plaintext data key. Use the plaintext key to decrypt your data and then remove the key from memory.

Envelope Encryption

AWS KMS uses envelope encryption to protect data. AWS KMS creates a data key, encrypts it under a customer master key, and returns plaintext and encrypted versions of the data key to you. You use the plaintext key to encrypt data and store the encrypted key alongside the encrypted data. The key should be removed from memory as soon as is practical after use. You can retrieve a plaintext data key only if you have the encrypted data key and you have permission to use the corresponding master key.

Encryption Context

All AWS KMS cryptographic operations accept an optional key/value map of additional contextual information called an *encryption context.* The specified context must be the same for both the encrypt and decrypt operations or decryption will not succeed. The encryption context is logged, can be used for additional auditing, and is available as context in the AWS policy language for fine-grained policy-based authorization. For more information, see Encryption Context (p. 80).

Permissions on Keys

When you create a master key, you can define a resource-based policy for it. If you do not define a policy, AWS KMS creates a default one for you that delegates permissions to your account. The permissions on default keys for a service under your account are created automatically by AWS KMS and cannot be changed. After you create a key, you can use the AWS KMS API or the AWS Identity and Access Management console to edit the key policy, defining the users and accounts that can use the key and specifying who can manage the key. For more information, see Creating Keys (p. 7).

Auditing Key Usage

You can use AWS CloudTrail to audit key usage. CloudTrail creates log files that contain a history of AWS API calls and related events for your account. The log file includes calls made by using the AWS Management Console, AWS SDKs, command line tools, and higher-level AWS services. It also includes actions requested by users on keys. For an AWS-managed key and customer-managed keys, you can get log information about when the key was used, get information about when the key was used, the action that was called, the name of the user, the IP address from which the action request originated, and so on. For more information about CloudTrail, see the CloudTrail User Guide.

Key Management Infrastructure

Common practice in cryptography requires that encryption and decryption use a publicly available and peer-reviewed algorithm such as AES (Advanced Encryption Standard) and a secret key. One of the main problems with cryptography is that it's very hard to keep a key secret. This is typically the job of a key management infrastructure (KMI). AWS KMS operates the KMI for you. AWS KMS creates and securely stores the master keys. AWS KMS also creates data keys and encrypts them by using master keys.

Grants

A *grant* is a delegation mechanism that you can use to provide other AWS *principals* long-term permissions to use master keys. Grants are intended to allow asynchronous use of customer master keys when the duration of usage is not known up-front but is expected to be long term. The permissions associated with a grant are described when the grant is created. You can create a grant that allows a subset of permitted actions or that supports further delegation. Grants are properties of a master key. They require an explicit AWS principal and a list of the permitted actions. Grants are valid until revoked.

Grant Tokens

There may a slight delay for a grant created in AWS KMS to take effect throughout a region. If you need to mitigate this delay, a grant token is a type of identifier that is designed to let the permissions in the grant take effect immediately when passed with any of the following APIs:

- CreateGrant

- Decrypt
- Encrypt
- GenerateDataKey
- GenerateDataKeyWithoutPlaintext
- ReEncrypt

A grant token is not considered a secret. The grant token contains information about who it's for and therefore who can use it to cause a grant to take effect more quickly.

AWS Services Integrated with AWS KMS

This topic lists the services that that are currently integrated with the AWS Key Management Service.

Service Name	Default Key Alias	Encryption Context Syntax	Support Date	More Information
Storage and Content Delivery				
Amazon S3	aws/s3	"aws:s3:arn":"arn:aws:s3:*bucket_name*/*file_name*"	November 12, 2014	How Amazon Simple Storage Service Uses AWS KMS (p. 25)
Amazon EBS	aws/ebs	"aws:ebs:id": "vol-*2cfb133e*"	November 12, 2014	How Amazon Elastic Block Store (Amazon EBS) Uses AWS KMS (p. 27)
Database				
Amazon RDS	aws/rds	"aws:rds:db-id":"db-*CQYSMDP-BRZ7BPMH7Y3RTDG5QY*", "aws:ebs:id":"vol-*57d21d70*"	January 06, 2015	How Amazon Relational Database Service Uses AWS KMS (p. 29)
Amazon Redshift	aws/redshift	"aws:redshift:arn": "arn:aws:redshift:*region*:*account_ID*:cluster:*cluster_name*", "aws:redshift:createtime": "*20150206T1832Z*"	November 12, 2014	How Amazon Redshift Uses AWS KMS (p. 28)
Analytics				

Service Name	Default Key Alias	Encryption Context Syntax	Support Date	More Information
Amazon EMR	aws/s3	"aws:s3:arn":"arn:aws:s3:*bucket_name*/*file_name*"	March 25, 2015	How Amazon EMR Uses AWS KMS (p. 37)
Application Services				
Elastic Transcoder	aws/s3 *	"service" : "elastictranscoder.amazonaws.com", "KeyId" : "*ARN of the key associated with your pipeline*", "Plaintext" : "*BLOB that contains your AES key*"	November 24, 2014	How Elastic Transcoder Uses AWS KMS (p. 30)
Enterprise Applications				
Amazon WorkMail	aws/workmail	"aws:workmail:arn":"arn:aws:workmail:*region*:*account ID*:organization/*organization ID*"	January 28, 2015	How Amazon WorkMail Uses AWS KMS (p. 34)

* When you choose server-side encryption, Amazon S3 performs all file encryption and decryption on your behalf. For more information, see How Elastic Transcoder Uses AWS KMS (p. 30).

Pricing for AWS Key Management Service

You incur charges from AWS KMS when you create and store enabled keys in AWS KMS, and when you make AWS KMS API requests. For the exact charges, go to the AWS Key Management Service Pricing page on the AWS website.

Charges for Key Storage

You are charged for all customer master keys (CMKs) that you create and that are enabled. To avoid storage charges for a CMK, disable the key (p. 13). You are not charged for the storage of default CMKs that are created automatically by an AWS service integrated with AWS KMS such as Amazon Elastic Block Store (Amazon EBS), Amazon Relational Database Service (Amazon RDS), and others.

Charges for API Requests

You are charged for all AWS KMS API requests. Common requests include the usage of CMKs for encryption and decryption, but you are also charged for all management API requests such as CreateKey, ListKeys, GetKeyPolicy, etc.

In addition to direct API requests, you are also charged when any AWS service that is integrated with AWS KMS makes a request on your behalf. For example, you may use server-side encryption with AWS KMS (SEE-KMS) to upload data to Amazon S3. When you do, each PUT request results in a GenerateDataKey API request from Amazon S3 to AWS KMS on your behalf to create an object key that is used to encrypt the Amazon S3 object. In this case, you would be charged for each API request that Amazon S3 makes on your behalf.

The first 20,000 API requests of each calendar month fall into the free tier which means you are not charged for them. To avoid charges for AWS KMS API requests you can limit your total number of requests to 20,000 or fewer in a calendar month, or stop making API requests to the service and stop using AWS KMS-backed encryption with the AWS services that are integrated with AWS KMS.

Getting Started

The following topics discuss how to use AWS Key Management Service in the IAM console to create, view, edit, and enable customer master keys. You enable keys to make them usable. Keys are enabled by default when first created.

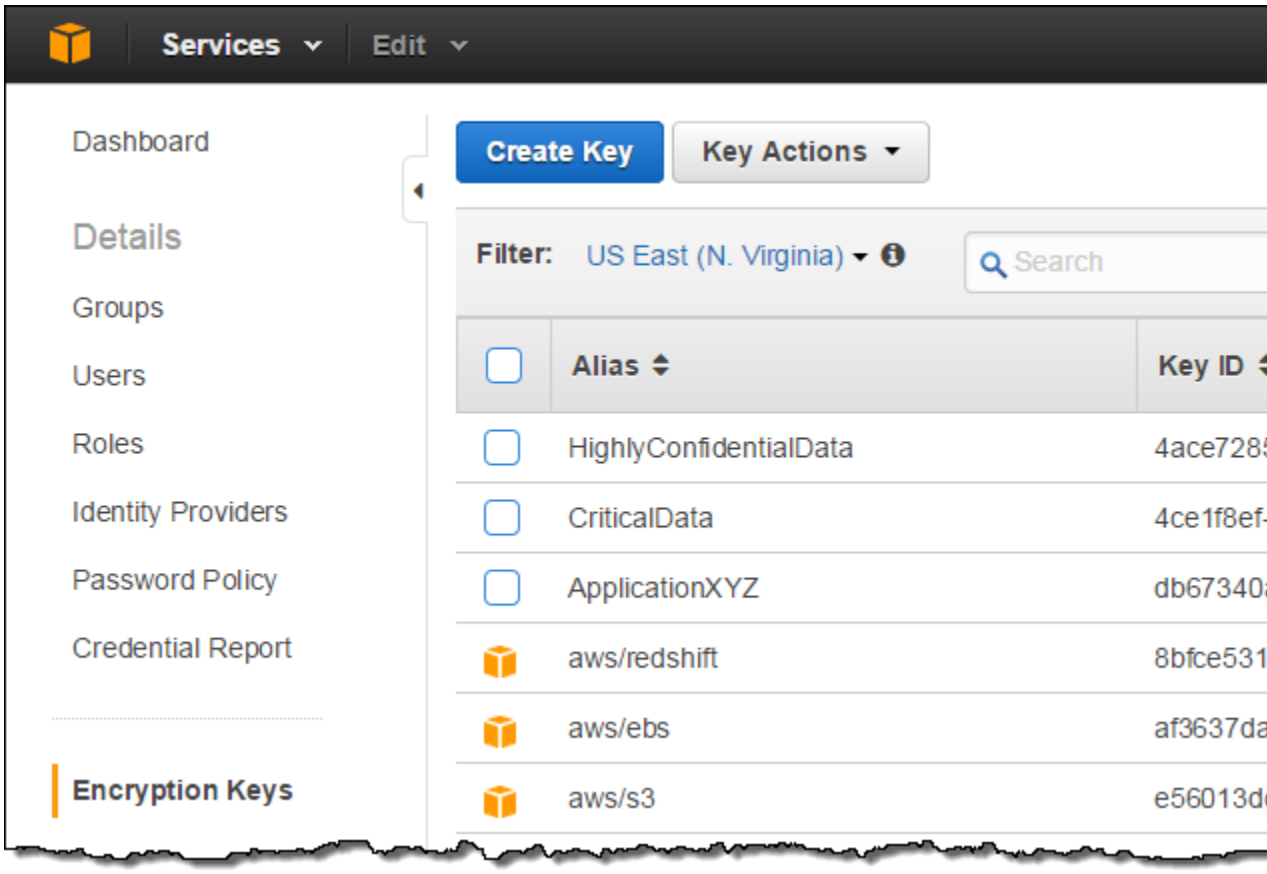
Topics

Creating Keys

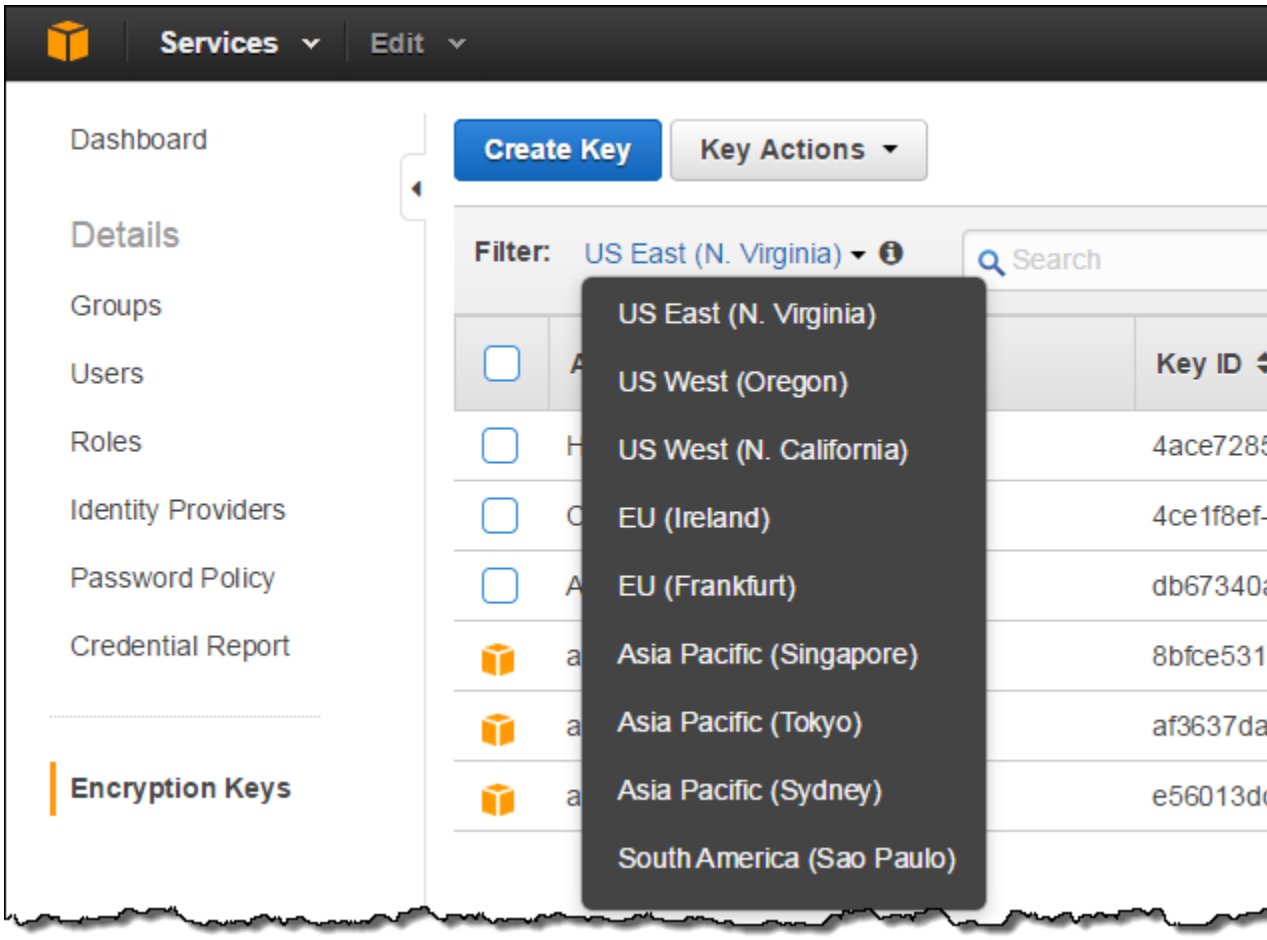
The following procedure describes how to use the IAM console to create a customer master key in your account. The console shows all of the customer master keys that have been created.

Sign into IAM

1. Sign in to the AWS Management Console and open the IAM console at https://console.aws.amazon.com/iam/. You will see a screen similar to the following.



2. In the navigation pane, click **Encryption Keys**.
3. Select a region from the region list.



4. Click **Create Key**.

Access to the AWS KMS section of the IAM console

Access to use the management console for AWS KMS requires appropriate permissions. Note that users with only IAM permissions will not have access to use the AWS KMS section of the IAM console. Users must have KMS permissions. The policy document below shows the permissions needed to use all features of AWS KMS in the IAM console:

-

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "Console Access",
      "Effect": "Allow",
      "Action": [
        "kms:CreateKey",
        "kms:CreateAlias",
        "kms:DeleteAlias",
        "kms:GenerateRandom",
        "kms:Describe*",
        "kms:Get*",
```

```
        "kms:List*",
        "iam:ListGroups",
        "iam:ListRoles",
        "iam:ListUsers"
      ],
      "Resource": [
        "*"
      ]
    }
  ]
}
```

Important
You cannot delete a key after you create it. You can only disable a key.

Enter an alias and a description

In this step you'll enter an alias and a description for the key. The alias is a display name that can be used to easily identify the key. We recommend, therefore, that you choose an alias that indicates the type of data the key will be used to encrypt or the application in which the key will be used. The alias must be between 1 and 32 characters long inclusive and must be an alphanumeric character, a dash, a forward slash (/), or an underscore. An alias must not begin with "aws". Aliases that begin with "aws" are reserved by Amazon Web Services to represent AWS-managed keys in your account.

The description can be up to 256 characters long and should tell users what the key will be used to encrypt.

1. Enter an alias for the key.
2. Enter a description for the key.
3. Click **Next Step**.

Specify who can manage the key

1. Define which IAM users or roles can administer the key via the AWS KMS API by placing a check beside the appropriate name. Additional IAM permissions may be needed for the user or role to manage the key from the console. See **Access to the AWS KMS section of the IAM console** above for information on which IAM permissions are needed.

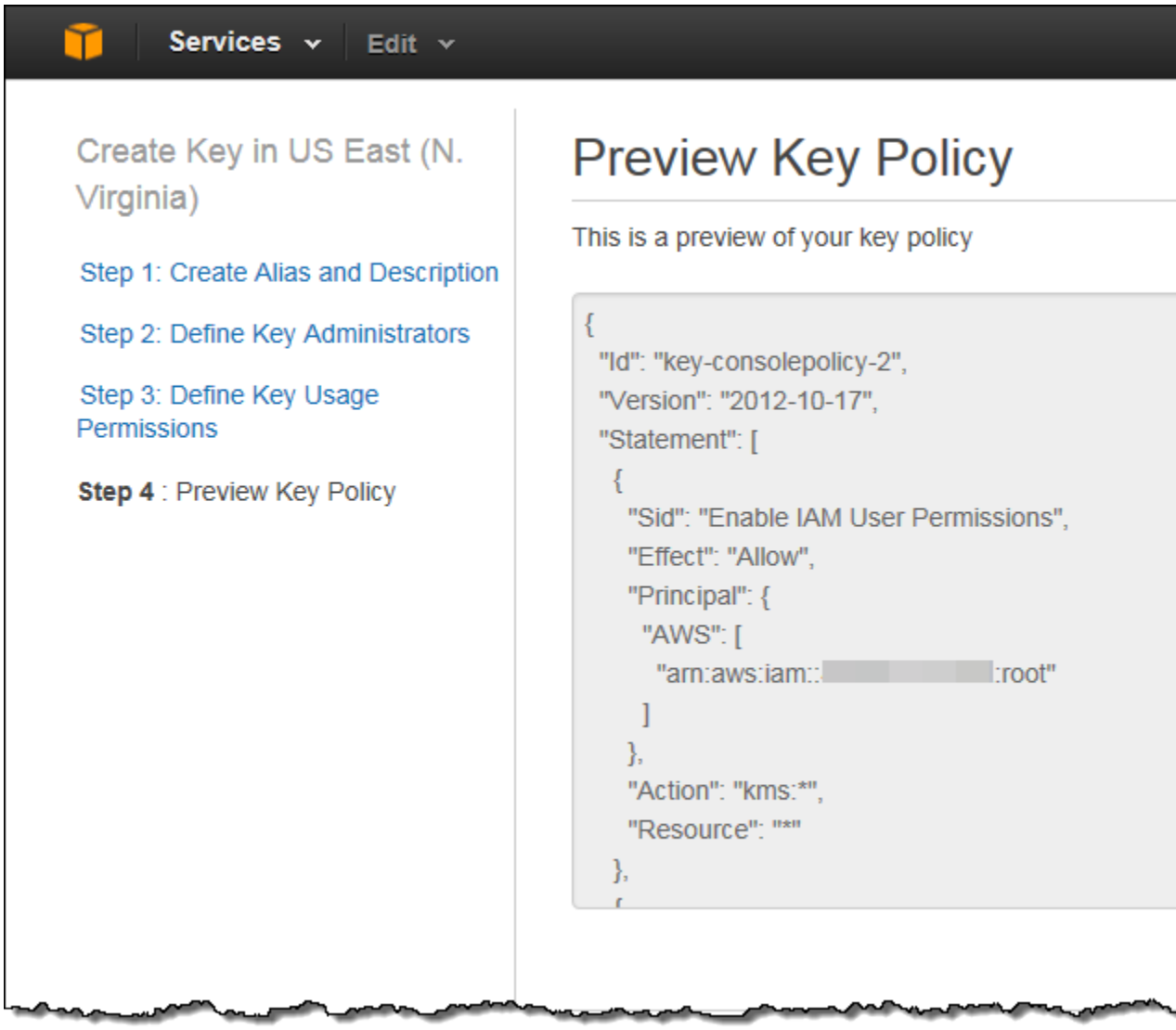
   Important
   Note that the root user for the account has permissions by default and that any users or roles that have appropriate AWS KMS management permissions attached to them can also manage the key.
2. Click **Next Step**.

Specify who can encrypt and decrypt

In this step you determine who can use the key to encrypt and decrypt data by identifying the users and roles to whom you want to grant this permission. Keep in mind, however, that any user or role that has appropriate AWS KMS permissions attached to them as resource -level policies for the key can also use the key to encrypt or decrypt data.

1. For the current account, place a check beside the appropriate names to define which IAM users and roles can use the key to encrypt and decrypt.

2. At the bottom of the page, you can also identify other accounts that can use this key to encrypt and decrypt data. Click **Add an External Account** and enter the IDs of the accounts that you want to be able to use this key. Note that the administrators of those accounts can further restrict access to the key by creating the necessary resource-level IAM policies for their users by including the ARN of the key. For more information, see Including External Accounts (p. 18).
3. Click **Finish** to create the key. A page similar to the following appears where you can preview the policy you just created.

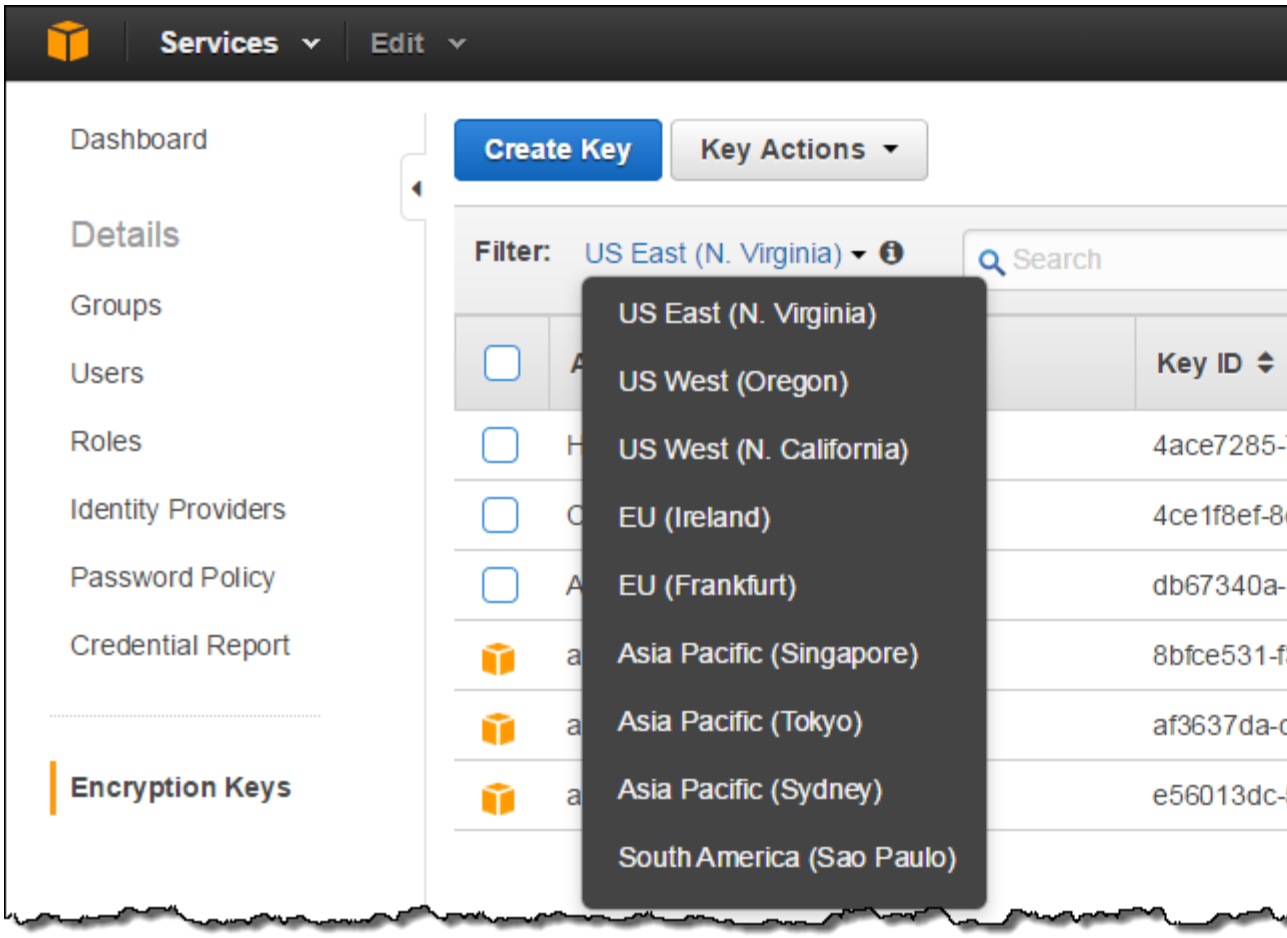


Viewing Keys

The following procedure describes how to use the IAM console to view master keys, including keys managed by customers and those managed by AWS.

1. Sign in to the AWS Management Console and open the IAM console at https://console.aws.amazon.com/iam/.

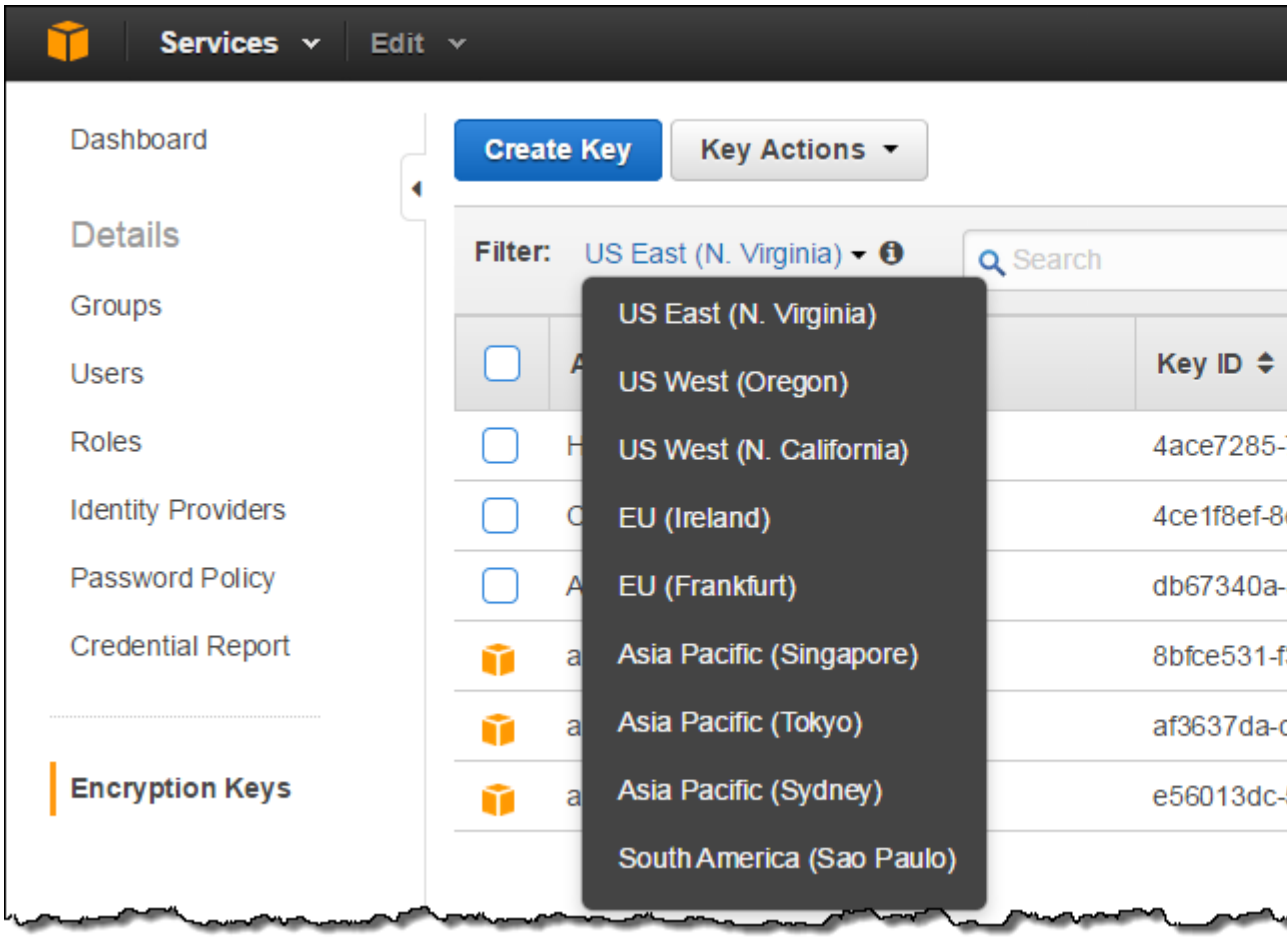
2. In the navigation pane, click **Encryption Keys**. The console shows all of the keys that have been created, including customer-managed and AWS-managed keys. The page lists an alias, description, and key ID for each key. AWS-managed keys, denoted by the orange AWS icon before the alias, are permanently enabled for use by services that support check box encryption. You cannot disable, edit, or delete them. You also cannot delete customer-managed keys you create.
3. Select the region by clicking the list shown in the following screenshot.



Editing Keys

The following procedure describes how to use the IAM console to edit customer master keys.

1. Sign in to the AWS Management Console and open the IAM console at https://console.aws.amazon.com/iam/.
2. In the navigation pane, click **Encryption Keys**. The console shows all of the keys that have been created, including customer-managed and AWS-managed keys. The page lists an alias, description, and key ID for each key. AWS-managed keys, denoted by the orange AWS icon before the alias, are permanently enabled for use by services that support check box encryption. You cannot disable, edit, or delete them. AWS will keep policies on AWS-managed keys updated for new service features as necessary.You also cannot delete customer-managed keys you create.
3. Select the region by clicking the drop down shown in the following screenshot.

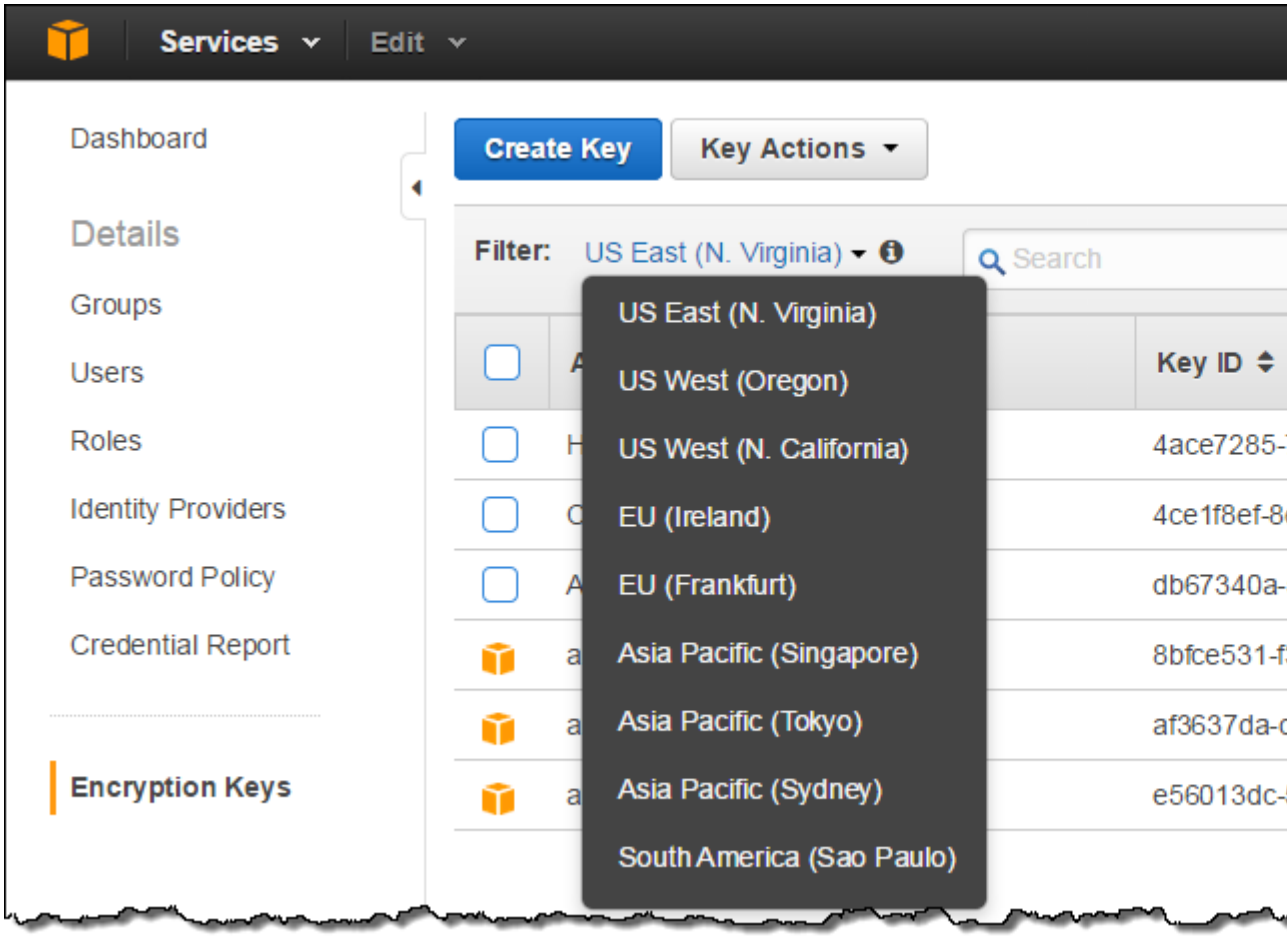


4. To view the key details, click the row for a key. The page following shows properties for the key, including the region, the Amazon Resource Name (ARN), the alias, and a description. The page also displays the IAM users and roles within your account and the specified region that can manage and use the key. You can use this page to edit the properties of the key.
5. You can add or remove key administrators. Clicking the **Add** button takes you to a new page that allows you to specify users or roles that can administer the key. To remove an administrator, place a check beside the appropriate name and click the **Remove** button.
6. You can specify the users or roles that can use the key for encryption and decryption. Clicking the **Add** button takes you to a new page that allows you to specify who can use the key. To remove a user or role, place a check beside the appropriate name and click the **Remove** button.
7. You can specify yearly key rotation by placing a check in the appropriate box.
8. Click **Save Changes** for each item for which you want to save changes. Click **Back to Encryption Keys** at the top of the page to return to the **Encryption Keys** summary page.

Enabling and Disabling Keys

The following procedure describes how to use the IAM console to enable and disable customer master keys. When you create a key, it is enabled by default. If you disable a key, nobody can use it. Therefore, disabling is a quick way to block access to a key. You cannot delete a key. Note that AWS-managed keys are permanently enabled for use by services that support check box encryption. You cannot disable them.

1. Sign in to the AWS Management Console and open the IAM console at https://console.aws.amazon.com/iam/.
2. In the navigation pane, click **Encryption Keys**. The console shows all of the keys that have been created, including customer-managed and AWS-managed keys. AWS-managed keys are permanently enabled for use by services that support check box encryption. You cannot disable them.
3. Select the region by clicking the list shown in the following screenshot.



4. Place a check beside the name of the key or keys you want to enable or disable. Note that the status column indicates whether a key is enabled or disabled.
5. Click **Key Actions**.
6. Choose **Enable** or **Disable** from the list.

Controlling Access to Your Keys

AWS Key Management Service provides two security mechanisms that you can use to protect your keys. *Key policies* are JSON documents that specify who can use a key, the operations the person can perform, and the conditions that govern use of the key. *Grants* are long-term alternate mechanisms to key policies that can also be used to grant permissions to keys.

In addition to these mechanisms, you can also use IAM user-based permissions as described in Overview of AWS IAM Permissions.

Topics

Key Policies

In order to define resource-based permissions, you need to attach policies to the keys. The policies let you specify who has access to the key and what actions they can perform. Best practice dictates that you define the users or roles that can use it when you create the key.

A key policy specifies who can manage a key and which user or role can encrypt or decrypt by using the key. Typically, most users set key policies by using the **Encryption Keys** section of the IAM console. You can also programmatically set a key policy by calling `PutKeyPolicy` and retrieve it by calling `GetKeyPolicy`. Key policies share a common syntax with the IAM policy specification. Conceptually, you can think of a policy as having the following JSON format:

```
{
    "Version": "2012-10-17",
    "Statement": [{
        "Effect": "effect",
        "Principal": "principal",
        "Action": "action",
        "Resource": "arn",
        "Condition": {
            "condition": {
                "key" : "value"
            }
```

```
        }
    ]
}
```

There can be multiple policy statements in a single document, and each statement can consist of up to five elements:

- `Effect:` This can be Allow or Deny.
- `Principal:` The AWS entity that is allowed or denied access to the key. This value is represented by the ARN of the principal.
- `Action:` Identifies the operations that can be performed by using the key. You can also use `NotAction` to specify an exception to a list of actions. This can often result in shorter policies than denying multiple actions.
- `Resource:` The key the action applies to. You can also specify `NotResource`. Note that you must use the full 32 character key ID when specifying the key as in the following example. Key aliases are not supported in policies either alone or as part of the ARN in the resource section of key policies when defining permissions on the key.

```
   "Resource": [
          "arn:aws:kms:us-east-1:0123456789012:key/12345678-1234-1234-1234-
123456789012"
   ]
```

- `Condition:` One or more conditions that must be met before a permitted action can be performed. With conditions you can use data coming in through the request context as additional criteria to evaluate the policy.

For more information about policy structure, see the AWS IAM Policy Reference.

Default Policy

If you do not specify a policy when you create a key, AWS Key Management Service creates the following default policy, which grants the AWS root account full access to all AWS KMS actions for the key the policy is attached to. This policy is also necessary so that you can create IAM user policies that give IAM users permission to use the key.

```
{
    "Id": "key-default",
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "Enable IAM User Permissions",
            "Effect": "Allow",
            "Principal": {"AWS":"arn:aws:iam::012345678901:root"},
            "Action": ["kms:*"],
            "Resource": "*"
        }
    ]
}
```

Console Key Management Policy

The IAM management console allows you to define more granular management permissions by creating the following policy that enables you to separate the principals that can manage the key from those that can use it. The first statement matches the default AWS KMS key policy. The second and third statements define which AWS principals can manage and use the key respectively. The fourth statement enables AWS Services that are integrated with AWS KMS to use the key on behalf of the specified principal. This statement permits AWS services to create and manage grants. The condition uses a context key that is set only for KMS calls made by AWS services on behalf of the customers.

```
{
  "Id": "key-consolepolicy-1",
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "Enable IAM User Permissions",
      "Effect": "Allow",
      "Principal": {"AWS": "arn:aws:iam::012345678901:root"},
      "Action": "kms:*",
      "Resource": "*"
    },
    {
      "Sid": "Allow access for Key Administrators",
      "Effect": "Allow",
      "Principal": {"AWS": "arn:aws:iam::012345678901:user/Alice"},
      "Action": [
        "kms:Create*",
        "kms:Describe*",
        "kms:Enable*",
        "kms:List*",
        "kms:Put*",
        "kms:Update*",
        "kms:Revoke*",
        "kms:Disable*",
        "kms:Get*",
        "kms:Delete*"
      ],
      "Resource": "*"
    },
    {
      "Sid": "Allow use of the key",
      "Effect": "Allow",
      "Principal": {"AWS": "arn:aws:iam::012345678901:user/Alice"},
      "Action": [
        "kms:Encrypt",
        "kms:Decrypt",
        "kms:ReEncrypt",
        "kms:GenerateDataKey*",
        "kms:DescribeKey"
      ],
      "Resource": "*"
    },
    {
      "Sid": "Allow attachment of persistent resources",
      "Effect": "Allow",
      "Principal": {"AWS": "arn:aws:iam::012345678901:user/Alice"},
```

```
      "Action": [
        "kms:CreateGrant",
        "kms:ListGrants",
        "kms:RevokeGrant"
      ],
      "Resource": "*",
      "Condition": {"Bool": {"kms:GrantIsForAWSResource": "true"}}
    }
  ]
}
```

Including External Accounts

When you use the console to give external accounts the ability to use a key, the account number is added to the policy in the following manner.

- The third statement in the following policy defines which AWS principals can use the key. Note that account number 123456789012 is the account that created the key and account number 109876543210 is an external account that you enabled to use the key during creation. For more information, see Creating Keys (p. 7)
- The fourth statement enables AWS Services that are integrated with AWS KMS to use the key on behalf of the specified principals. Two principals are specified —a user in the account that created the key and the root of an external account specified when the key was created.

```
{
  "Id": "key-consolepolicy-1",
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "Enable IAM User Permissions",
      "Effect": "Allow",
      "Principal": {"AWS": "012345678901"},
      "Action": "kms:*",
      "Resource": "*"
    },
    {
      "Sid": "Allow access for Key Administrators",
      "Effect": "Allow",
      "Principal": {"AWS": "arn:aws:iam::012345678901:user/Alice"},
      "Action": [
        "kms:Create*",
        "kms:Describe*",
        "kms:Enable*",
        "kms:List*",
        "kms:Put*",
        "kms:Update*",
        "kms:Revoke*",
        "kms:Disable*",
        "kms:Get*",
        "kms:Delete*"
      ],
      "Resource": "*"
    },
```

```
    {
      "Sid": "Allow use of the key",
      "Effect": "Allow",
      "Principal": {"AWS":
"arn:aws:iam::012345678901:user/Alice","arn:aws:iam::109876543210:root"},
      "Action": [
        "kms:Encrypt",
        "kms:Decrypt",
        "kms:ReEncrypt",
        "kms:GenerateDataKey*",
        "kms:DescribeKey"
      ],
      "Resource": "*"
    },
    {
      "Sid": "Allow attachment of persistent resources",
      "Effect": "Allow",
      "Principal": {"AWS": "arn:aws:iam::012345678901:user/Alice",
"arn:aws:iam::109876543210:root"},
      "Action": [
        "kms:CreateGrant",
        "kms:ListGrants",
        "kms:RevokeGrant"
      ],
      "Resource": "*",
      "Condition": {"Bool": {"kms:GrantIsForAWSResource": "true"}}
    }
  ]
}
```

Administrators of an external account that have usage permissions to a key can further restrict access to that key by creating a resource-level IAM policy. The third statement in the above policy adds external account number 109876543210 to the list of principals that can use the key. However, IAM users in account 109876543210 can use the key only if that account's administrator creates and attaches a resource-level policy that specifies the key ARN and permitted actions. For example, assume that users need to be able to create encrypted AWS resources in any service that integrates with AWS KMS using key `12345678-1234-1234-1234-123456789012`. The administrator must attach a policy similar to the following to grant the users the appropriate permissions. This policy was created by using the IAM policy generator. For more information about creating IAM policies, see Managing IAM Policies. Note that key 12345678-1234-1234-1234-123456789012 will not appear in the AWS console for administrators to directly manage or select when creating encrypted resources. The key's ARN can only be referenced in a policy document associated with IAM users under the target account (109876543210 in this example).

```
{
  "Id": "key-consolepolicy-1",
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowUseOfTheKey",
      "Effect": "Allow",
      "Action": [
        "kms:Encrypt",
        "kms:Decrypt",
        "kms:ReEncrypt*",
```

```
        "kms:GenerateDataKey*",
        "kms:DescribeKey"
         ],
      "Resource": [
          "arn:aws:kms:us-east-1:0123456789012:key/12345678-1234-1234-1234-
123456789012"
        ]
    },
    {
      "Sid": "AllowAttachmentOfPersistentResources",
      "Effect": "Allow",
      "Action": [
        "kms:CreateGrant",
        "kms:ListGrants",
        "kms:RevokeGrant"
      ],
      "Resource": [
         "arn:aws:kms:us-east-1:0123456789012:key/12345678-1234-1234-1234-
123456789012”
      ]
      "Condition": {"Bool": {"kms:GrantIsForAWSResource": true}}
     }
  ]
}
```

Policy Evaluation

When authorizing access to a key, AWS KMS evaluates both the resource-based permissions as well as any relevant user-based permissions attached to the key as well as any resource level permissions attached to the user. The default policy on the key enables * access for the owner account and that, in turn, enables IAM users permissions defined in the account to be applied. AWS KMS uses the algorithm described in the IAM Policy Evaluation Logic to authorize access to the key. In particular, `Deny` statements take precedence wherever they occur.

For example, assume that there are two keys and three users. The keys and users have the following policies:

- Key 1 allows the account principal to use the key.
- Key 2 allows users A and C to use the key.
- User A has no attached policy.
- User B's attached policy allows B to perform all AWS KMS actions on all keys.
- User C's attached policy denies all AWS KMS actions on all keys.

User C cannot access either key 1 or key 2 because all AWS KMS actions are denied for user C via `Deny` statements in the respective IAM policy. User B can access key 1 but not key 2 because the key policy for key 2 does not grant the account principal use of the key and, therefore, the IAM policy defined in the account is not evaluated. User A is not allowed to access key 1 because user A has no AWS KMS rights in the IAM policy, but user A can access key 2 because the key policy grants explicit access to user A.

When managing key policies, be careful to not lock yourself out of a key and make the key unmanageable. For example, in the example above, if someone accidentally removes the account principal access from key 1, or if users A and C are removed from the account, the respective keys cannot be managed or used by anyone.

Grants

AWS KMS supports two resourced-based access control mechanisms — key policies and grants. Grants enable customers to programmatically delegate programmatic use of KMS keys to other AWS principals. Key policies can also be used to enable access to other principals, but they work best for relatively static assignments of rules for key use. Key policy changes follow the same permissions model used for policy editing elsewhere in AWS. That is, users either have permissions to manage the policy or they do not. Those users with `PutPolicy` access for a key can completely replace the policy on a key with the policy of their choice. To enable more granular permissions around permissions management, use grants.

When creating a grant, you specify the principal, the set of operations that can be performed on a particular key, and optional constraints based on the encryption context. Once the grant has been created, the principal identified in the grant can execute the permitted operations subject to the defined constraints for as long as the grant is active. Grants must be explicitly revoked by a user with `RevokeGrant` access on the key or may be retired by the principal designated as the `RetiringPrincipal` for the grant.

You call the `CreateGrant` API to create a grant. Pass both an identifier of the key for which the grant is to be created, the grantee principal being given permission to use the key, and a list of operations to be permitted. `CreateGrant` returns a grant ID that you can use to identify the grant in subsequent operations. To further customize the grant permissions, you can also pass optional parameters that define grant constraints.

There are two supported grant constraints - `EncryptionContextEquals` and `EncryptionContextSubset`. `EncryptionContextEquals` specifies that the grant applies only when the exact specified `EncryptionContext` is present in the request (for example, Encrypt, Decrypt etc). `EncryptionContextSubset` specifies that the grant applies as long as all the entries in the encryption context subset constraint are matched by the request. The request may contain additional encryption context entries. For example, a grant for operations `Encrypt` and `Decrypt` with `EncryptionContextSubset` constraint `{“Department”:”Finance”, “Classification”:”Public”}` will allow encryption and decryption of either: `{“Department”:”Finance”, “Classification”:”Public”}` or `{“Department”:”Finance”, “Classification”:”Public”, “Customer”:”12345”}` but will not apply to `{“Department”:”Finance”}`.

When the grant includes `CreateGrant` as a `GrantOperation`, the grant only allows creation of equally or more restrictive grants. That is, the grant operations may be any subset of the grant operations and the `GrantConstraint` can be the same or more restrictive (fields can be added to an `EncryptionContextSubset` constraint, or an `EncryptionContextSubset` constraint can be turned into an `EncryptionContextEquals` constraint).

Rotating Keys

When you request AWS KMS to create a customer master key (CMK), the service creates a key ID for the CMK and key material referred to as a backing key that is tied to the key ID of the CMK. If you choose to enable key rotation for a given CMK, AWS KMS will create a new version of the backing key for each rotation. It is the backing key that is used to perform cryptographic operations such as encryption and decryption. When you choose a CMK to encrypt new data, AWS KMS automatically uses the latest version of the backing key to perform the encryption. When you want to decrypt data, AWS KMS automatically determines the correct version of the backing key to use. From your point of view, your CMK is simply a logical resource that does not change regardless of whether or of how many times the underlying backing keys have been rotated.

Automated key rotation currently retains all prior backing keys so that decryption of encrypted data can take place transparently. If you want to prevent decryption of old ciphertexts, you can create a new CMK and change your alias to point to the new key. You can then control when you choose to disable the old key. Disabling a CMK prevents the backing keys tied to it from being used to encrypt or to decrypt.

For more detailed information about backing keys and rotation, see the KMS Cryptographic Details whitepaper.

How AWS Services use AWS KMS

AWS KMS is integrated with multiple AWS services. The following topics discuss how these services use AWS KMS to provide encryption of customer data. The first topic in the list describes how envelope encryption works in the context of the supported services.

Topics

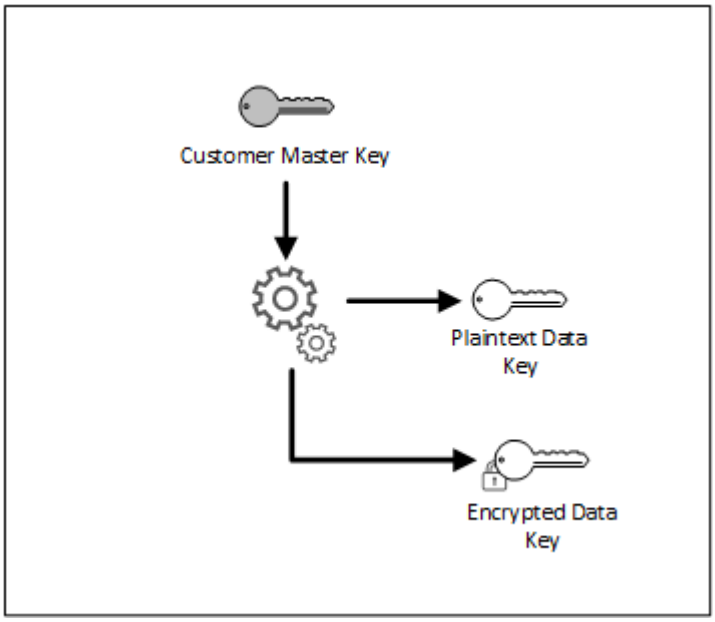
AWS KMS Workflow with Supported AWS Services

This topic describes how and when AWS KMS generates, encrypts, and decrypts keys that can be used to encrypt your data in a supported AWS service.

Envelope Encryption

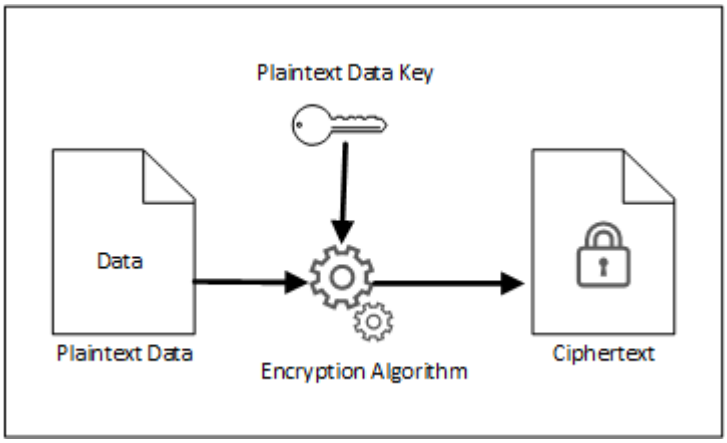
AWS KMS supports two kinds of keys — *master keys* and *data keys*. Master keys can be used to directly encrypt and decrypt up to 4 kilobytes of data and can also be used to protect data keys. The data keys are then used to encrypt and decrypt customer data.

Customer master keys are stored securely in AWS KMS. They can never be exported from AWS KMS. Data keys created inside of AWS KMS can be exported and are protected for export by being encrypted under a master key. The data key encryption process is illustrated by the following diagram:



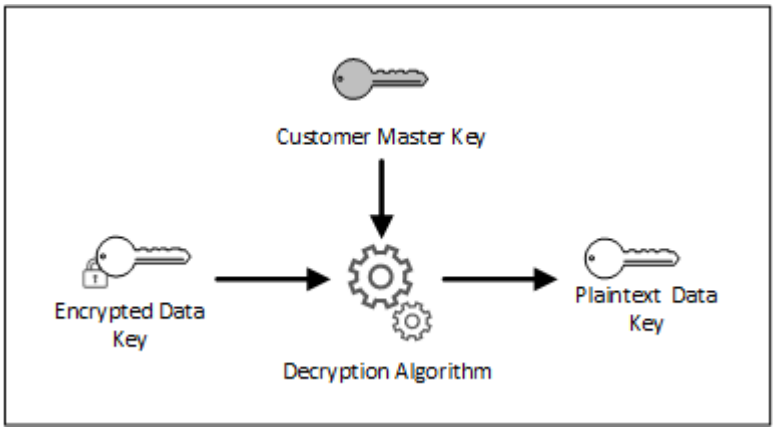
Encrypting User Data

When a data key is requested, AWS KMS returns both the encrypted and plaintext key back to the service or application that requested it. The plaintext data key is used to encrypt the user's data in memory. This key should never be written to disk and should be deleted from memory as soon as practical. The encrypted copy of the data key should be written to disk alongside of the encrypted data. This is acceptable and simplifies management of the encrypted data key.

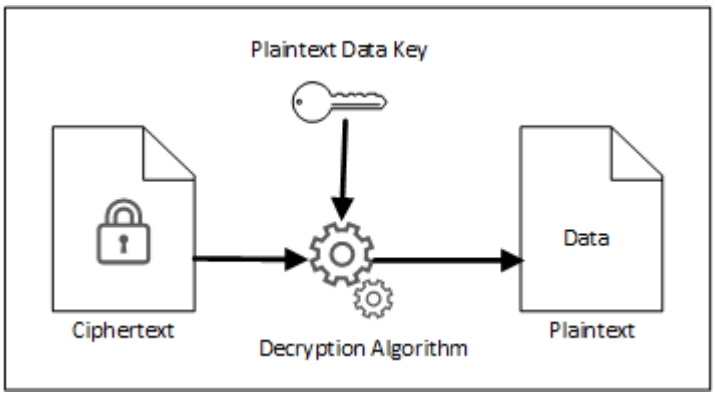


Decrypting User Data

Decryption reverses the encryption process. When a service or application needs to decrypt data, it sends AWS KMS the encrypted data key. AWS KMS decrypts the data key by automatically using the correct customer master key and then sends the plaintext key back to the service or application that requested it. The plaintext key is used to decrypt the data. This key should never be written to disk and should be deleted as soon as is practical. The following illustration shows the customer master key used with the symmetric decryption algorithm to decrypt the data key.



The next illustration shows the plaintext data key and symmetric algorithm used together to decrypt the user's encrypted data. The plaintext data key should be removed from memory as soon as is practical.



Managed Keys in AWS Services and in Custom Applications

You can choose to encrypt data in one of the services integrated with AWS KMS by using the AWS-managed key for that service under your account. In this case, all users who have access to that service can use the key. For more granular control, you can choose to create a customer-managed key and set policies that define who can use the key and what actions the users can perform.

How Amazon Simple Storage Service Uses AWS KMS

This topic discusses how to protect data at rest within Amazon S3 data centers by using AWS KMS. There are two ways to use AWS KMS with Amazon S3. You can use server-side encryption to protect your data with a customer master key or you can use a AWS KMS customer master key with the Amazon S3 encryption client to protect your data on the client side.

Topics

- Encryption Context (p. 27)

Server-Side Encryption: Using SSE-KMS

You can protect data at rest in Amazon S3 by using three different modes of server-side encryption: SSE-S3, SSE-C, or SSE-KMS.

- SSE-S3 requires that Amazon S3 manage the data and master encryption keys. For more information about SSE-S3, see Protecting Data Using Server-Side Encryption with AWS-Managed Encryption Keys.
- SSE-C requires that you manage the encryption key. For more information about SSE-C, see Protecting Data Using Server-Side Encryption with Customer-Provided Encryption Keys (SSE-C).
- SSE-KMS requires that AWS manage the data key but you manage the master key in AWS KMS. The remainder of this topic discusses how to protect data by using server-side encryption with AWS KMS-managed keys (SSE-KMS).

You can request encryption and the master key you want by using the Amazon S3 console or API. In the console, check the appropriate box to perform encryption and select your key from the list. For the Amazon S3 API, specify encryption and choose your key by setting the appropriate headers in a GET or PUT request. For more information, see Protecting Data Using Server-Side Encryption with AWS KMS-Managed Keys (SSE-KMS).

You can choose a specific customer-managed master key or accept the AWS-managed key for Amazon S3 under your account. If you choose to encrypt your data, AWS KMS and Amazon S3 perform the following actions:

- Amazon S3 requests a plaintext data key and a copy of the key encrypted by using the specified customer-managed master key or the AWS-managed master key.
- AWS KMS creates a data key, encrypts it by using the master key, and sends both the plaintext data key and the encrypted data key to Amazon S3.
- Amazon S3 encrypts the data using the data key and removes the plaintext key from memory as soon as possible after use.
- Amazon S3 stores the encrypted data key as metadata with the encrypted data.

Amazon S3 and AWS KMS perform the following actions when you request that your data be decrypted.

- Amazon S3 sends the encrypted data key to AWS KMS.
- AWS KMS decrypts the key by using the appropriate master key and sends the plaintext key back to Amazon S3.
- Amazon S3 decrypts the ciphertext and removes the plaintext data key from memory as soon as possible.

Using the Amazon S3 Encryption Client

You can use the Amazon S3 encryption client in the AWS SDK from your own application to encrypt objects and upload them to Amazon S3. This method allows you to encrypt your data locally to ensure its security as it passes to the Amazon S3 service. The S3 service receives your encrypted data and does not play a role in encrypting or decrypting it.

The Amazon S3 encryption client encrypts the object by using envelope encryption. The client calls AWS KMS as a part of the encryption call you make when you pass your data to the client. AWS KMS verifies that you are authorized to use the customer master key and, if so, returns a new plaintext data key and

the data key encrypted under the customer master key. The encryption client encrypts the data by using the plaintext key and then deletes the key from memory. The encrypted data key is sent to Amazon S3 to store alongside your encrypted data.

Encryption Context

Each service that is integrated with AWS KMS specifies an encryption context when requesting data keys, encrypting, and decrypting. The encryption context is additional authenticated information that AWS KMS uses to check for data integrity. That is, when an encryption context is specified for an encryption operation, the service also specifies it for the decryption operation or decryption will not succeed. If you are using SSE-KMS or the Amazon S3 encryption client to perform encryption, Amazon S3 uses the bucket path as the encryption context. In the `requestParameters` field of a CloudTrail log file, the encryption context will look similar to this.

```
"encryptionContext": {
    "aws:s3:arn": "arn:aws:s3:::bucket_name/file_name"
},
```

For more information, see Encryption Context (p. 80).

How Amazon Elastic Block Store (Amazon EBS) Uses AWS KMS

This topic discusses how Amazon EBS uses AWS KMS to encrypt volumes.

Topics

Amazon EBS Encryption

When you create an encrypted Amazon EBS volume and attach it to a supported Amazon EC2 instance type, data stored at rest on the volume, disk I/O, and snapshots created from the volume are all encrypted. The encryption occurs on the servers that host Amazon EC2 instances.

This feature is supported on all Amazon EBS volume types: General Purpose (SSD), Provisioned IOPS (SSD), and Magnetic. You access encrypted volumes the same way you access existing volumes; encryption and decryption are handled transparently and they require no additional action from you, your Amazon EC2 instance, or your application. Snapshots of encrypted volumes are automatically encrypted, and volumes that are created from encrypted snapshots are also automatically encrypted.

To create an encrypted Amazon EBS volume, you select the appropriate box in the Amazon EBS section of the Amazon EC2 console. You can use a custom customer master key (CMK) by choosing one from the list that appears below the encryption box. If you do not specify a custom CMK, Amazon EBS uses the default CMK in your account. If there is no default CMK in your account, Amazon EBS creates one.

When you create and use an encrypted volume, AWS KMS and Amazon EBS perform the following actions:

1. AWS KMS determines whether you have permission to use the specified customer master key (CMK).

2. If you have permission to use the key, AWS KMS generates a data key and encrypts it using the CMK.
3. AWS KMS sends the encrypted data key to Amazon EBS.
4. When you attach the encrypted volume to an Amazon EC2 instance, Amazon EC2 sends your encrypted data key to AWS KMS to decrypt it.
5. Amazon EC2 encrypts all data going to and from the Amazon EBS volume by using the plaintext data key returned from AWS KMS. The plaintext data key is kept in memory for as long as your volume is attached.
6. Amazon EBS stores the encrypted data key with the volume metadata for future use if you need to re-attach the same encrypted volume to an instance.

Encryption Context

Each service that is integrated with AWS KMS specifies an encryption context when requesting data keys, encrypting, and decrypting. The encryption context is additional authenticated information that AWS KMS uses to check for data integrity. That is, when an encryption context is specified for an encryption operation, the same encryption context must be specified for the decryption operation or decryption will not succeed. Amazon EBS uses the volume ID for the encryption context. In the `requestParameters` field of a CloudTrail log entry, the encryption context will look similar to the following:

```
"encryptionContext": {
    "aws:ebs:id": "vol-2cfb133e"
}
```

You can search for the volume ID in your CloudTrail logs to understand what operations were performed using a customer master key. The operations include encryption, decryption, and generating data keys.

For more information, see Encryption Context (p. 80).

Using AWS CloudFormation To Create Encrypted Amazon EBS Volumes

You can use AWS CloudFormation to create encrypted Amazon EBS volumes. For more information, go to AWS::EC2::Volume in the Template Reference chapter of the *AWS CloudFormation User Guide*.

How Amazon Redshift Uses AWS KMS

This topic discusses how Amazon Redshift uses AWS KMS to encrypt data.

Topics

Amazon Redshift Encryption

An Amazon Redshift data warehouse is a collection of computing resources called nodes, which are organized into a group called a cluster. Each cluster runs an Amazon Redshift engine and contains one or more databases.

Amazon Redshift uses a four-tier, key-based architecture for encryption. The architecture consists of data encryption keys, a database key, a cluster key, and a master key.

Data encryption keys encrypt data blocks in the cluster. Each data block is assigned a randomly-generated AES-256 key. These keys are encrypted by using the database key for the cluster.

The database key encrypts data encryption keys in the cluster. The database key is a randomly-generated AES-256 key. It is stored on disk in a separate network from the Amazon Redshift cluster and passed to the cluster across a secure channel.

The cluster key encrypts the database key for the Amazon Redshift cluster. You can use AWS KMS, AWS CloudHSM, or an external hardware security module (HSM) to manage the cluster key. See the Amazon Redshift Database Encryption documentation for more details.

If the master key resides in AWS KMS, it encrypts the cluster key. You can request encryption by checking the appropriate box in the Amazon Redshift console. You can specify a customer-managed master key to use by choosing one from the list that appears below the encryption box. If you do not specify a customer-managed key, the AWS-managed key for Amazon Redshift under your account will be used.

Encryption Context

Each service that is integrated with AWS KMS specifies an encryption context when requesting data keys, encrypting, and decrypting. The encryption context is additional authenticated information that AWS KMS uses to check for data integrity. That is, when an encryption context is specified for an encryption operation, the service also specifies it for the decryption operation or decryption will not succeed. Amazon Redshift uses the cluster ID and the creation time for the encryption context. In the `requestParameters` field of a CloudTrail log file, the encryption context will look similar to this.

```
"encryptionContext": {
    "aws:redshift:arn": "arn:aws:redshift:region:ac
count_ID:cluster:cluster_name",
    "aws:redshift:createtime": "20150206T1832Z"
},
```

You can search on the cluster name in your CloudTrail logs to understand what operations were performed by using a customer master key. The operations include cluster encryption, cluster decryption, and generating data keys.

For more information, see Encryption Context (p. 80).

How Amazon Relational Database Service Uses AWS KMS

This topic discusses how Amazon RDS uses Amazon EBS encryption to provide full disk encryption for database volumes, and how to encrypt Amazon RDS databases using AWS KMS. It also describes the Amazon RDS encryption context.

Topics

Amazon RDS Encryption

Amazon Relational Database Service (Amazon RDS) is a web service that makes it easier to set up, operate, and scale a relational database in the cloud. It provides cost-efficient, resizable capacity for an industry-standard relational database and manages common database administration tasks.

The basic building block of Amazon RDS is the DB instance. A DB instance is an isolated database environment in the cloud. A DB instance can contain multiple user-created databases, and you can access it by using the same tools and applications that you use with a stand-alone database instance. Each DB instance runs a DB engine. Amazon RDS currently supports the MySQL, PostgreSQL, Oracle, Microsoft SQL Server, and Aurora DB engines. Amazon RDS encryption with AWS KMS is currently available for all except the Aurora DB engine.

To enable Amazon RDS encryption, select the **Enable encryption** option in the Amazon RDS console. If you are using the `rds-create-db-instance` CLI command to create an encrypted RDS DB instance, set the `--storage-encrypted` parameter to `true` and the `--kms-key-id` parameter to the Amazon Resource Name (ARN) for the customer master key. If you are using the `CreateDBInstance` API action, set the `StorageEncrypted` parameter to `true` and the `KmsKeyId` parameter to the ARN for your customer master key.

For more information about Amazon RDS encryption, see Encrypting Amazon RDS Resources.

Amazon RDS uses Amazon EBS encryption to provide full disk encryption for database volumes. When you create an encrypted Amazon RDS database, an encrypted Amazon EBS volume is created on your behalf to store the database. Data stored at rest on the volume, input and output to and from the database instance, and database snapshots are all encrypted using the AWS KMS key you specify. For more information about how encryption of EBS volumes works, see How Amazon Elastic Block Store (Amazon EBS) Uses AWS KMS (p. 27).

Amazon RDS Encryption Context

Each service that is integrated with AWS KMS specifies an encryption context when requesting data keys, encrypting, and decrypting. The encryption context is additional authenticated information that AWS KMS uses to ensure data integrity. That is, when an encryption context is specified for an encryption operation, the service also specifies it for the decryption operation or decryption will not succeed. The encryption context is also written to your CloudTrail logs to help you understand why a given AWS KMS key was used. Amazon RDS uses the volume ID and the database instance ID for the encryption context. In the `requestParameters` field of a CloudTrail log file, the encryption context will look similar to this.

```
"encryptionContext": {
    "aws:rds:db-id":"db-CQYSMDPBRZ7BPMH7Y3RTDG5QY",
    "aws:ebs:id":"vol-57d21d70"
}
```

For more information about the encryption context, see Encryption Context (p. 80).

How Elastic Transcoder Uses AWS KMS

You can use Elastic Transcoder to convert media files stored in an Amazon S3 bucket into formats required by consumer playback devices. Both input and output files can be encrypted and decrypted. The following sections discuss how AWS KMS is used for both processes.

Encrypting the input file

Before you can use Elastic Transcoder, you must create an Amazon S3 bucket and upload your media file into it. You can encrypt the file before uploading by using Amazon S3 server-side encryption or AES client-side encryption.

If you choose client-side encryption using AES, you are responsible for encrypting the file before uploading it to Amazon S3, and you must provide Elastic Transcoder access to the encryption key. You do this by using an AWS KMS customer master key to protect the AES encryption key you used to encrypt the media file.

If you choose server-side encryption, you are allowing Amazon S3 to perform all encryption and decryption of files on your behalf. You can configure Amazon S3 to use one of three different master keys to protect the unique data key used to encrypt your file:

- The Amazon S3 master key, a key that is owned and managed by AWS
- The default key for Amazon S3, a key that is owned by your account but managed by AWS
- Any of the customer-managed keys you create by using AWS KMS

You can request encryption and the master key you want by using the Amazon S3 console or the appropriate Amazon S3 APIs. For more information about how Amazon S3 performs encryption, see Protecting Data Using Encryption in the *Amazon Simple Storage Service Developer Guide*.

When you specify that the AWS-managed key for Amazon S3 under your account or a customer-managed key be used to encrypt the input file, Amazon S3 and AWS KMS interact in the following manner:

- Amazon S3 requests a plaintext data key and a copy of the key encrypted by using the specified customer-managed key or the AWS-managed key.
- AWS KMS creates a data key, encrypts it by using the customer master key, and sends both the plaintext data key and the encrypted data key to Amazon S3.
- Amazon S3 encrypts the media file using the plaintext data key and stores the file in the specified Amazon S3 bucket.
- Amazon S3 stores the encrypted data key alongside of the encrypted media file.

Decrypting the input file

If you choose Amazon S3 server-side encryption to encrypt the input file before uploading it, Elastic Transcoder does not decrypt the file. Instead, Elastic Transcoder relies on Amazon S3 to perform decryption depending on the settings you specify when you create a job and a pipeline. The following combination of settings are available.

Encryption mode	AWS KMS key	Meaning
S3	Default	Amazon S3 creates and manages the keys used to encrypt and decrypt the media file. The process is opaque to the user.
S3-AWS-KMS	Default	Amazon S3 uses a data key encrypted by the default AWS-managed key for S3 under your account to encrypt the media file.

Encryption mode	AWS KMS key	Meaning
S3-AWS-KMS	Custom (with ARN)	Amazon S3 uses a data key encrypted by the specified customer-managed key to encrypt the media file.

When **S3-AWS-KMS** is specified, Amazon S3 and AWS KMS work together in the following manner to perform the decryption.

- Amazon S3 sends the encrypted data key to AWS KMS.
- AWS KMS decrypts the key by using the appropriate customer master key and sends the plaintext key back to Amazon S3.
- Amazon S3 uses the plaintext key to decrypt the ciphertext.

If you choose client-side encryption using an AES key, then Elastic Transcoder retrieves the encrypted file from the Amazon S3 bucket and decrypts it. Elastic Transcoder uses the customer master key you specify when creating the pipeline to decrypt the AES key and then uses the AES key to decrypt the media file.

Encrypting the output file

Elastic Transcoder encrypts the output file depending on how you specify the encryption settings when you create a job and a pipeline. The following options are available.

Encryption mode	AWS KMS key	Meaning
S3	Default	Amazon S3 creates and manages the keys used to encrypt the output file.
S3-AWS-KMS	Default	Amazon S3 uses a data key created by AWS KMS and encrypted by the AWS-managed key for Amazon S3 under your account.
S3-AWS-KMS	Custom (with ARN)	Amazon S3 uses a data key encrypted by using the customer-managed key specified by the ARN to encrypt the media file.
AES-	Default	Elastic Transcoder uses the AWS-managed key for Amazon S3 under your account to decrypt the specified AES key you provide and uses that key to encrypt the output file.
AES-	Custom (with ARN)	Elastic Transcoder uses the customer-managed key specified by the ARN to decrypt the specified AES key you provide and uses that key to encrypt the output file.

When you specify that the AWS-managed key for Amazon S3 under your account or a customer-managed key be used to encrypt the output file, Amazon S3 and AWS KMS interact in the following manner:

- Amazon S3 requests a plaintext data key and a copy of the key encrypted by using the specified customer-managed key or the AWS-managed key.
- AWS KMS creates a data key, encrypts it by using the master key, and sends both the plaintext data key and the encrypted data key to Amazon S3.
- Amazon S3 encrypts the media using the data key and stores it in the specified Amazon S3 bucket.
- Amazon S3 stores the encrypted data key alongside the encrypted media file.

When you specify that an AES key you provide must be used to encrypt the output file, the AES key must be encrypted by using a master key in AWS KMS. Elastic Transcoder, AWS KMS and you interact in the following manner:

- You encrypt your AES key by calling the AWS KMS `Encrypt` API. AWS KMS encrypts the key by using the specified AWS-managed key for Amazon S3 under your account or a customer-managed key you have previously created. You specify which key to use when you are creating the pipeline.
- You specify the file containing the encrypted AES key when you create the Elastic Transcoder job.
- Elastic Transcoder decrypts the key by calling the AWS KMS `Decrypt` API, passing the encrypted key as ciphertext.
- Elastic Transcoder uses the decrypted AES key to encrypt the output media file and then deletes the decrypted AES key from memory. Only the encrypted copy you originally defined in the job is saved to disk.
- You can download the encrypted output file and decrypt it locally by using the original AES key that you defined.

Important
Your private encryption keys are never stored by AWS. Therefore, it is important that you safely and securely manage your keys. If you lose them, you won't be able to decrypt your data.

HLS Content Protection

HTTP Live Streaming (HLS) is an adaptive streaming protocol created by Apple. Elastic Transcoder supports HLS by breaking your input file into smaller individual files, called media segments. A set of corresponding individual media segments contain the same material encoded at different bit rates, thereby enabling the player to select the stream that best fits the available bandwidth. Elastic Transcoder also creates playlists that contain metadata for the various segments that are available to be streamed.

You can use AES-128 encryption to protect the transcoded media segments. When you enable HLS content protection, each media segment is encrypted using an AES-128 encryption key. When the content is viewed, the player downloads the key and decrypts the media segments during the playback process.

Two types of keys are used - a customer master key (CMK) and a data key. You must create a CMK that can be used to encrypt and decrypt the data key. The data key is used by Elastic Transcoder to encrypt and decrypt media segments. The data key must be AES-128. All variations and segments of the same content are encrypted using the same key. You can provide a data key or have Elastic Transcoder create it for you.

The CMK can be used to encrypt the data key at the following points:

1. If you provide your own data key, you must encrypt it before passing it to Elastic Transcoder.
2. If you request that Elastic Transcoder generate the data key, then Elastic Transcoder encrypts the key for you.

The CMK can be used to decrypt the data key at the following points:

1. Elastic Transcoder decrypts a data key that you provide when it needs to use the key to encrypt the output file or decrypt the input file.
2. You decrypt a data key generated by Elastic Transcoder and use it to decrypt output files.

For more information, see HLS Content Protection in the *Amazon Elastic Transcoder Developer Guide*.

Encryption Context

Each service that is integrated with AWS KMS specifies an encryption context when requesting data keys, encrypting, and decrypting. The encryption context is additional authenticated information that AWS KMS uses to ensure data integrity. That is, when an encryption context is specified for an encryption operation, the service also specifies it for the decryption operation or decryption will not succeed. The encryption context is written to CloudTrail logs to help you understand why a given AWS KMS key was used. Elastic Transcoder uses the key ID and your AES key as the encryption context. In the `requestParameters` field of a CloudTrail log file, the encryption context will look similar to this.

```
"EncryptionContext": {
    "service" : "elastictranscoder.amazonaws.com",
    "KeyId" : "ARN of the key associated with your pipeline",
    "Plaintext" : "BLOB that contains your AES key"
}
```

For more information, see Encryption Context (p. 80). For more information about how to configure Elastic Transcoder jobs to use one of the supported encryption options, see Data Encryption Options in the *Amazon Elastic Transcoder Developer Guide*.

How Amazon WorkMail Uses AWS KMS

This topic discusses how Amazon WorkMail uses AWS KMS to encrypt email messages.

Topics

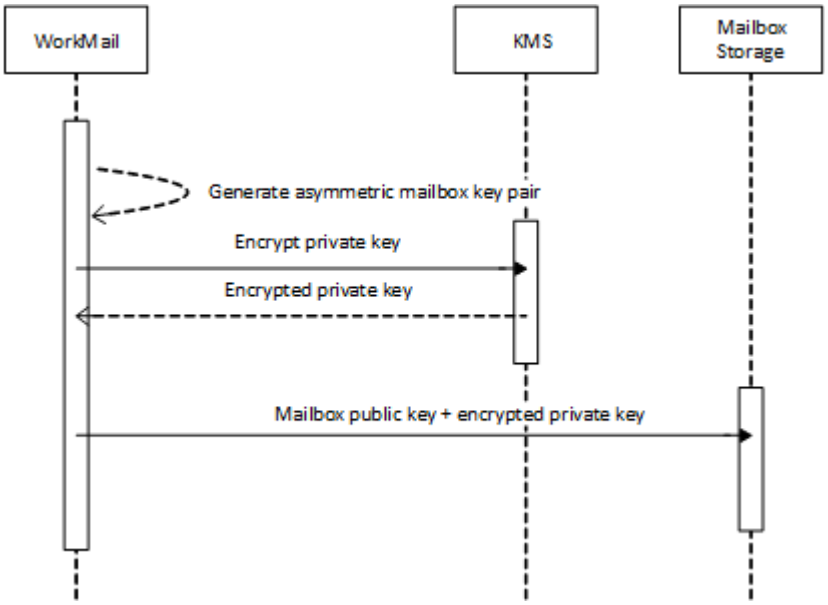
Amazon WorkMail Overview

Amazon WorkMail is an email service in the cloud that provides a cost-effective way for your organization to receive and send email and use calendars. Amazon WorkMail supports existing desktop and mobile clients and integrates with your existing corporate directory. Users can leverage their existing credentials to sign on to their email by using Microsoft Outlook, a mobile device, or a browser.

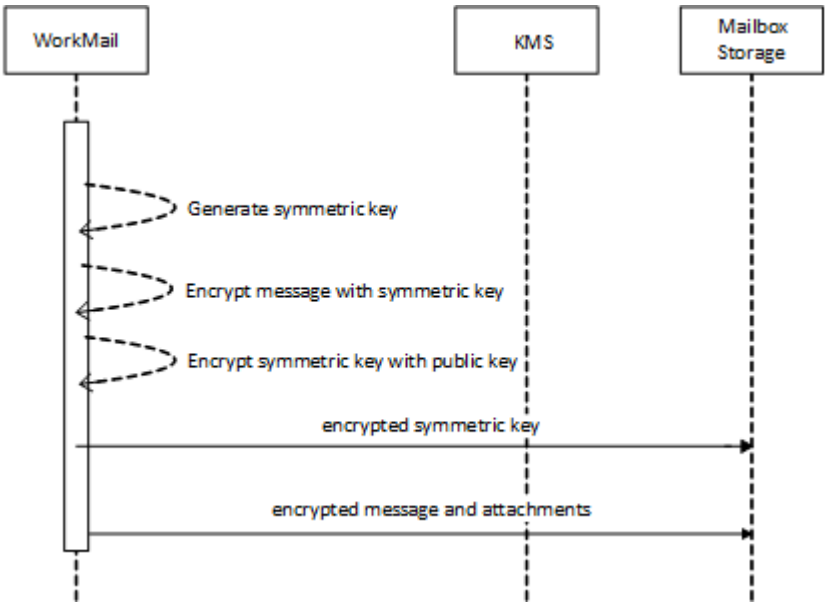
Using the Amazon WorkMail console, you can create an Amazon WorkMail organization and optionally assign to it one or more email domains that you own. Then you can create new email users and email distribution groups. Users can then send and receive messages. The messages are encrypted and stored until ready to be viewed.

Amazon WorkMail Encryption

Each end user you create is associated with one mailbox. Amazon WorkMail creates an asymmetric key pair for each mailbox and sends the private key portion of the key pair to AWS KMS to be encrypted under a customer master key (CMK). The CMK can be a custom key that you choose for your organization or the default Amazon WorkMail service CMK. The encrypted private key and unencrypted public key is then saved for later use.

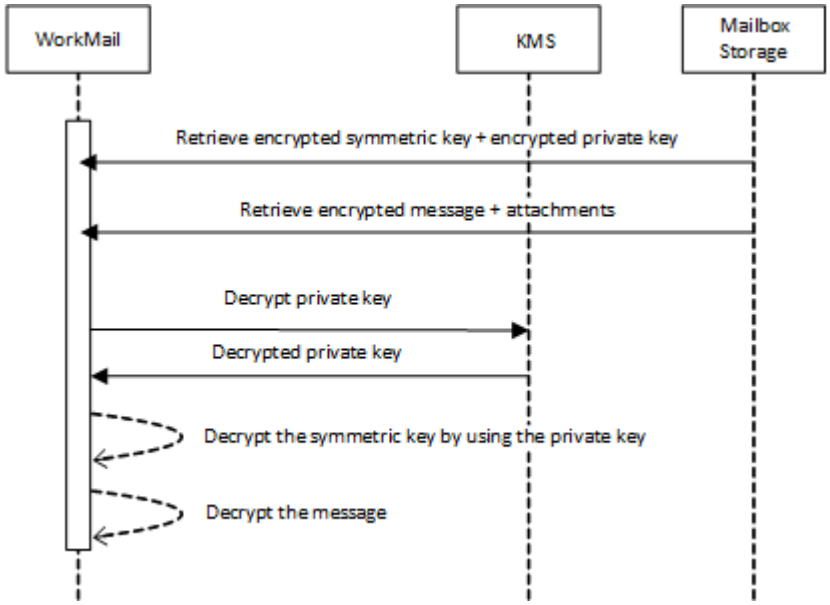


Each message received is encrypted by using a symmetric key dynamically generated by Amazon WorkMail. The symmetric key is then encrypted by using the public key associated with the user's mailbox. The encrypted symmetric key and the encrypted message and attachments are then stored.



In asymmetric cryptography, data that is encrypted by using the public key can be decrypted only by using the corresponding private key. As mentioned above, however, Amazon WorkMail encrypts the

private key by using an AWS KMS CMK. To make the private key ready to use, it must therefore be decrypted by using the same CMK used to encrypt it. Thus, when a user is ready to retrieve email messages, Amazon WorkMail sends the private key to AWS KMS for decryption and uses the plaintext private key returned by AWS KMS to decrypt the symmetric key that was used to encrypt the email message. Amazon WorkMail then uses the symmetric key to decrypt the message before presenting it to the user.



Amazon WorkMail Encryption Context

Each service that is integrated with AWS KMS specifies an encryption context when requesting data keys, encrypting, and decrypting. The encryption context is additional authenticated information that AWS KMS uses to check for data integrity. That is, when an encryption context is specified for an encryption operation, the service also specifies it for the decryption operation or decryption will not succeed. The encryption context is written to your CloudTrail logs to help you understand why a given AWS KMS key was used. Amazon WorkMail uses the organization ID for the encryption context. In the `requestParameters` field of a CloudTrail log file, the encryption context will look similar to this.

```
"encryptionContext": {
    "aws:workmail:arn":"arn:aws:workmail:region:account ID:organization/organization ID"
}
```

The organization ID is a unique identifier that Amazon WorkMail generates when an organization is created. A customer can have multiple organizations in an AWS account. The following example shows an organization ID in the us-east-1 region.

```
   "aws:workmail:arn":"arn:aws:workmail:us-east-1:123456789012:organization/m-68755160c4cb4e29a2b2f8fb58f359d7"
```

For more information about the encryption context, see Encryption Context (p. 80).

How Amazon EMR Uses AWS KMS

This topic discusses how the EMR File System (EMRFS) uses AWS KMS to decrypt and encrypt input and output stored in Amazon S3 for Amazon EMR workloads.

Topics

Amazon EMR Overview

Amazon Elastic MapReduce (Amazon EMR) is a web service that makes it easy to process vast amounts of data quickly and cost-effectively. Amazon EMR uses Hadoop, an open source framework, to distribute your data and processing across a resizeable cluster of Amazon EC2 instances. It can also run other distributed frameworks such as Spark and Presto and can process data stored in other services such as Amazon S3, DynamoDB, and Amazon Kinesis.

For more information about Amazon EMR and Hadoop, see What is Amazon EMR?.

Amazon EMR Support for Encrypted Objects in Amazon S3

Amazon EMR can store data directly on a cluster by using the Hadoop distributed file system (HDFS) or in Amazon S3 by using the EMR File System (EMRFS). The input and output for your Amazon EMR workloads is commonly stored in Amazon S3. EMRFS supports Amazon S3 server-side or client-side encryption when downloading encrypted objects from Amazon S3 and when uploading encrypted objects to Amazon S3. The server-side encryption option uses the Amazon S3 system master key and does not currently support the use of customer master keys (CMK) in AWS KMS. However, the client-side option does support the use of CMKs in AWS KMS. Note that data stored on the HDFS cluster and any data in transit between HDFS nodes is not encrypted at this time.

When encrypting Amazon EMR output to send to Amazon S3, the EMRFS client calls AWS KMS as part of the encryption routine that you execute when you pass in your data. The plaintext data key that the Amazon S3 encryption client generates is then sent to AWS KMS to be encrypted under the specified CMK. AWS KMS verifies that you are authorized to use the specified CMK. The encryption client uses the plaintext key to encrypt the data and sends the encrypted data key to Amazon S3 to store alongside your encrypted data.

You can enable Amazon S3 server-side or client-side encryption in the Amazon EMR console, CLI, or SDK when creating your cluster. For more information, see Create a Cluster With Amazon S3 Client-Side Encryption.

Amazon EMR Use of the AWS KMS Encryption Context

Each service that is integrated with AWS KMS specifies an encryption context when requesting data keys, encrypting, and decrypting. The encryption context is additional authenticated information that AWS KMS uses to check for data integrity. That is, when a service specifies an encryption context for an encryption operation, the service must also specify it for the decryption operation; otherwise, the decryption will not succeed. Amazon EMR uses the bucket path as the encryption context. In the `requestParameters` field of a CloudTrail log file, the encryption context looks similar to this:

```
"encryptionContext": {
    "aws:s3:arn":"arn:aws:s3:::bucket_name/file_name"
}
```

For more information about the encryption context, see Encryption Context (p. 80).

Logging AWS KMS API Calls Using AWS CloudTrail

AWS KMS is integrated with CloudTrail, a service that captures API calls made by or on behalf of AWS KMS in your AWS account and delivers the log files to an Amazon S3 bucket that you specify. CloudTrail captures API calls from the AWS KMS console or from the AWS KMS API. Using the information collected by CloudTrail, you can determine what request was made, the source IP address from which the request was made, who made the request, when it was made, and so on. To learn more about CloudTrail, including how to configure and enable it, see the AWS CloudTrail User Guide.

When you enable CloudTrail logging in your AWS account, API calls made to AWS KMS actions are tracked in log files. AWS KMS records are written together with other AWS service records in a log file. CloudTrail determines when to create and write to a new log file based on a time period and file size.

CloudTrail logs all of the AWS KMS actions. For example, calls to the `CreateKey`, `Encrypt`, and `Decrypt` actions generate entries in the CloudTrail log files.

Every log entry contains information about who generated the request. The user identity information in the log helps you determine whether the request was made with root or IAM user credentials, with temporary security credentials for a role or federated user, or by another AWS service. For more information, see userIdentity Element in the CloudTrail Event Reference chapter in the *AWS CloudTrail User Guide*.

You can store your log files in your bucket for as long as you want, but you can also define Amazon S3 lifecycle rules to archive or delete log files automatically. By default, your log files are encrypted by using Amazon S3 server-side encryption (SSE) with a key managed by the Amazon S3 service.

You can choose to have CloudTrail publish Amazon SNS notifications when new log files are delivered if you want to take quick action upon log file delivery. For more information, see Configuring Amazon SNS Notifications in the *AWS CloudTrail User Guide*.

You can also aggregate AWS KMS log files from multiple AWS regions and multiple AWS accounts into a single Amazon S3 bucket. For more information, see Aggregating CloudTrail Log Files to a Single Amazon S3 Bucket in the *AWS CloudTrail User Guide*.

CloudTrail log files can contain one or more log entries where each entry is made up of multiple JSON-formatted events. A log entry represents a single request from any source and includes information about the requested action, any parameters, the date and time of the action, and so on. The log entries are not guaranteed to be in any particular order. That is, they are not an ordered stack trace of the public

API calls. For more information about the fields that make up a log entry, see the CloudTrail Event Reference in the *AWS CloudTrail User Guide*.

The following topics show log examples for selected AWS KMS actions.

Topics

CreateAlias

The following example shows a log file generated by calling `CreateAlias`.


```
{
    "Records": [
    {
        "eventVersion": "1.02",
        "userIdentity": {
            "type": "IAMUser",
            "principalId": "EX_PRINCIPAL_ID",
            "arn": "arn:aws:iam::123456789012:user/Alice",
            "accountId": "123456789012",
            "accessKeyId": "EXAMPLE_KEY_ID",
            "userName": "Alice",
            "sessionContext": {
                "attributes": {
                    "mfaAuthenticated": "false",
                    "creationDate": "2014-11-04T00:52:27Z"
                }
            }
        },
        "eventTime": "2014-11-04T00:52:27Z",
        "eventSource": "kms.amazonaws.com",
        "eventName": "CreateAlias",
```

```
        "awsRegion": "us-east-1",
        "sourceIPAddress": "192.0.2.0",
        "userAgent": "AWS Internal",
        "requestParameters": {
            "aliasName": "alias/my_alias",
            "targetKeyId": "arn:aws:kms:us-east-1:123456789012:key/64e07f97-
2489-4d04-bfdf-41723ad130bd"
        },
        "responseElements": null,
        "requestID": "d9472f40-63bc-11e4-bc2b-4198b6150d5c",
        "eventID": "f72d3993-864f-48d6-8f16-e26e1ae8dff0",
        "readOnly": false,
        "resources": [{
            "ARN": "arn:aws:kms:us-east-1:123456789012:key/64e07f97-2489-4d04-
bfdf-41723ad130bd",
            "accountId": "123456789012"
        },
        {
            "ARN": "arn:aws:kms:us-east-1:123456789012:alias/my_alias",
            "accountId": "123456789012"
        }],
        "eventType": "AwsApiCall",
        "recipientAccountId": "123456789012"
    }
  ]
}
```

CreateGrant

The following example shows a log file generated by calling `CreateGrant`.

```
{
    "Records": [
    {
        "eventVersion": "1.02",
        "userIdentity": {
            "type": "IAMUser",
            "principalId": "EX_PRINCIPAL_ID",
            "arn": "arn:aws:iam::123456789012:user/Alice",
            "accountId": "123456789012",
            "accessKeyId": "EXAMPLE_KEY_ID",
            "userName": "Alice"
        },
        "eventTime": "2014-11-04T00:53:12Z",
        "eventSource": "kms.amazonaws.com",
        "eventName": "CreateGrant",
        "awsRegion": "us-east-1",
        "sourceIPAddress": "192.0.2.0",
        "userAgent": "AWS Internal",
        "requestParameters": {
          "keyId": "arn:aws:kms:us-east-1:123456789012:key/65f61d18-c45c-41ca-
90c9-179982e9b716",
            "constraints": {
```

```
                "encryptionContextSubset": {
                    "ContextKey1": "Value1"
                }
            },
            "operations": ["Encrypt",
            "RetireGrant"],
            "granteePrincipal": "EX_PRINCIPAL_ID"
        },
        "responseElements": {
            "grantId":
"f020fe75197b93991dc8491d6f19dd3cebb24ee62277a05914386724f3d48758"
        },
        "requestID": "f3c08808-63bc-11e4-bc2b-4198b6150d5c",
        "eventID": "5d529779-2d27-42b5-92da-91aaea1fc4b5",
        "readOnly": false,
        "resources": [{
            "ARN": "arn:aws:kms:us-east-1:123456789012:key/65f61d18-c45c-41ca-
90c9-179982e9b716",
            "accountId": "123456789012"
        }],
        "eventType": "AwsApiCall",
        "recipientAccountId": "123456789012"
    }
  ]
}
```

CreateKey

The following example shows a log file generated by calling `CreateKey`.

```
{
    "Records": [
    {
        "eventVersion": "1.02",
        "userIdentity": {
            "type": "IAMUser",
            "principalId": "EX_PRINCIPAL_ID",
            "arn": "arn:aws:iam::123456789012:user/Alice",
            "accountId": "123456789012",
            "accessKeyId": "EXAMPLE_KEY_ID",
            "userName": "Alice"
        },
        "eventTime": "2014-11-04T00:52:59Z",
        "eventSource": "kms.amazonaws.com",
        "eventName": "CreateKey",
        "awsRegion": "us-east-1",
        "sourceIPAddress": "192.0.2.0",
        "userAgent": "AWS Internal",
        "requestParameters": {
            "policy": "{\n  \"Version\":\"2012-10-17\",\n  \"Statement\":[{\n
   \"Effect\":\"Allow\",\n    \"Princip
al\":{\"AWS\":\"arn:aws:iam::123456789012:user/Alice\"},\n    \"Ac
```

```
tion\":\"kms:*\",\n    \"Resource\":\"*\"\n  }, {\n    \"Effect\":\"Allow\",\n
    \"Principal\":{\"AWS\":\"arn:aws:iam::012345678901:user/Bob\"},\n    \"Ac
tion\":\"kms:CreateGrant\",\n    \"Resource\":\"*\"\n  }, {\n    \"Effect\":\"Al
low\",\n    \"Principal\":{\"AWS\":\"arn:aws:iam::012345678901:user/Charlie\"},\n
    \"Action\":\"kms:Encrypt\",\n    \"Resource\":\"*\"\n}]\n}",
            "description": "",
            "keyUsage": "ENCRYPT_DECRYPT"
        },
        "responseElements": {
            "keyMetadata": {
                "aWSAccountId": "123456789012",
                "enabled": true,
                "creationDate": "Nov 4, 2014 12:52:59 AM",
                "keyId": "06dc80ca-1bdc-4d0b-be5b-b7009cd14f13",
                "keyUsage": "ENCRYPT_DECRYPT",
                "description": "",
                "arn": "arn:aws:kms:us-east-1:123456789012:key/06dc80ca-1bdc-
4d0b-be5b-b7009cd14f13"
            }
        },
        "requestID": "ebe8ee68-63bc-11e4-bc2b-4198b6150d5c",
        "eventID": "ba116326-1792-4784-87dd-a688d1cb42ec",
        "readOnly": false,
        "resources": [{
            "ARN": "arn:aws:kms:us-east-1:123456789012:key/06dc80ca-1bdc-4d0b-
be5b-b7009cd14f13",
            "accountId": "123456789012"
        }],
        "eventType": "AwsApiCall",
        "recipientAccountId": "123456789012"
    }
  ]
}
```


Decrypt

The following example shows a log file generated by calling `Decrypt`.


```
{
    "Records": [
    {
        "eventVersion": "1.02",
        "userIdentity": {
            "type": "IAMUser",
            "principalId": "EX_PRINCIPAL_ID",
            "arn": "arn:aws:iam::123456789012:user/Alice",
            "accountId": "123456789012",
            "accessKeyId": "EXAMPLE_KEY_ID",
            "userName": "Alice"
        },
        "eventTime": "2014-11-04T00:52:20Z",
        "eventSource": "kms.amazonaws.com",
        "eventName": "Decrypt",
```

```
        "awsRegion": "us-east-1",
        "sourceIPAddress": "192.0.2.0",
        "userAgent": "AWS Internal",
        "errorCode": "InvalidCiphertextException",
        "requestParameters": null,
        "responseElements": null,
        "requestID": "d5239dea-63bc-11e4-bc2b-4198b6150d5c",
        "eventID": "954983cf-7da9-4adf-aeaa-261a1292c0aa",
        "readOnly": true,
        "resources": [{
            "ARN": "arn:aws:kms:us-east-1:123456789012:key/e17cebae-e7a6-4864-
b92f-0365f2feff38",
            "accountId": "123456789012"
        }],
        "eventType": "AwsApiCall",
        "recipientAccountId": "123456789012"
    }
  ]
}
```

DeleteAlias

The following example shows a log file generated by calling `DeleteAlias`.

```
{
    "Records": [
    {
        "eventVersion": "1.02",
        "userIdentity": {
            "type": "IAMUser",
            "principalId": "EX_PRINCIPAL_ID",
            "arn": "arn:aws:iam::123456789012:user/Alice",
            "accountId": "123456789012",
            "accessKeyId": "EXAMPLE_KEY_ID",
            "userName": "Alice",
            "sessionContext": {
                "attributes": {
                    "mfaAuthenticated": "false",
                    "creationDate": "2014-11-04T00:52:27Z"
                }
            }
        },
        "eventTime": "2014-11-04T00:52:27Z",
        "eventSource": "kms.amazonaws.com",
        "eventName": "DeleteAlias",
        "awsRegion": "us-east-1",
        "sourceIPAddress": "192.0.2.0",
        "userAgent": "AWS Internal",
        "requestParameters": {
            "aliasName": "alias/my_alias"
        },
        "responseElements": null,
```

```
        "requestID": "d9542792-63bc-11e4-bc2b-4198b6150d5c",
        "eventID": "12f48554-bb04-4991-9cfc-e7e85f68eda0",
        "readOnly": false,
        "resources": [{
            "ARN": "arn:aws:kms:us-east-1:123456789012:alias/my_alias",
            "accountId": "123456789012"
        },
        {
            "ARN": "arn:aws:kms:us-east-1:123456789012:key/64e07f97-2489-4d04-
bfdf-41723ad130bd",
            "accountId": "123456789012"
        }],
        "eventType": "AwsApiCall",
        "recipientAccountId": "123456789012"
    }
  ]
}
```

DescribeKey

The following example shows a log file generated by multiple calls to `DescribeKey` in response to viewing keys in the IAM management console.

```
{
  "Records": [
    {
      "eventVersion": "1.02",
      "userIdentity": {
        "type": "IAMUser",
        "principalId": "EX_PRINCIPAL_ID",
        "arn": "arn:aws:iam::123456789012:user/Alice",
        "accountId": "123456789012",
        "accessKeyId": "EXAMPLE_KEY_ID",
        "userName": "Alice",
        "sessionContext": {
          "attributes": {
            "mfaAuthenticated": "false",
            "creationDate": "2014-11-05T20:51:21Z"
          }
        },
        "invokedBy": "signin.amazonaws.com"
      },
      "eventTime": "2014-11-05T20:51:34Z",
      "eventSource": "kms.amazonaws.com",
      "eventName": "DescribeKey",
      "awsRegion": "us-east-1",
      "sourceIPAddress": "192.0.2.0",
      "userAgent": "signin.amazonaws.com",
      "requestParameters": {
        "keyId": "30a9a1e7-2a84-459d-9c61-04cbeaebab95"
      },
      "responseElements": null,
      "requestID": "874d4823-652d-11e4-9a87-01af2a1ddecb",
```

```
      "eventID": "f715da9b-c52c-4824-99ae-88aa1bb58ae4",
      "readOnly": true,
      "resources": [
        {
          "ARN": "arn:aws:kms:us-east-1:123456789012:key/30a9a1e7-2a84-459d-
9c61-04cbeaebab95",
          "accountId": "123456789012"
        }
      ],
      "eventType": "AwsApiCall",
      "recipientAccountId": "123456789012"
    },
    {
      "eventVersion": "1.02",
      "userIdentity": {
        "type": "IAMUser",
        "principalId": "EX_PRINCIPAL_ID",
        "arn": "arn:aws:iam::123456789012:user/Alice",
        "accountId": "123456789012",
        "accessKeyId": "EXAMPLE_KEY_ID",
        "userName": "Alice",
        "sessionContext": {
          "attributes": {
            "mfaAuthenticated": "false",
            "creationDate": "2014-11-05T20:51:21Z"
          }
        },
        "invokedBy": "signin.amazonaws.com"
      },
      "eventTime": "2014-11-05T20:51:55Z",
      "eventSource": "kms.amazonaws.com",
      "eventName": "DescribeKey",
      "awsRegion": "us-east-1",
      "sourceIPAddress": "192.0.2.0",
      "userAgent": "signin.amazonaws.com",
      "requestParameters": {
        "keyId": "e7b6d35a-b551-4c8f-b51a-0460ebc04565"
      },
      "responseElements": null,
      "requestID": "9400c720-652d-11e4-9a87-01af2a1ddecb",
      "eventID": "939fcefb-dc14-4a52-b918-73045fe97af3",
      "readOnly": true,
      "resources": [
        {
          "ARN": "arn:aws:kms:us-east-1:123456789012:key/e7b6d35a-b551-4c8f-
b51a-0460ebc04565",
          "accountId": "123456789012"
        }
      ],
      "eventType": "AwsApiCall",
      "recipientAccountId": "123456789012"
    },
    ... additional entries ...
  ]
}
```

DisableKey

The following example shows a log file generated by calling `DisableKey`.

```
{
    "Records": [
    {
        "eventVersion": "1.02",
        "userIdentity": {
            "type": "IAMUser",
            "principalId": "EX_PRINCIPAL_ID",
            "arn": "arn:aws:iam::123456789012:user/Alice",
            "accountId": "123456789012",
            "accessKeyId": "EXAMPLE_KEY_ID",
            "userName": "Alice"
        },
        "eventTime": "2014-11-04T00:52:43Z",
        "eventSource": "kms.amazonaws.com",
        "eventName": "DisableKey",
        "awsRegion": "us-east-1",
        "sourceIPAddress": "192.0.2.0",
        "userAgent": "AWS Internal",
        "requestParameters": {
            "keyId": "262d9fcb-f1a0-4447-af16-3714cff61ec1"
        },
        "responseElements": null,
        "requestID": "e26552bc-63bc-11e4-bc2b-4198b6150d5c",
        "eventID": "995c4653-3c53-4a06-a0f0-f5531997b741",
        "readOnly": false,
        "resources": [{
            "ARN": "arn:aws:kms:us-east-1:123456789012:key/262d9fcb-f1a0-4447-
af16-3714cff61ec1",
            "accountId": "123456789012"
        }],
        "eventType": "AwsApiCall",
        "recipientAccountId": "123456789012"
    },
  ]
}
```

EnableKey

The following example shows a log file generated by calling `EnableKey`.

```
{
    "Records": [
    {
        "eventVersion": "1.02",
        "userIdentity": {
            "type": "IAMUser",
```

```
            "principalId": "EX_PRINCIPAL_ID",
            "arn": "arn:aws:iam::123456789012:user/Alice",
            "accountId": "123456789012",
            "accessKeyId": "EXAMPLE_KEY_ID",
            "userName": "Alice"
        },
        "eventTime": "2014-11-04T00:52:20Z",
        "eventSource": "kms.amazonaws.com",
        "eventName": "EnableKey",
        "awsRegion": "us-east-1",
        "sourceIPAddress": "192.0.2.0",
        "userAgent": "AWS Internal",
        "requestParameters": {
            "keyId": "e17cebae-e7a6-4864-b92f-0365f2feff38"
        },
        "responseElements": null,
        "requestID": "d528a6fb-63bc-11e4-bc2b-4198b6150d5c",
        "eventID": "be393928-3629-4370-9634-567f9274d52e",
        "readOnly": false,
        "resources": [{
            "ARN": "arn:aws:kms:us-east-1:123456789012:key/e17cebae-e7a6-4864-
b92f-0365f2feff38",
            "accountId": "123456789012"
        }],
        "eventType": "AwsApiCall",
        "recipientAccountId": "123456789012"
    }
  ]
}
```


Encrypt

The following example shows a log file generated by calling `Encrypt`.


```
{
    "Records": [
    {
        "eventVersion": "1.02",
        "userIdentity": {
            "type": "IAMUser",
            "principalId": "EX_PRINCIPAL_ID",
            "arn": "arn:aws:iam::123456789012:user/Alice",
            "accountId": "123456789012",
            "accessKeyId": "EXAMPLE_KEY_ID",
            "userName": "Alice"
        },
        "eventTime": "2014-11-04T00:53:11Z",
        "eventSource": "kms.amazonaws.com",
        "eventName": "Encrypt",
        "awsRegion": "us-east-1",
        "sourceIPAddress": "192.0.2.0",
        "userAgent": "AWS Internal",
        "requestParameters": {
```

```
            "encryptionContext": {
                "ContextKey1": "Value1"
            },
          "keyId": "arn:aws:kms:us-east-1:012345678901:key/8d3acf57-6bba-480a-
9459-ed1b8e79d3d0"
        },
        "responseElements": null,
        "requestID": "f3423043-63bc-11e4-bc2b-4198b6150d5c",
        "eventID": "91235988-eb87-476a-ac2c-0cdc244e6dca",
        "readOnly": true,
        "resources": [{
            "ARN": "arn:aws:kms:us-east-1:012345678901:key/8d3acf57-6bba-480a-
9459-ed1b8e79d3d0",
            "accountId": "012345678901"
        }],
        "eventType": "AwsServiceEvent",
        "recipientAccountId": "012345678901"
    }
  ]
}
```

GenerateDataKey

The following example shows a log file created by calling `GenerateDataKey`.

```
{
    "Records": [
    {
        "eventVersion": "1.02",
        "userIdentity": {
            "type": "IAMUser",
            "principalId": "EX_PRINCIPAL_ID",
            "arn": "arn:aws:iam::123456789012:user/Alice"",
            "accountId": "123456789012",
            "accessKeyId": "EXAMPLE_KEY_ID",
            "userName": "Alice"
        },
        "eventTime": "2014-11-04T00:52:40Z",
        "eventSource": "kms.amazonaws.com",
        "eventName": "GenerateDataKey",
        "awsRegion": "us-east-1",
        "sourceIPAddress": "192.0.2.0",
        "userAgent": "AWS Internal",
        "requestParameters": {
            "keyId": "637e8678-3d08-4922-a650-e77eb1591db5",
            "numberOfBytes": 32
        },
        "responseElements": null,
        "requestID": "e0eb83e3-63bc-11e4-bc2b-4198b6150d5c",
        "eventID": "a9dea4f9-8395-46c0-942c-f509c02c2b71",
        "readOnly": true,
        "resources": [{
            "ARN": "arn:aws:kms:us-east-1:123456789012:key/637e8678-3d08-4922-
```

```
a650-e77eb1591db5",
            "accountId": "123456789012"
        }],
        "eventType": "AwsApiCall",
        "recipientAccountId": "123456789012"
    }
  ]
}
```

GenerateDataKeyWithoutPlaintext

The following example shows a log file created by calling `GenerateDataKeyWithoutPlaintext`.

```
{
    "Records": [
    {
        "eventVersion": "1.02",
        "userIdentity": {
            "type": "IAMUser",
            "principalId": "EX_PRINCIPAL_ID",
            "arn": "arn:aws:iam::123456789012:user/Alice",
            "accountId": "123456789012",
            "accessKeyId": "EXAMPLE_KEY_ID",
            "userName": "Alice"
        },
        "eventTime": "2014-11-04T00:52:23Z",
        "eventSource": "kms.amazonaws.com",
        "eventName": "GenerateDataKeyWithoutPlaintext",
        "awsRegion": "us-east-1",
        "sourceIPAddress": "192.0.2.0",
        "userAgent": "AWS Internal",
        "errorCode": "InvalidKeyUsageException",
        "requestParameters": {
            "keyId": "d4f2a88d-5f9c-4807-b71d-4d0ee5225156",
            "numberOfBytes": 16
        },
        "responseElements": null,
        "requestID": "d6b8e411-63bc-11e4-bc2b-4198b6150d5c",
        "eventID": "f7734272-9ec5-4c80-9f36-528ebbe35e4a",
        "readOnly": true,
        "resources": [{
            "ARN": "arn:aws:kms:us-east-1:123456789012:key/d4f2a88d-5f9c-4807-
b71d-4d0ee5225156",
            "accountId": "123456789012"
        }],
        "eventType": "AwsApiCall",
        "recipientAccountId": "123456789012"
    }
  ]
}
```

GenerateRandom

The following example shows a log file created by calling `GenerateRandom`.


```
{
    "Records": [
    {
        "eventVersion": "1.02",
        "userIdentity": {
            "type": "IAMUser",
            "principalId": "EX_PRINCIPAL_ID",
            "arn": "arn:aws:iam::123456789012:user/Alice",
            "accountId": "123456789012",
            "accessKeyId": "EXAMPLE_KEY_ID",
            "userName": "Alice"
        },
        "eventTime": "2014-11-04T00:52:37Z",
        "eventSource": "kms.amazonaws.com",
        "eventName": "GenerateRandom",
        "awsRegion": "us-east-1",
        "sourceIPAddress": "192.0.2.0",
        "userAgent": "AWS Internal",
        "requestParameters": null,
        "responseElements": null,
        "requestID": "df1e3de6-63bc-11e4-bc2b-4198b6150d5c",
        "eventID": "239cb9f7-ae05-4c94-9221-6ea30eef0442",
        "readOnly": true,
        "resources": [],
        "eventType": "AwsApiCall",
        "recipientAccountId": "123456789012"
    }
  ]
}
```


GetKeyPolicy

The following example shows a log file generated by calling `GetKeyPolicy`.


```
{
    "Records": [
    {
        "eventVersion": "1.02",
        "userIdentity": {
            "type": "IAMUser",
            "principalId": "EX_PRINCIPAL_ID",
            "arn": "arn:aws:iam::123456789012:user/Alice",
            "accountId": "123456789012",
            "accessKeyId": "EXAMPLE_KEY_ID",
            "userName": "Alice"
        },
        "eventTime": "2014-11-04T00:50:30Z",
```

```
        "eventSource": "kms.amazonaws.com",
        "eventName": "GetKeyPolicy",
        "awsRegion": "us-east-1",
        "sourceIPAddress": "192.0.2.0",
        "userAgent": "AWS Internal",
        "requestParameters": {
          "keyId": "arn:aws:kms:us-east-1:123456789012:key/e923fe55-d3ef-4f9c-
89a1-2752f98c3a70",
            "policyName": "default"
        },
        "responseElements": null,
        "requestID": "93746dd6-63bc-11e4-bc2b-4198b6150d5c",
        "eventID": "4aa7e4d5-d047-452a-a5a6-2cce282a7e82",
        "readOnly": true,
        "resources": [{
            "ARN": "arn:aws:kms:us-east-1:123456789012:key/e923fe55-d3ef-4f9c-
89a1-2752f98c3a70",
            "accountId": "123456789012"
        }],
        "eventType": "AwsApiCall",
        "recipientAccountId": "123456789012"
    }
  ]
}
```

ListAliases

The following example shows a log file generated by calling `ListAliases`.

```
{
    "Records": [
    {
        "eventVersion": "1.02",
        "userIdentity": {
            "type": "IAMUser",
            "principalId": "EX_PRINCIPAL_ID",
            "arn": "arn:aws:iam::123456789012:user/Alice",
            "accountId": "123456789012",
            "accessKeyId": "EXAMPLE_KEY_ID",
            "userName": "Alice"
        },
        "eventTime": "2014-11-04T00:51:45Z",
        "eventSource": "kms.amazonaws.com",
        "eventName": "ListAliases",
        "awsRegion": "us-east-1",
        "sourceIPAddress": "192.0.2.0",
        "userAgent": "AWS Internal",
        "requestParameters": {
            "limit": 5,
            "marker": "eyJiIjoiYWxpYXMvZTU0Y2MxOTMtYTMwNC00YzEwLTl
iZWItYTJjZjA3NjA2OTJhIiwiYSI6ImFsaWFzL2U1NGNjMTkzLWEzMDQtNGMxMC05YmViLWEyY2YwN
zYwNjkyYSJ9"
        },
```

```
        "responseElements": null,
        "requestID": "bfe6c190-63bc-11e4-bc2b-4198b6150d5c",
        "eventID": "a27dda7b-76f1-4ac3-8b40-42dfba77bcd6",
        "readOnly": true,
        "resources": [],
        "eventType": "AwsApiCall",
        "recipientAccountId": "123456789012"
    }
  ]
}
```

ListGrants

The following example shows a log file generated by calling `ListGrants`.

```
{
    "Records": [
    {
        "eventVersion": "1.02",
        "userIdentity": {
            "type": "IAMUser",
            "principalId": "EX_PRINCIPAL_ID",
            "arn": "arn:aws:iam::123456789012:user/Alice",
            "accountId": "123456789012",
            "accessKeyId": "EXAMPLE_KEY_ID",
            "userName": "Alice"
        },
        "eventTime": "2014-11-04T00:52:49Z",
        "eventSource": "kms.amazonaws.com",
        "eventName": "ListGrants",
        "awsRegion": "us-east-1",
        "sourceIPAddress": "192.0.2.0",
        "userAgent": "AWS Internal",
        "requestParameters": {
          "keyId": "arn:aws:kms:us-east-1:123456789012:key/ea22a751-e707-40d0-
92ac-13a28fa9eb11",
            "marker": "eyJncmFudElkIjoiMWY4M2U2ZmM0YTY2NDgxYjQ2Yzc4MTdhM2Y4Ym
QwMDFkZDNiYmQ1MGVlYTMyY2RmOWFiNWY1Nzc1NDNjYmNmMyIsImtleUFybiI6ImFybjphd3M6dHJlb
nQtc2FuZGJveDp1cy1lYXN0LTE6NTc4Nzg3Njk2NTM
wOmtleS9lYTIyYTc1MS1lNzA3LTQwZDAtOTJhYy0xM2EyOGZhOWViMTEifQ\u003d\u003d",
            "limit": 10
        },
        "responseElements": null,
        "requestID": "e5c23960-63bc-11e4-bc2b-4198b6150d5c",
        "eventID": "d24380f5-1b20-4253-8e92-dd0492b3bd3d",
        "readOnly": true,
        "resources": [{
            "ARN": "arn:aws:kms:us-east-1:123456789012:key/ea22a751-e707-40d0-
92ac-13a28fa9eb11",
            "accountId": "123456789012"
        }],
        "eventType": "AwsApiCall",
        "recipientAccountId": "123456789012"
```

```
    }
  ]
}
```

ReEncrypt

The following example shows a log file generated by calling `ReEncrypt`.

```
{
    "Records": [
    {
        "eventVersion": "1.02",
        "userIdentity": {
            "type": "IAMUser",
            "principalId": "EX_PRINCIPAL_ID",
            "arn": "arn:aws:iam::123456789012:user/Alice",
            "accountId": "123456789012",
            "accessKeyId": "EXAMPLE_KEY_ID",
            "userName": "Alice"
        },
        "eventTime": "2014-11-04T00:52:19Z",
        "eventSource": "kms.amazonaws.com",
        "eventName": "ReEncrypt",
        "awsRegion": "us-east-1",
        "sourceIPAddress": "192.0.2.0",
        "userAgent": "AWS Internal",
        "requestParameters": {
          "destinationKeyId": "arn:aws:kms:us-east-1:123456789012:key/116b8956-
a086-40f1-96d6-4858ef794ba5"
        },
        "responseElements": null,
        "requestID": "d3eeee63-63bc-11e4-bc2b-4198b6150d5c",
        "eventID": "627c13b4-8791-4983-a80b-4c28807b964c",
        "readOnly": false,
        "resources": [{
            "ARN": "arn:aws:kms:us-east-1:123456789012:key/ff0c0fc1-cbaa-41ab-
a267-69481da8a4c8",
            "accountId": "123456789012"
        },
        {
            "ARN": "arn:aws:kms:us-east-1:123456789012:key/116b8956-a086-40f1-
96d6-4858ef794ba5",
            "accountId": "123456789012"
        }],
        "eventType": "AwsServiceEvent",
        "recipientAccountId": "123456789012"
    }
  ]
}
```

Amazon EC2 Example One

The following example demonstrates an IAM user creating an encrypted volume using the default volume key in the Amazon EC2 management console. The

The following example shows a CloudTrail log entry that demonstrates the user Alice creating an encrypted volume using a default volume key in AWS EC2 Management Console. The EC2 log file record includes a `volumeId` field with a value of `"vol-13439757"`. The AWS KMS record contains an `encryptionContext` field with a value of `"aws:ebs:id": "vol-13439757"`. Similarly, the `principalId` and `accountId` between the two records match. The records reflect the fact that creating an encrypted volume generates a data key that is used to encrypt the volume content.


```
{
  "Records": [
     {
      "eventVersion": "1.02",
      "userIdentity": {
        "type": "IAMUser",
        "principalId": "EX_PRINCIPAL_ID",
        "arn": "arn:aws:iam::123456789012:user/Alice",
        "accountId": "123456789012",
        "accessKeyId": "EXAMPLE_KEY_ID",
        "userName": "Alice",
        "sessionContext": {
          "attributes": {
            "mfaAuthenticated": "false",
            "creationDate": "2014-11-05T20:40:44Z"
          }
        },
        "invokedBy": "signin.amazonaws.com"
      },
      "eventTime": "2014-11-05T20:50:18Z",
      "eventSource": "ec2.amazonaws.com",
      "eventName": "CreateVolume",
      "awsRegion": "us-east-1",
      "sourceIPAddress": "72.72.72.72",
      "userAgent": "signin.amazonaws.com",
      "requestParameters": {
        "size": "10",
        "zone": "us-east-1a",
        "volumeType": "gp2",
        "encrypted": true
      },
      "responseElements": {
        "volumeId": "vol-13439757",
        "size": "10",
        "zone": "us-east-1a",
        "status": "creating",
        "createTime": 1415220618876,
        "volumeType": "gp2",
        "iops": 30,
        "encrypted": true
      },
      "requestID": "1565210e-73d0-4912-854c-b15ed349e526",
      "eventID": "a3447186-135f-4b00-8424-bc41f1a93b4f",
      "eventType": "AwsApiCall",
```

```
      "recipientAccountId": "123456789012"
    },
    {
      "eventVersion": "1.02",
      "userIdentity": {
        "type": "IAMUser",
        "principalId": "EX_PRINCIPAL_ID",
        "arn": "arn:aws:iam::123456789012:user/Alice",
        "accountId": "123456789012",
        "accessKeyId": "EXAMPLE_KEY_ID",
        "userName": "Alice",
        "sessionContext": {
          "attributes": {
            "mfaAuthenticated": "false",
            "creationDate": "2014-11-05T20:40:44Z"
          }
        },
        "invokedBy": "AWS Internal"
      },
      "eventTime": "2014-11-05T20:50:19Z",
      "eventSource": "kms.amazonaws.com",
      "eventName": "GenerateDataKeyWithoutPlaintext",
      "awsRegion": "us-east-1",
      "sourceIPAddress": "AWS Internal",
      "userAgent": "AWS Internal",
      "requestParameters": {
        "encryptionContext": {
          "aws:ebs:id": "vol-13439757"
        },
        "numberOfBytes": 64,
        "keyId": "alias/aws/ebs"
      },
      "responseElements": null,
      "requestID": "create-123456789012-758241111-1415220618",
      "eventID": "4bd2a696-d833-48cc-b72c-05e61b608399",
      "readOnly": true,
      "resources": [
        {
          "ARN": "arn:aws:kms:us-east-1:123456789012:key/e29ddfd4-1bf6-4e1b-
8ecb-08216bd70d07",
          "accountId": "123456789012"
        }
      ],
      "eventType": "AwsApiCall",
      "recipientAccountId": "123456789012"
    },
    ... additional entries ...
  ]
}
```


Amazon EC2 Example Two

The following example shows an IAM user running an Amazon EC2 instance that mounts a data volume encrypted by using a default volume key. The action taken by the user generates multiple AWS KMS log records. Creating the encrypted volume generates a datakey, and the Amazon EC2 service generates a

grant, on behalf of the customer, that enables it to decrypt the data key. The `instanceId`, `"i-81e2f56c"`, is referred to in the `granteePrincipal` field of the `CreateGrant` record as `"123456789012:aws:ec2-infrastructure:i-81e2f56c"` as well as in the identity of the principal calling Decrypt, `“arn:aws:sts::123456789012:assumed-role/aws:ec2-infrastructure/i-81e2f56c”`. The key identified by the UUID `“e29ddfd4-1bf6-4e1b-8ecb-08216bd70d07”` is common across all three KMS calls.


```
{
  "Records": [
    {
      "eventVersion": "1.02",
      "userIdentity": {
        "type": "IAMUser",
        "principalId": "EX_PRINCIPAL_ID",
        "arn": "arn:aws:iam::123456789012:user/Alice",
        "accountId": "123456789012",
        "accessKeyId": "EXAMPLE_KEY_ID",
        "userName": "Alice",
        "sessionContext": {
          "attributes": {
            "mfaAuthenticated": "false",
            "creationDate": "2014-11-05T21:34:36Z"
          }
        },
        "invokedBy": "signin.amazonaws.com"
      },
      "eventTime": "2014-11-05T21:35:27Z",
      "eventSource": "ec2.amazonaws.com",
      "eventName": "RunInstances",
      "awsRegion": "us-east-1",
      "sourceIPAddress": "72.72.72.72",
      "userAgent": "signin.amazonaws.com",
      "requestParameters": {
        "instancesSet": {
          "items": [
            {
              "imageId": "ami-b66ed3de",
              "minCount": 1,
              "maxCount": 1
            }
          ]
        },
        "groupSet": {
          "items": [
            {
              "groupId": "sg-98b6e0f2"
            }
          ]
        },
        "instanceType": "m3.medium",
        "blockDeviceMapping": {
          "items": [
            {
              "deviceName": "/dev/xvda",
              "ebs": {
                "volumeSize": 8,
```

```
            "deleteOnTermination": true,
            "volumeType": "gp2"
          }
        },
        {
          "deviceName": "/dev/sdb",
          "ebs": {
            "volumeSize": 8,
            "deleteOnTermination": false,
            "volumeType": "gp2",
            "encrypted": true
          }
        }
      ]
    },
    "monitoring": {
      "enabled": false
    },
    "disableApiTermination": false,
    "instanceInitiatedShutdownBehavior": "stop",
    "clientToken": "XdKUT141516171819",
    "ebsOptimized": false
  },
  "responseElements": {
    "reservationId": "r-5ebc9f74",
    "ownerId": "123456789012",
    "groupSet": {
      "items": [
        {
          "groupId": "sg-98b6e0f2",
          "groupName": "launch-wizard-2"
        }
      ]
    },
    "instancesSet": {
      "items": [
        {
          "instanceId": "i-81e2f56c",
          "imageId": "ami-b66ed3de",
          "instanceState": {
            "code": 0,
            "name": "pending"
          },
          "amiLaunchIndex": 0,
          "productCodes": {

          },
          "instanceType": "m3.medium",
          "launchTime": 1415223328000,
          "placement": {
            "availabilityZone": "us-east-1a",
            "tenancy": "default"
          },
          "monitoring": {
            "state": "disabled"
          },
          "stateReason": {
            "code": "pending",
```

```
            "message": "pending"
          },
          "architecture": "x86_64",
          "rootDeviceType": "ebs",
          "rootDeviceName": "/dev/xvda",
          "blockDeviceMapping": {

          },
          "virtualizationType": "hvm",
          "hypervisor": "xen",
          "clientToken": "XdKUT1415223327917",
          "groupSet": {
            "items": [
              {
                "groupId": "sg-98b6e0f2",
                "groupName": "launch-wizard-2"
              }
            ]
          },
          "networkInterfaceSet": {

          },
          "ebsOptimized": false
        }
      ]
    }
  },
  "requestID": "41c4b4f7-8bce-4773-bf0e-5ae3bb5cbce2",
  "eventID": "cd75a605-2fee-4fda-b847-9c3d330ebaae",
  "eventType": "AwsApiCall",
  "recipientAccountId": "123456789012"
},
{
  "eventVersion": "1.02",
  "userIdentity": {
    "type": "IAMUser",
    "principalId": "EX_PRINCIPAL_ID",
    "arn": "arn:aws:iam::123456789012:user/Alice",
    "accountId": "123456789012",
    "accessKeyId": "EXAMPLE_KEY_ID",
    "userName": "Alice",
    "sessionContext": {
      "attributes": {
        "mfaAuthenticated": "false",
        "creationDate": "2014-11-05T21:34:36Z"
      }
    },
    "invokedBy": "AWS Internal"
  },
  "eventTime": "2014-11-05T21:35:35Z",
  "eventSource": "kms.amazonaws.com",
  "eventName": "CreateGrant",
  "awsRegion": "us-east-1",
  "sourceIPAddress": "AWS Internal",
  "userAgent": "AWS Internal",
  "requestParameters": {
    "constraints": {
      "encryptionContextSubset": {
```

```
            "aws:ebs:id": "vol-f67bafb2"
          }
        },
        "granteePrincipal": "123456789012:aws:ec2-infrastructure:i-81e2f56c",
        "keyId": "arn:aws:kms:us-east-1:123456789012:key/e29ddfd4-1bf6-4e1b-
8ecb-08216bd70d07"
      },
      "responseElements": {
        "grantId":
"6caf442b4ff8a27511fb6de3e12cc5342f5382112adf75c1a91dbd221ec356fe"
      },
      "requestID": "41c4b4f7-8bce-4773-bf0e-5ae3bb5cbce2",
      "eventID": "c1ad79e3-0d3f-402a-b119-d5c31d7c6a6c",
      "readOnly": false,
      "resources": [
        {
          "ARN": "arn:aws:kms:us-east-1:123456789012:key/e29ddfd4-1bf6-4e1b-
8ecb-08216bd70d07",
          "accountId": "123456789012"
        }
      ],
      "eventType": "AwsApiCall",
      "recipientAccountId": "123456789012"
    },
    {
      "eventVersion": "1.02",
      "userIdentity": {
        "type": "IAMUser",
        "principalId": "EX_PRINCIPAL_ID",
        "arn": "arn:aws:iam::123456789012:user/Alice",
        "accountId": "123456789012",
        "accessKeyId": "EXAMPLE_KEY_ID",
        "userName": "Alice",
        "sessionContext": {
          "attributes": {
            "mfaAuthenticated": "false",
            "creationDate": "2014-11-05T21:34:36Z"
          }
        },
        "invokedBy": "AWS Internal"
      },
      "eventTime": "2014-11-05T21:35:32Z",
      "eventSource": "kms.amazonaws.com",
      "eventName": "GenerateDataKeyWithoutPlaintext",
      "awsRegion": "us-east-1",
      "sourceIPAddress": "AWS Internal",
      "userAgent": "AWS Internal",
      "requestParameters": {
        "encryptionContext": {
          "aws:ebs:id": "vol-f67bafb2"
        },
        "numberOfBytes": 64,
        "keyId": "alias/aws/ebs"
      },
      "responseElements": null,
      "requestID": "create-123456789012-758247346-1415223332",
      "eventID": "ac3cab10-ce93-4953-9d62-0b6e5cba651d",
      "readOnly": true,
```

```
      "resources": [
        {
          "ARN": "arn:aws:kms:us-east-1:123456789012:key/e29ddfd4-1bf6-4e1b-
8ecb-08216bd70d07",
          "accountId": "123456789012"
        }
      ],
      "eventType": "AwsApiCall",
      "recipientAccountId": "123456789012"
    },
    {
      "eventVersion": "1.02",
      "userIdentity": {
        "type": "AssumedRole",
        "principalId": "123456789012:aws:ec2-infrastructure:i-81e2f56c",
       "arn": "arn:aws:sts::123456789012:assumed-role/aws:ec2-infrastructure/i-
81e2f56c",
        "accountId": "123456789012",
        "accessKeyId": "
",
        "sessionContext": {
          "attributes": {
            "mfaAuthenticated": "false",
            "creationDate": "2014-11-05T21:35:38Z"
          },
          "sessionIssuer": {
            "type": "Role",
            "principalId": "123456789012:aws:ec2-infrastructure",
            "arn": "arn:aws:iam::123456789012:role/aws:ec2-infrastructure",
            "accountId": "123456789012",
            "userName": "aws:ec2-infrastructure"
          }
        }
      },
      "eventTime": "2014-11-05T21:35:47Z",
      "eventSource": "kms.amazonaws.com",
      "eventName": "Decrypt",
      "awsRegion": "us-east-1",
      "sourceIPAddress": "172.172.172.172",
      "requestParameters": {
        "encryptionContext": {
          "aws:ebs:id": "vol-f67bafb2"
        }
      },
      "responseElements": null,
      "requestID": "b4b27883-6533-11e4-b4d9-751f1761e9e5",
      "eventID": "edb65380-0a3e-4123-bbc8-3d1b7cff49b0",
      "readOnly": true,
      "resources": [
        {
          "ARN": "arn:aws:kms:us-east-1:123456789012:key/e29ddfd4-1bf6-4e1b-
8ecb-08216bd70d07",
          "accountId": "123456789012"
        }
      ],
      "eventType": "AwsApiCall",
      "recipientAccountId": "123456789012"
    },
```

```
...
  ]
}
```

Programming the AWS KMS API

You can use the AWS KMS API to perform the following actions:

- Create, describe, list, enable, and disable keys.
- Create, delete, list, and update aliases.
- Encrypt, decrypt, and re-encrypt content.
- Set, list, and retrieve key policies.
- Create, retire, revoke, and list grants.
- Retrieve key rotation status.
- Update key descriptions.
- Generate data keys with or without plaintext.
- Generate random data.

Topics

Creating a Client

Before you can program to the AWS Key Management Service, you must create a client, as shown by the following example. The client object, kms, is used throughout all of the code in the sections that follow.

```
package com.amazon.kms;

import java.io.File;
import java.io.IOException;
import java.nio.ByteBuffer;
```

```
import com.amazonaws.auth.AWSCredentials;
import com.amazonaws.auth.PropertiesCredentials;
import com.amazonaws.services.kms.AWSKMS;
import com.amazonaws.services.kms.AWSKMSClient;
import com.amazonaws.services.kms.model.*;

public class kmsSDKExample {

    private final AWSKMSClient kms;

    public kmsSDKExample() {
        kms = getClient();
    }

    public static void main(String[] args) {
        new kmsSDKExample();
    }

    private AWSKMS getClient() {
        final AWSCredentials creds;

        AWSKMSClient kms = new AWSKMSClient(creds);

        .
        .
        .

        kms.setEndpoint("https://kms.us-east-1.amazonaws.com");

        return kms;
    }

    .
    .
    .

}
```

Working With Keys

This topic discusses how to create, describe, list, enable, and disable keys.

Creating a Customer Master Key

Call the `CreateKey` function to create a customer master key. The function takes three optional parameters, as shown in the following example.

```
// Creating a key.
//
// Input Parameters:
//   The function takes three optional parameters.
//      Description  - Contains a string description for the key
//      KeyUsage     - Use the default value (ENCRYPT_DECRYPT)
```

```
//      Policy       - Use the default policy, which grants rights to all key
actions
//
// Return Values:
//   The function returns a CreateKeyResult structure that contains the follow
ing:
//      AWSAccountId - Account ID of the account the key is associated with
//      ARN          - Amazon Resource Name for the key
//      CreationDate - Date the key was created in UTC format
//      Description  - Key description
//     Enabled      - A Boolean value that specifies whether the key is enabled
//      KeyID        - A unique value that can be used to identify the key in
other operations
//      KeyUsage     - A value that shows what the key can be used for
//
String desc = "Key for protecting critical data";

CreateKeyRequest req = new CreateKeyRequest().withDescription(desc);
CreateKeyResult result = kms.createKey(req);
```

Generating a Data Key

Call the `GenerateDataKey` function to create a data key. The function takes up to five parameters, as shown in the following example.

```
// Generate a data key
//
// Input Parameters:
//   The function takes five parameters.
//     KeyId             - Unique identifier for the key to be used for encryp
tion
//     EncryptionContext - Authenticated data
//     NumberOfBytes     - The number of bytes of data key being requested
//     KeySpec           - The key specification being requested ("AES_128" or
 "AES_256")
//     GrantTokens       - List of grant tokens
//
//  Return Values:
//    The function returns a byte buffer that contains the encrypted key, a
byte buffer
//    of the plaintext key, and the KeyID of the master key under which the key
 is encrypted.
//
//
String keyId = "arn:aws:kms:us-east-1:123456789012:key/12345678-1234-1234-1234-
123456789012";

GenerateDataKeyRequest dataKeyRequest = new GenerateDataKeyRequest();
dataKeyRequest.setKeyId(keyId);
dataKeyRequest.setKeySpec("AES_128");

GenerateDataKeyResult dataKeyResult = kmsClient.generateDataKey(dataKeyRequest);

ByteBuffer plaintextKey = dataKeyResult.getPlaintext();
```

```
ByteBuffer encryptedKey = dataKeyResult.getCiphertextBlob();
```

Describing a Key

Call the `DescribeKey` function to retrieve detailed information about a customer master key.

```
// Describing a key.
//
// Input Parameters:
//   The function takes one required parameter.
//      KeyId        - Unique identifier of the key. This can be an ARN, an
alias, or a globally unique
//                     identifier.
//
// Return Values:
//   The function returns a DescribeKeyResult object that contains metadata
about
//   the key.
//      AWSAccountId - ID of the account the key is associated with
//      ARN          - Amazon Resource Name for the key
//      CreationDate - Date the key was created in UTC format
//      Description  - Key description
//     Enabled      - A Boolean value that specifies whether the key is enabled
//      KeyId        - A unique value that can be used to identify the key in
other operations
//      KeyUsage     - A value that shows what the key can be used for
//
String keyId = "arn:aws:kms:us-east-1:123456789012:key/12345678-1234-1234-1234-
123456789012";

DescribeKeyRequest req = new DescribeKeyRequest().withKeyId(keyId);
DescribeKeyResult result = kms.describeKey(req);
```

Listing Keys

Call the `ListKeys` function to retrieve a list of the customer master keys.

```
// Listing keys.
//
// Input Parameters:
//   The function takes two required parameters.
//      Limit      -  Specify this parameter only when paginating results to
indicate the
//                    maximum number of keys you want listed in the response.
If there are
//                    additional keys beyond the maximum you specify, the
Truncated
//                    response element will be set to true.
//      Marker     -  Use this parameter only when paginating results, and only
```

```
 in a subsequent
//                   request after you've received a response where the results
 are truncated.
//                    Set it to the value of the NextMarker in the response you

//                     just received.
//
// Return Values:
//   The function returns a ListKeyResult object that contains the following
//   values:
//      Keys       - A list of keys
//      NextMarker - If Truncated is true, this value is present and contains
the value
//                     to use for the Marker request parameter in a subsequent
pagination
//                     request.
//      Truncated  - A flag that indicates whether there are more items in the
 list. If your results
//                  were truncated, you can make a subsequent pagination request
 using the
//                   Marker request parameter to retrieve more keys in the list.

//
Integer limit = 10;
String marker = null;

ListKeysRequest req = new ListKeysRequest().withMarker(marker).withLimit(limit);
ListKeysResult result = kms.listKeys(req);
```

Enabling Keys

Call the `EnableKey` function to mark a key as enabled.

```
// Enabling a key.
//
// Input Parameters:
//   The function takes one required parameter.
//      KeyId    - Unique identifier of the customer master key to be enabled.
 This can be an
//                 ARN, an alias, or a globally unique identifier.
//
// Return Values:
//   The function does not return a value.
//
String keyId = "arn:aws:kms:us-east-1:123456789012:key/12345678-1234-1234-1234-
123456789012";

EnableKeyRequest req = new EnableKeyRequest().withKeyId(keyId);
kms.enableKey(req);
```

Disabling Keys

Call the `DisableKey` function to prevent a key from being used.

```
// Disabling a key.
//
// Input Parameters:
//   The function takes one required parameter.
//      KeyId    - Unique identifier of the customer master key to be disabled.
 This can be an
//                 ARN, an alias, or a globally unique identifier.
//
// Return Values:
//   The function does not return a value.
//
String keyId = "arn:aws:kms:us-east-1:123456789012:key/12345678-1234-1234-1234-
123456789012";;

DisableKeyRequest req = new DisableKeyRequest().withKeyId(keyId);
kms.disableKey(req);
```

Encrypting and Decrypting Data

This topic discusses how to encrypt, decrypt, and re-encrypt content.

Encrypting Data

Call the `Encrypt` function to encrypt plaintext data.

```
// Encrypting content
//
// Input Parameters:
//   The function takes four parameters.
//     KeyId             - Unique identifier for the key to be used for encryp
tion
//     Plaintext        - Byte buffer that contains the content to be encrypted
//     EncryptionContext - Authenticated data
//     GrantTokens       - List of grant tokens
//
//  Return Values:
//    The function returns a byte buffer that contains the encrypted content
and the key ID
//    of the master key used.
//
String keyId = "arn:aws:kms:us-east-1:123456789012:key/12345678-1234-1234-1234-
123456789012";
ByteBuffer plaintext = ByteBuffer.wrap(new byte[]{1,2,3,4,5,6,7,8,9,0});

EncryptRequest req = new EncryptRequest().withKeyId(keyId).withPlaintext(plain
text);
```

```
ByteBuffer ciphertext = kms.encrypt(req).getCiphertextBlob();
```

Decrypting Data

Call the `Decrypt` function to decrypt ciphertext. The data to decrypt must be valid ciphertext that you receive from the `Encrypt` function.

```
// Decrypting content
//
// Input Parameters:
//   The function takes three parameters.
//     CipherTextBlob    - Ciphertext to be decrypted
//     EncryptionContext - Authenticated data
//     GrantTokens       - List of grant tokens
//
// Return Values:
//   The function returns a byte buffer that contains the decrypted content.
//
ByteBuffer ciphertextBlob = ciphertext here;

DecryptRequest req = new DecryptRequest().withCiphertextBlob(ciphertextBlob);
ByteBuffer plainText = kms.decrypt(req).getPlaintext();
```

Re-Encrypting Data Under a Different Key

Call the `ReEncrypt` function to encrypt previously encrypted data by using a new key. This function decrypts your ciphertext and re-encrypts it by using a different key that you specify. The function never exposes your plaintext outside of AWS KMS.

```
// ReEncrypt content
// Input parameters:
//   The function takes three parameters.
//     CipherTextBlob              - Ciphertext to be re-encrypted
//     SourceEncryptionContext     - Authenticated data used for the original
 encryption
//     DestinationKeyId            - Key identifier for the re-encrypted data
//     DestinationEncryptionContext - encryption context for the re-encrypted
data
//     GrantTokens                 - List of grant tokens
//
// Return Values:
//   The function returns a byte buffer that contains the re-encrypted content.
//
ByteBuffer sourceCiphertextBlob = ciphertext here
String destinationKeyId = "arn:aws:kms:us-east-1:123456789012:key/12345678-1234-
1234-1234-123456789012";

ReEncryptRequest req = new ReEncryptRequest();
req.setCiphertextBlob(sourceCiphertextBlob);
req.setDestinationKeyId(destinationKeyId);
```

```
ByteBuffer destinationCipherTextBlob = kms.reEncrypt(req).getCiphertextBlob();
```

Working with Key Policies

This topic discusses how to list, retrieve, and set key policies.

Listing Key Policies

Call the `ListKeyPolicies` function to list key policies for a specified key.

```
// Listing key policies
//
// Input Parameters:
//   The function takes three parameters.
//     KeyId    - Unique identifier of the key for which the policies are to
be listed.
//    Limit    - Specify this parameter only when paginating results to indicate
 the
//                maximum number of policies you want listed in the response.
If there are
//                additional policies beyond the maximum you specify, the
Truncated
//                response element will be set to true.
//     Marker   - Use this parameter only when paginating results, and only in
 a subsequent
//                request after you've received a response where the results
are truncated.
//                Set it to the value of the NextMarker in the response you
//                just received.
//
// Return Values:
//   The function returns a list of key policies.
//
String keyId = "arn:aws:kms:us-east-1:123456789012:key/12345678-1234-1234-1234-
123456789012";
String marker = null;
Integer limit = 10;

ListKeyPoliciesRequest req = new ListKeyPoliciesRequest().withKeyId(keyId).with
Marker(marker).withLimit(limit);
ListKeyPoliciesResult result = kms.listKeyPolicies(req);
```

Retrieving a Key Policy

Call the `GetKeyPolicy` function to retrieve a key policy.

```
// Retrieving a key policy
```

```
//
// Input Parameters:
//   The function takes two parameters.
//     KeyId        - Unique identifier of the key for which the policy will
be returned.
//     PolicyName   - Name of the policy to return.
//
// Return Values:
//   The function returns a key policy.
//
String keyId = "arn:aws:kms:us-east-1:123456789012:key/12345678-1234-1234-1234-
123456789012";
String policyName = "default";

GetKeyPolicyRequest req = new GetKeyPolicyRequest().withKeyId(keyId).withPoli
cyName(policyName);
GetKeyPolicyResult result = kms.getKeyPolicy(req);
```

Setting a Key Policy

Call the `PutKeyPolicy` function to set a policy on a key.

```
// Setting a policy on a key
//
// Input Parameters:
//   The function takes three parameters.
//     KeyId        - Unique identifier of the key
//     PolicyName   - Name of the policy to set
//     Policy       - Policy to set
//
// Return Values:
//   The function does not return a value.
//
String keyId = "arn:aws:kms:us-east-1:123456789012:key/12345678-1234-1234-1234-
123456789012";
String policyName = "default";
String policy = "{ " +
                "   \"Version\":\"2012-10-17\", " +
                "   \"Statement\": [          " +
                "      {                   " +
                "         \"Sid\": \"Allow access for Alice\", " +
                "         \"Effect\": \"Allow\",               " +
                "         \"Principal\":
\"arn:aws:iam::123456789012:user/Alice\", " +
              "        \"Action\": [\"kms:Encrypt\", \"kms:GenerateDataKey*\",
 \"kms:Decrypt\", \"kms:DescribeKey\", \"kms:ReEncrypt*\"], " +
                "         \"Resource\": \"*\" " +
                "      }"+
                "    ]" +
                "}";

PutKeyPolicyRequest req = new PutKeyPolicyRequest().withKeyId(keyId).with
Policy(policy).withPolicyName(policyName);
```

```
kms.putKeyPolicy(req);
```

Working with Grants

This topic discusses how to create, retire, revoke, and list grants.

Creating a Grant

Call the `CreateGrant` function to create a grant.

```
// Creating a grant
//
// Input Parameters:
//   The function takes up to six parameters.
//     KeyId              - Unique identifier for the key. This can be an ARN,
 an alias, or a globally unique value.
//    GranteePrincipal   - Principal given permission to use the key identified
 by the KeyId parameter
//     RetiringPrincipal  - Principal given permission to retire the grant
//     Operations         - List of operations permitted by the grant
//     Constraints        - The conditions under which the actions specified
by the Operations parameter are allowed
//     GrantTokens        - List of grant tokens
//
// Return Values:
//   The function returns two values.
//     GrantToken        - Signed and encrypted string value that contains all
 of the information needed to create the grant
//     GrantID            - Globally unique identifier of the grant
//
String keyId = "arn:aws:kms:us-east-1:123456789012:key/12345678-1234-1234-1234-
123456789012";
String granteePrincipal = "arn:aws:iam::123456789012:user/Alice"
String operation = GrantOperation.Encrypt;

CreateGrantRequest req = new CreateGrantRequest();
req.setKeyId(keyId);
req.setGranteePrincipal(granteePrincipal);
req.setOperation(operation);

CreateGrantResult result = kms.createGrant(req);
```

Retiring a Grant

Call the `RetireGrant` function to retire a grant. You should retire a grant to clean up after you are done using it.

```
// Retiring a grant
```

```
//
// Input Parameters:
//   GrantToken - unique grant identifier
//
// Return Values:
//   The function does not return a value.

String grantToken = grant token here

RetireGrantRequest req = new RetireGrantRequest().withGrantToken(grantToken);
kms.retireGrant(req);
```

Revoking Grants

Call the `RevokeGrant` function to revoke a grant. You should revoke a grant to deny operations that depend on it.

```
// Revoking a grant
//
// Input Parameters:
//   KeyId   - Unique identifier for the key
//   GrantId - Unique identifier for the grant
//
// Return Values:
//   The function does not return a value.
//
String keyId = "arn:aws:kms:us-east-1:123456789012:key/12345678-1234-1234-1234-
123456789012";
String grantId = "grant1";

RevokeGrantRequest req = new RevokeGrantRequest().withKeyId(keyId).withGrant
Id(grantId);
kms.revokeGrant(req);
```

Listing Grants

Call the `ListGrant` function to list all of the grants on a given key.

```
// Listing grants
//
// Input Parameters:
//   The function takes three parameters.
//     KeyId    - Unique identifier for the key
//    Limit    - Specify this parameter only when paginating results to indicate
 the
//               maximum number of grants you want listed in the response. If
 there are
//             additional grants beyond the maximum you specify, the Truncated

//               response element will be set to true.
```

```
//     Marker    - Use this parameter only when paginating results, and only in
 a subsequent
//                 request after you've received a response where the results
are truncated.
//                 Set it to the value of the NextMarker in the response you
//                 just received.
//
// Return Values:
//   The function returns a list of grants for the key.
//
String keyId = "arn:aws:kms:us-east-1:123456789012:key/12345678-1234-1234-1234-
123456789012";
Integer limit = 10;
String marker = null;

ListGrantsRequest req = new ListGrantsRequest().withKeyId(keyId).withMarker(mark
er).withLimit(limit);
ListGrantsResult result = kms.listGrants(req);
```

Working with Aliases

This topic discusses how to create, delete, and update an alias.

An alias is a display name for a key. It can be used to identify a key in the following functions where a KeyId is required:

- **DescribeKey**
- **Encrypt**
- **GenerateDataKey**
- **GenerateDataKeyWithoutPlaintext**
- **ListKeyPolicies**
- **ReEncrypt**

You can use a full ARN to specify an alias or just the alias name as shown in the following example. If you use the alias name, be sure to prepend "alias/" to it.

```
// Fully specified ARN
arn:aws:kms:region:123456789012:alias/MyAliasName

// Alias name (prefixed with "alias/")
alias/MyAliasName
```

An alias is not a property of a key, and therefore can be associated with and disassociated from an existing key without changing the properties of the key. Deleting an alias does not delete the underlying key.

Creating an Alias

Call the `CreateAlias` function to create an alias. The alias should be unique.

```
// Creating an alias
//
// Input Parameters:
//   The function takes two parameters.
//     AliasName    - String that contains a display name for a key. This is
of the format
//                    "alias/[a-zA-Z0-9/_-]+". That is, the alias name can be
an alphanumeric
//                    value and contain an underscore or a dash. Alias names
that begin with
//                    "alias/aws..." are reserved for AWS use.
//     TargetKeyId  - Unique key identifier of the key to which the display
name will
//                    be associated
//
// Return Values:
//   The function does not return a value.
//
String aliasName = "alias/privatekey1";
String targetKeyId = "arn:aws:kms:us-east-1:123456789012:key/12345678-1234-1234-
1234-123456789012";

CreateAliasRequest req = new CreateAliasRequest().withAliasName(ali
asName).withTargetKeyId(targetKeyId);
kms.createAlias(req);
```

Deleting an Alias

Call the `DeleteAlias` function to delete an alias.

```
// Deleting an alias
//
// Input Parameters:
//   The function takes one parameter.
//     AliasName - String that contains a display name for a key
//
// Return Values:
//   The function does not return a value.
//
String aliasName = "alias/privatekey1";

DeleteAliasRequest req  = new DeleteAliasRequest().withAliasName(aliasName);
kms.deleteAlias(req);
```

Listing Aliases

Call the `ListAliases` function to list all of the key aliases for your account.

```
// Listing aliases
//
// Input Parameters:
//   The function takes three parameters.
//    Limit    - Specify this parameter only when paginating results to indicate
 the
//               maximum number of aliases you want listed in the response.
If there are
//               additional aliases beyond the maximum you specify, the Trun
cated
//               response element will be set to true.
//     Marker   - Use this parameter only when paginating results, and only in
 a subsequent
//               request after you've received a response where the results
are truncated.
//               Set it to the value of the NextMarker in the response you
//               just received.
//     Prefix   - Prefix of the alias to list. This value can be null.
//
// Return Values:
//   The function returns a list of aliases for the keys in your account.
//
String prefix = null;
String marker = null;
Integer limit = 10;

ListAliasesRequest req = new ListAliasesRequest().withPrefix(prefix).withMark
er(marker).withLimit(limit);
ListAliasesResult result = kms.listAliases(req);
```

Updating an Alias

Call the `UpdateAlias` function to associate an alias with a different key.

```
// Updating an alias
//
// Input Parameters:
//   The function takes two parameters.
//     AliasName   - String that contains the name of the alias to be modified.
 An alias name can
//                   contain only alphanumeric characters, forward slashes,
underscores, and dashes.
//                   An alias must start with the word "alias" followed by a
forward slash (alias/).
//                   An alias that begins with "aws" after the forward slash
is reserved by
//                   Amazon Web Services (AWS).
//     TargetKeyId - Unique identifier of the customer master key to be associ
ated with the alias.
//                  This value can be a globally unique identifier or the fully
 specified ARN of
//                   a key.
//
```

```
// Return Values:
//   The function does not return a value.
//
String aliasName = "alias/critical_data_protection_key";
String targetKeyId = "arn:aws:kms:us-east-1:123456789012:key/12345678-1234-1234-
1234-123456789012";

UpdateAliasRequest req = new UpdateAliasRequest()
     .withAliasName(aliasName)
     .withTargetKeyId(targetKeyId);

kms.updateAlias(req);
```

Cryptography Basics

This topic discusses terms and concepts in cryptography that you'll encounter when you work with AWS KMS.

Plaintext and Ciphertext

Plaintext refers to information or data in an un-encrypted, or unprotected, form. Ciphertext refers to the output of an encryption algorithm operating on plaintext. AWS KMS uses encryption algorithms. The ciphertext is unreadable without knowledge of the secret key.

Algorithms and Keys

Encryption algorithms used by AWS KMS require a key. An *encryption algorithm* is a step-by-step set of instructions that specify precisely how plaintext is transformed into ciphertext. The Advanced Encryption Standard (AES) is an example of an algorithm that is used in AWS KMS. A *key* is a secret value that the algorithm uses to turn plaintext into ciphertext. Master keys in AWS KMS are 256 bits in length.

Symmetric and Asymmetric Encryption

Encryption can be either symmetric or asymmetric. *Symmetric encryption* uses the same secret key to perform both the encryption and decryption processes. *Asymmetric encryption* uses two keys, a public key and a private key. The public key and the private key are mathematically related but different. If the public key is used for encryption, the private key must be used for decryption.

> **Important**
> AWS KMS currently supports only symmetric (private) key cryptography.

Cryptography is discussed in more detail in the following topics.

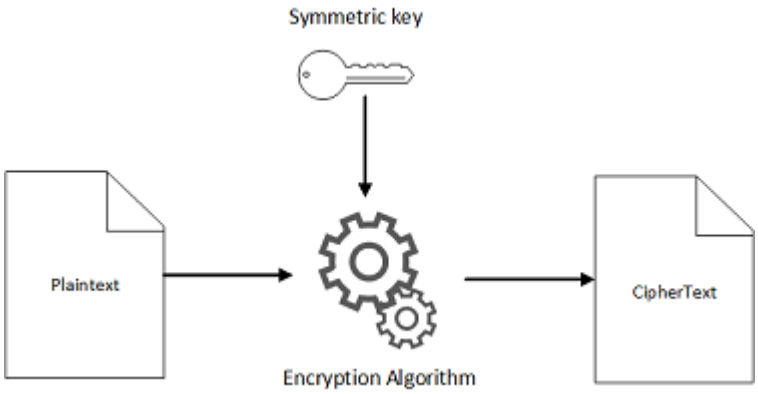
Topics

How Symmetric Key Cryptography Works

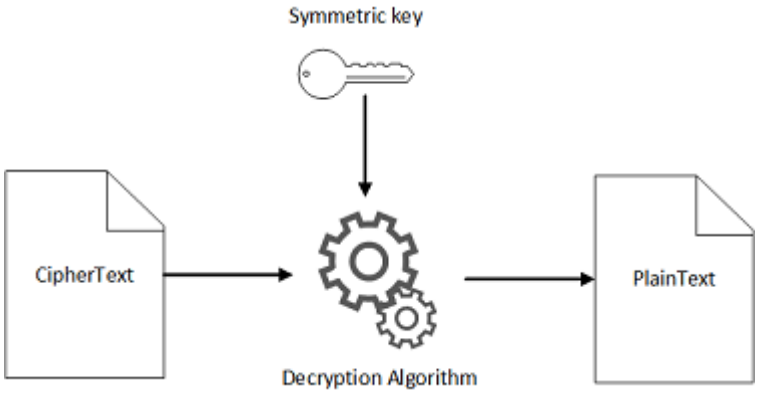
This topic provides a high-level introduction to how symmetric key cryptography uses algorithms to encrypt and decrypt data, the difference between block and stream ciphers, and how block ciphers use encryption modes to expand the effectiveness of the generic encryption schemes.

Encryption and Decryption

AWS KMS uses symmetric key cryptography to perform encryption and decryption. Symmetric key cryptography uses the same algorithm and key to both encrypt and decrypt digital data. The unencrypted data is typically called plaintext whether it is text or not. The encrypted data is typically called ciphertext. The following illustration shows a secret (symmetric) key and a symmetric algorithm being used to turn plaintext into ciphertext.



The next illustration shows the same secret key and symmetric algorithm being used to turn ciphertext back into plaintext.



Authenticated Encryption

Authenticated encryption provides confidentiality, data integrity, and authenticity assurances on encrypted data. The `Encrypt` function takes plaintext, a key identifier, and an encryption context and returns ciphertext. The encryption context represents additional authenticated data (AAD). The encryption process uses the AAD only to generate an authentication tag. The tag is included with the output ciphertext and used as input to the decryption process. This means that the encryption context that you supply to the

`Decrypt` function must be the same as the encryption context you supply to the `Encrypt` function. Otherwise, the tag computed during decryption will not equal the tag computed during encryption, and the decryption process will fail without producing plaintext. Further, if any one of the parameters has been tampered with—specifically if the ciphertext has been altered—the authentication tag will not compute to the same value that it did during encryption. The decryption process will fail and the ciphertext will not be decrypted.

Encryption Context

An *encryption context* is a key/value pair that you can pass to AWS KMS when you call the **Encrypt** function. It is integrity checked but not stored as part of the ciphertext that is returned. Although the encryption context is not literally included in the ciphertext, it is cryptographically bound to the ciphertext during encryption and must be passed again when you call the **Decrypt** function. Decryption will only succeed if the value you pass for decryption is exactly the same as the value you passed during encryption and the ciphertext has not been tampered with. The encryption context is logged by using CloudTrail.

The encryption context can be any value that you want. However, because it is not encrypted and because it is logged if CloudTrail logging is turned on, the encryption context should not include sensitive information. We further recommend that your context describe the data being encrypted or decrypted so that you can better understand the CloudTrail events produced by AWS KMS. For example, Amazon EBS uses the ID of the encrypted volume as the encryption context for server-side encryption. If you are encrypting a file, you might use part of the file path as the encryption context.

Encryption Context in Grants and Key Policies

In addition to using the encryption context to check ciphertext integrity and authenticity, AWS KMS supports authorization by using grants and key policies that incorporate the encryption context. Authorization that uses an encryption context more tightly controls access to encrypted resources. When you create a grant, for example, you can optionally specify an encryption context that unambiguously identifies the resource to which long term access is being granted. As an example, consider Amazon EBS. When an Amazon EBS volume is attached to an Amazon EC2 instance, a grant is created that allows only that instance to decrypt only that volume. This is accomplished by encoding the volume ID in the encryption context passed to the **CreateGrant** function. Without the encryption context, the Amazon EC2 instance would require access to all volumes encrypted under the key before it could decrypt a specific volume.

Logging the Encryption Context

AWS KMS uses AWS CloudTrail to log the encryption context, thereby enabling you to determine which keys and data have been accessed. That is, the log entry enables you to determine exactly which key was used to encrypt or decrypt a specific data item referenced by the encryption context.

> **Important**
> Because the encryption context value is logged, it must not contain sensitive information.

Storing the Encryption Context

You should store the encryption context alongside the encrypted data to simplify using it when you call the `Decrypt` API. One security enhancement you might consider is to store only enough of the encryption context that you can create the full context on the fly when needed for encryption or decryption. For example, if you are encrypting a file and decide that the encryption context should be the full file path, store only part of that path alongside of the encrypted file contents. Then, when you need the full encryption context, reconstruct it from the stored fragment. If someone then moves the file to a different location, when you recreate the encryption context, the context will be different and the decryption process will fail, thereby informing you that your data has been tampered with.

Reference: AWS KMS and Cryptography Terminology

This section provides a brief glossary of terms for working with encryption in AWS KMS.

- **Additional authenticated data (AAD):** Offers both data-integrity and authenticity by using additional authenticated data during the encryption process. The AAD is authenticated but not encrypted. Using AAD with authenticated encryption enables the decryption process to detect any changes that may have been made to either the ciphertext or the additional authenticated data after encryption.
- **Authentication:** The process of determining whether an entity is who it claims to be, or that information has not been manipulated by unauthorized entities.
- **Authorization:** Specifies an entity's legitimate access to a resource.
- **Block cipher modes:** Encrypts plaintext to ciphertext where the plaintext and cipher text are of arbitrary length. Modes are typically used to encrypt something that is longer than one block.
- **Block ciphers:** An algorithm that operates on blocks of data, one block at a time.
- **Data key:** A symmetric key generated by AWS KMS for your service. Inside of your service or custom application, the data key is used to encrypt or decrypt data. It can be considered a resource by a service or application, or it can simply be metadata associated with the encrypted data.
- **Decryption:** The process of turning ciphertext back into the form it had before encryption. A decrypted message is called plaintext.
- **Encryption:** The process of providing data confidentiality to a plaintext message. An encrypted message is called ciphertext.
- **Encryption context:** AWS KMS specific AAD in the form of a "key":"value" pair. Although not encrypted, it is bound to the ciphertext during encryption and must be passed again during decryption. If the encryption context passed for encryption is not the same as the encryption context passed for decryption or the ciphertext has been changed, the decryption process will fail.
- **Master key:** A key created by AWS KMS that can only be used within the AWS KMS service. The master key is commonly used to encrypt data keys so that the encrypted key can be securely stored by your service. However, AWS KMS master keys can also be used to encrypt or decrypt arbitrary chunks of data that are no greater than 4 KB. Master keys are categorized as either customer managed keys or AWS managed keys. Customer managed keys are created by a customer for use by a service or application. AWS managed keys are the default keys used by AWS services that support encryption.
- **Symmetric key cryptography:** Uses a single secret key to encrypt and decrypt a message.

Document History

The following table describes the important changes to the documentation since the last release of AWS Key Management Service.

- **Current API version**: 2014-11-01
- **Latest documentation update**: August 31, 2015

Change	Description	Release Date
Update	Updated the Limits (p. 83) page to explain the new requests per second limits.	August 31, 2015
New content	Added information about the charges for using AWS KMS. See Pricing (p. 5).	August 14, 2015
New content	Added requests per second to the Limits (p. 83) topic.	June 11, 2015
New content	Added a new Java code sample demonstrating use of the UpdateAlias API. See Updating an Alias (p. 76).	June 1, 2015
Update	Moved the AWS Key Management Service regions table to the *AWS General Reference*.	May 29, 2015
Added service support	Added support for Amazon EMR. See How Amazon EMR Uses AWS KMS (p. 37).	January 28, 2015
Added service support	Added support for Amazon WorkMail. See How Amazon WorkMail Uses AWS KMS (p. 34).	January 28, 2015
Added service support	Added support for Amazon RDS. See How Amazon Relational Database Service Uses AWS KMS (p. 29).	January 6, 2015
Added service support	Added support for Elastic Transcoder. See How Elastic Transcoder Uses AWS KMS (p. 30).	November 24, 2014
New Guide	Introduced AWS Key Management Service.	November 12, 2014

Limits

All AWS KMS objects have limits that apply to each region and each AWS account. If you need to exceed these limits, please visit the AWS Support Center and create a case.

Keys: 100
: You can have up to 100 keys per region. All keys count towards this limit regardless of their status (enabled or disabled). You can request more keys in a region; however, managing more than 100 keys from the AWS Management Console may be slower than acceptable. If you have more than 100 keys in a region, we recommend managing them programmatically with the AWS SDKs or AWS Command Line Tools.

Aliases: 200
: An alias is an independent display name that you can map to a key. It is not a property of a key. You can map multiple aliases to a single key, so the limit for aliases is higher than the limit for keys. If you request an increase in the number of keys, you may also need to request an increase in the number of aliases.

  For more information about aliases, see Working with Aliases (p. 74).

Grants per key: 250
: Grants are advanced mechanisms for specifying permissions that you or an AWS service integrated with AWS KMS can use to limit how and when a key can be used. Grants are attached to a key, and each grant contains the principal who receives permission to use the key, the ID of the key, and a list of operations that can be performed. Grants are an alternative to the key policy.

  Each key can have up to 250 grants, including the grants created by AWS services that are integrated with AWS KMS. For a list of these services, see AWS Services Integrated with AWS KMS (p. 4). One effect of this limit is that you cannot create more than 250 resources that use the same key. For example, you cannot create more than 250 encrypted Amazon Elastic Block Store (Amazon EBS) volumes that use the same customer master key.

  For more information about grants, see Grants (p. 21).

Grants for a given principal per key: 30
: For a given customer master key (CMK), no more than 30 grants may specify the same grantee principal. For example, assume that you want to encrypt multiple Amazon EBS volumes and attach them to a single Amazon Elastic Compute Cloud (Amazon EC2) instance. In this case, a unique grant is created for each encrypted EBS volume, but the EC2 instance is the grantee principal in all of these grants. Each grant gives permission to the EC2 instance to decrypt the EBS volume key using a specific CMK. For each CMK, you can have up to 30 grants that name the same EC2 instance as the grantee principal. This effectively means that there can be no more than 30 encrypted EBS volumes per EC2 instance for a given CMK.

Requests per second: varies

The point at which AWS KMS begins to throttle API requests differs depending on the API operation. The following table lists each API operation and the point at which AWS KMS begins to throttle API requests for that operation.

The API operations in the first row share a limit of 100 requests per second. The remaining AWS KMS APIs each have a unique limit for requests per second, which means the limit is not shared.

For example, when you make 50 GenerateDataKey requests and 30 Decrypt requests per second, you can make an additional 20 requests per second using any of the APIs in the first row of the following table before AWS KMS begins to throttle your requests. When you make 70 Encrypt requests and 40 GenerateRandom requests per second, AWS KMS will throttle your requests because you are making more than 100 requests per second using APIs in the first row of the following table.

API operation	Requests-per-second limit
Encrypt Decrypt ReEncrypt GenerateRandom GenerateDataKey GenerateDataKeyWithoutPlaintext	100 (shared)
CreateAlias	5
CreateGrant	15
CreateKey	5
DeleteAlias	5
DescribeKey	30
DisableKey	5
DisableKeyRotation	5
EnableKey	5
EnableKeyRotation	5
GetKeyPolicy	30
GetKeyRotationStatus	5
ListAliases	5
ListGrants	5
ListKeyPolicies	5
ListKeys	5
PutKeyPolicy	5
RetireGrant	15
RevokeGrant	15
UpdateAlias	5

API operation	Requests-per-second limit
UpdateKeyDescription	5

Note that API requests may be made directly or by an integrated AWS service on your behalf. For example, you may use server-side encryption with AWS KMS (SSE-KMS) to upload data to Amazon Simple Storage Service (Amazon S3). When you do, each PUT request to Amazon S3 results in a GenerateDataKey API request from Amazon S3 to AWS KMS on your behalf to create an object key that is used to encrypt the Amazon S3 object. In this case, each API request that Amazon S3 makes on your behalf counts towards your limit, and your requests will be throttled if you exceed 100 Amazon S3 PUT requests per second.