Amazon Web Services KMS External Key Store (XKS)

nShield® HSM Integration Guide

2026-08-24

Table of Contents

Chapter 1. Introduction

This document describes how to deploy an AWS Key Management Service (KMS) External Key Store (XKS) supported by an Entrust nShield Hardware Security Module (HSM).

1.1. Overview

AWS KMS External Key Store helps organizations meet regulatory requirements for storing encryption keys outside the AWS Cloud. With this feature, AWS KMS customer managed keys can be protected by your own Entrust nShield HSM or by nShield as a service (nSaaS) within AWS data centers.

To enable AWS KMS External Key Store, the AWS KMS key hierarchy is extended with an external root of trust. The root keys are generated and stored within your nShield HSM. When encryption or decryption of a data key is required, AWS KMS forwards the request to your nShield HSM through an external key store proxy (XKS proxy) that you manage.

The XKS proxy plays a critical role in mediating all interactions between AWS KMS and your Entrust nShield HSM. It translates AWS KMS requests into a format understood by your Entrust nShield HSM, enabling seamless communication between the two systems.

1.2. Product configuration

Entrust tested the integration with the following versions:

Product	Version
Operating System	AWS Linux

An Amazon Linux EC2 instance was used for this deployment. However, the choice of Linux distribution may vary depending on your organization's preferences, standards, and toolset. Adapt the deployment process as necessary to meet your specific requirements and infrastructure.

1.3. Supported nShield hardware and software versions

Entrust has successfully tested the integration with the following nShield hardware and software versions:

HSM	Security World Software	Firmware	Netimage
Connect 5c	13.9.5	13.8.4	13.9.5
nShield XC	13.9.5	13.8.3	13.9.5

1.4. Supported nShield features

Entrust has successfully tested nShield HSM integration with the following features:

Feature	Supported
Module Only Key	Yes
nSaaS	Supported but not tested

1.5. Requirements

You must have:

- Access to the Entrust TrustedCare Portal. To request an account, contact nshield.support@entrust.com.
- Access to the AWS KMS XKS Proxy GitHub.

This integration uses public endpoint connectivity for AWS XKS. This means that:

- The external key store proxy must be reachable through a publicly routable endpoint.
- You must obtain a TLS certificate issued by an AWS KMS XKS Proxy API specification trusted certificate authority.
- The Subject Common Name (CN) on the TLS certificate must match the domain name specified in the proxy URI endpoint for the external key store proxy.
  For example, if the public endpoint is `https://myproxy.xks.example.com`, the CN on the TLS certificate must be `myproxy.xks.example.com` or `*.xks.example.com`.
- The AWS security group must allow custom TCP traffic between the AWS EC2 instance and the remote nShield HSM or nSaaS.

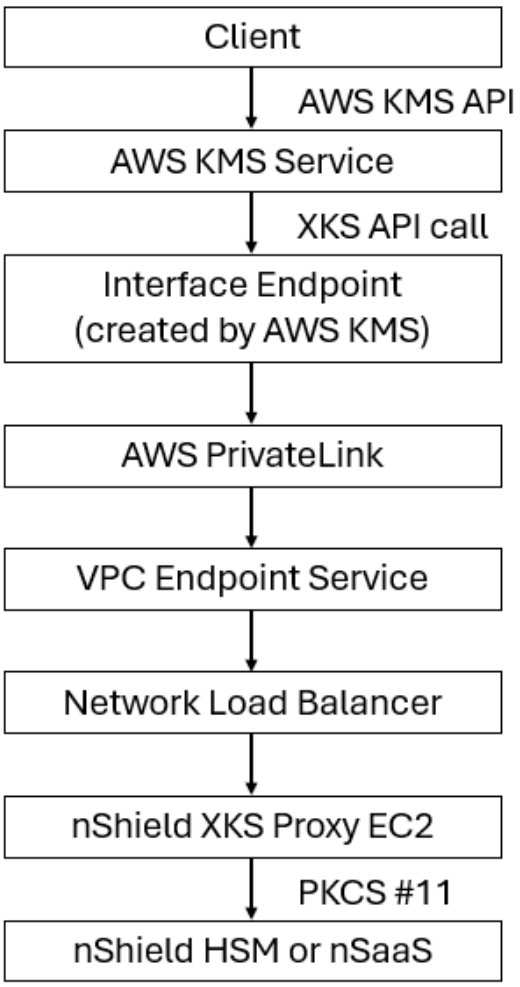
You should also familiarize yourself with:

- The AWS Key Management Service External Key Store Documentation.
- The nShield Documentation.

- Your organization's certificate policy, certificate practice statement (CPS), and any security policies or procedures governing the administration of the PKI and HSM environment.
- The number of Administrator Cards in the Administrator Card Set (ACS), the quorum required for administrative operations, and the policies governing the management of these cards.
- The Security World compliance level. For more information, see FIPS 140 Level 3 compliance.
- Key attributes and requirements, such as key size, timeout settings, audit requirements, and key usage policies.
- Whether the Security World should be instantiated as recoverable or non-recoverable.

Chapter 2. Architecture overview

The following diagram illustrates the architecture between the AWS KMS API interface to the nShield HSM.



- nShield HSM or nSaaS
- nShield XKS proxy EC2 instance
- Network load balancer
- VPC endpoint service
- Key Administrators: AWS IAM user
- Security rule for the health check and XKS traffic

2.1. nShield HSM or nSaaS

The AWS security group must allow **Custom TCP** inbound and outbound traffic to and from

the nShield HSM or nSaaS IP address. Multiple nShield HSMs can be deployed to provide high availability (HA) and increased throughput.

2.2. nShield XKS proxy EC2 instance

The minimum supported EC2 instance type is t3.large. More powerful EC2 instance types may also be used based on performance and capacity requirements.

2.3. Network load balancer

The target group associated with the Network Load Balancer (NLB) must have the following characteristics:

Parameter	Value
Target type	Instances
Protocol	TCP, port 443
IP address type	IPv4
Health check protocol	HTTP
Health check path	/ping
Advanced health check settings port	Override, port 80
Register targets	nShield XKS proxy EC2

The NLB must have the following characteristics:

Parameter	Value
Type	Network
Scheme	Internet-facing
IP address type	IPv4
Availability Zones	Minimum of two zones
Listeners	TCP, port 443
Cross-zone load balancing	On

2.4. VPC endpoint service

The VPC endpoint service must have the following characteristics:

Parameter	Value
Load balancer type	Network
Supported regions	Minimum of two (2)
Acceptance required	Unchecked
Supported IP address types	IPv4
Allow principals	Your primary zone
Private DNS name	Common name (CN) of nShield XKS proxy EC2

2.5. Key Administrators: AWS IAM user

To enable the integration, you must designate an IAM user as the key administrator. This user will be responsible for managing the AWS KMS key and performing key administration tasks. The user may also be granted permissions to use the KMS key for cryptographic operations, as required by your organization's security policies.

You can either select an existing IAM user or create a new one. When creating an access key for the IAM user, select **Application running outside AWS** as the use case to align with AWS access key best practices.

2.6. Security rule for the health check and XKS traffic

Add the following rules to the security group's inbound and outbound rule sets to allow health check and XKS traffic.

Health check

Parameter	Value
Type	HTTP
Protocol	TCP
Port	80
Source	IPv4 CIDR of your VPC, for example, 10.0.0.0/16

Parameter	Value
Description	NLB health check

XKS traffic

Parameter	Value
Type	HTTPS
Protocol	TCP
Port	443
Source	IPv4 CIDR of your VPC, for example, 10.0.0.0/16
Description	NLB XKS traffic

Chapter 3. Obtain a publicly trusted TLS certificate

AWS requires a domain name and a publicly trusted TLS certificate for the key store. The TLS certificate must be issued by a public certificate authority supported for external key stores. For a list of certificate authorities, see https://github.com/aws/awskms-xksproxy-api-spec/blob/main/TrustedCertificateAuthorities.

- Public domain name
- Publicly trusted TLS certificate
- DNS TXT record for AWS PrivateLink domain verification
- Verify the endpoint service domain

3.1. Public domain name

Because the nShield XKS proxy EC2 is fronted by an NLB, the hostname should resolve to the NLB rather than directly to the EC2 instance.

Create a DNS record for the CN pointing to the NLB DNS name. This is the same CN specified in VPC Endpoint Service.

For example:

```
nshield-xks.entrust.com -> nshield-xks-99c54d711bcb6315.elb.us-east-1.amazonaws.com
```

3.2. Publicly trusted TLS certificate

1. On the EC2 instance, generate and save a 4096-bit RSA private key.

   For example:

   ```
   $ openssl genrsa -out nshield-xks-private.key 4096

   ls -al nshield-xks-private.key
   -rw-------. 1 ec2-user ec2-user 3268 Jul  9 13:31 nshield-xks-private.key
   ```

2. Create a certificate signing request (CSR) using the key.

   Use the same CN specified in VPC Endpoint Service.

   For example:

```
$ openssl req -new -key nshield-xks-private.key -out nshield-xks-private.pem
...
```

You can view the CSR with the following command.

```
$ openssl req -text -noout -verify -in nshield-xks-private.pem
...
```

3. Submit the CSR above to a trusted public certificate authority and obtain a signed certificate.

3.3. DNS TXT record for AWS PrivateLink domain verification

Create a public DNS record for AWS PrivateLink domain verification as follows:

Parameter	Value
Record name	Service endpoint Domain verification name
Type	Service endpoint Domain verification type
Value	Service endpoint Domain verification value

3.4. Verify the endpoint service domain

After the records have been created and resolve successfully (you can test this using `nslookup`) and the TLS certificate has been issued, you should verify the endpoint service domain.

1. In your endpoint, select **Actions** > **Verify domain ownership for private DNS name**.
2. When prompted to confirm the action, enter **verify**.

   The status should be **Verified**.

Chapter 4. Install and configure the Entrust nShield HSM

- Install the nShield Security World Software
- Install the Entrust nShield HSM
- Configure the AWS security group and your corporate firewall
- Enroll the Entrust nShield HSM
- Create a security world

4.1. Install the nShield Security World Software

1. Copy the Security World software to your EC2 instance.
2. Install the Security World software as described in the nShield Security World Software v13.9.5 Installation Guide.
3. Add the Security World utilities path to the system path by creating the `/etc/profile.d/nfast.sh` file.
   This path is typically `/opt/nfast/bin`.

```
$ cat /etc/profile.d/nfast.sh
# Entrust nShield HSM
export PATH=$PATH:/opt/nfast/bin
```

4. Add `ec2-user` to the nfast group:

```
$ sudo gpasswd -a ec2-user nfast
Adding user ec2-user to group nfast
```

5. Run the `enquiry` utility to confirm the Security World is **operational**:

```
$ /opt/nfast/bin/enquiry
Server:
 enquiry reply flags  none
 enquiry reply level  Six
 serial number        7852-268D-3BF9 5F08-02E0-D947
 mode                 operational
 version              13.9.5
...
```

4.2. Install the Entrust nShield HSM

Skip this section if you are using nSaaS.

Install the nShield HSM. You can do this locally, remotely, or remotely via the serial console. Condensed instructions are available in Entrust TrustedCare Portal:

- How To: Locally Set up a new or replacement nShield Connect.
- How To: Set up new or replacement nShield 5s.
- How To: Remotely Setup a new or replacement nShield Connect.
- How To: Remotely Setup a new or replacement nShield Connect XC Serial Console Model.

For more detailed instructions, see the nShield v13.9.5 Hardware Install and Setup Guides.

4.3. Configure the AWS security group and your corporate firewall

1. Add inbound and outbound **Custom TCP** security group rules to allow communication with the HSM on port 9004.

   When using multiple remote HSMs, it is possible to configure all HSMs behind a single public IP address, with each HSM listening on a unique port.

   For example:

AWS Security Group					Remote HSM or nSaaS		
HSM	Type	Port	IP	Internet	Public IP	Mapping	Internal IP
1	Custom TPC	9004	<hsm-public-ip>	<--->	<hsm-public-ip>:9004	<--->	<hsm-ip-1>:9004
2	Custom TPC	9014	<hsm-public-ip>		<hsm-public-ip>:9014	<--->	<hsm-ip-2>:9004

2. If you are using your own nShield HSM, configure a corresponding firewall exception on your corporate firewall as described in the previous step.
3. If you use Remote Administration, repeat the previous steps for port 9005, which is required by the Entrust nShield Trusted Verification Device (TVD). Ensure you include the system hosting the TVD in the corresponding firewall and security group rules.

4.4. Enroll the Entrust nShield HSM

1. Inform the HSM of the client's location.

   In this integration, the client is the AWS EC2 instance. For instructions, see Configuring the nShield HSM to use the client.

   If the deployment is configured for high availability (HA), repeat the client configuration steps for each HSM.

2. Enroll the AWS EC2 instance as a client of the HSM.

   For instructions, see Configuring client computers to use the nShield HSM.

   If the deployment is configured for high availability (HA), repeat the client configuration steps for each HSM.

3. Run the `enquiry` utility to confirm that the HSM is `operational`:

```
$ /opt/nfast/bin/enquiry
Server:
 enquiry reply flags  none
 enquiry reply level  Six
 serial number        7852-268D-3BF9 5F08-02E0-D947
 mode                 operational
 version              13.9.5
 ...
Module #1:
 enquiry reply flags  UnprivOnly
 enquiry reply level  Six
 serial number        7852-268D-3BF9
 mode                 operational
 version              13.8.4
 ...
 module type          nShield 5c
 ...
Module #2:
 enquiry reply flags  UnprivOnly
 enquiry reply level  Six
 serial number        5F08-02E0-D947
 mode                 operational
 version              13.8.3
 ...
 module type          nShield Connect XC
 ...
```

4.5. Create a security world

1. Create a Security World if one does not already exist, or import an existing Security World.

   Follow your organization's security policies and procedures when creating a Security World. For more information, see the Entrust documentation Create a new Security World.

   ACS cards cannot be duplicated after the Security World has been created. Consider creating additional ACS cards during setup to protect against card loss, damage, or failure.

2. Confirm the Security World is `Usable` with `nfkminfo`:

```
$ /opt/nfast/bin/nfkminfo
```

```
World
 generation  2
 state       0x37270008 Initialised Usable ...
 ...
Module #1
 generation 2
 state      0x2 Usable
 ...
Module #2
 generation 2
 state      0x2 Usable
 ...
```

3. Based on your configuration, create the `/opt/nfast/cknfastrc` file containing the nShield PKCS #11 library environment variables.

   For example:

```
# Enable Module protection
CKNFAST_FAKE_ACCELERATOR_LOGIN=1

# PKCS #11 log level and file location
CKNFAST_DEBUG=3
CKNFAST_DEBUGFILE=/opt/nfast/log/pkcs11.log
```

4. Change ownership of the `/opt/nfast/cknfastrc` file to `nfast`:

```
$ ls -al /opt/nfast/cknfastrc
-rw-rw-rw-. 1 root root 324 Apr  3 16:12 /opt/nfast/cknfastrc

$ sudo chown nfast:nfast /opt/nfast/cknfastrc

$ ls -al /opt/nfast/cknfastrc
-rw-rw-rw-. 1 nfast nfast 324 Apr  3 16:12 /opt/nfast/cknfastrc
```

5. Make the `/opt/nfast/log/pkcs11.log` file writable:

```
$ sudo touch /opt/nfast/log/pkcs11.log

$ sudo chmod 664 /opt/nfast/log/pkcs11.log
```

6. Restart the nShield service:

```
$ sudo /opt/nfast/sbin/init.d-ncipher restart
...
```

Chapter 5. Generate a PKCS #11 key in the nShield HSM

According to the AWS documentation, you must create keys in your HSM and map them to the external key store resource in KMS. These keys are used for data encryption with AWS services that support customer keys or within your applications. For more information, refer to https://aws.amazon.com/blogs/aws/announcing-aws-kms-external-key-store-xks/.

The key must be a 256-bit AES key that is enabled and capable of performing encryption and decryption. For more information, refer to https://docs.aws.amazon.com/kms/latest/developerguide/keystore-external.html.

1. On the AWS EC2 instance, run the following command. Note the key type and size used.

   For example:

```
$ /opt/nfast/bin/generatekey pkcs11
module: Module to use? (1, 2) [1] > 1
protect: Protected by? (token, module) [token] > module
type: Key type? (DES3, DH, DHEx, DSA, HMACSHA1, HMACSHA256, HMACSHA384,
                 HMACSHA512, RSA, DES2, AES, Rijndael, ECDSA, ECDH, Ed25519,
                 Ed448, X25519, MLDSA, MLKEM, SLHDSA) [RSA] > AES
size: Key size? (bits, 128-256) [] > 256
plainname: Key name? [] > nshield-xks-hsm-key
nvram: Blob in NVRAM (needs ACS)? (yes/no) [no] > no
key generation parameters:
 operation    Operation to perform       generate
 application  Application                pkcs11
 module       Module to use              1
 protect      Protected by               module
 verify       Verify security of key     yes
 type         Key type                   AES
 size         Key size                   256
 plainname    Key name                   nshield-xks-hsm-key
 nvram        Blob in NVRAM (needs ACS)  no
Key successfully generated.
Path to key: /opt/nfast/kmdata/local/key_pkcs11_uab9d4256896a7dc8fad4966f9081bd873dab12262
```

2. Verify that the key was generated using one of the following methods.

   nfkminfo

```
$ /opt/nfast/bin/nfkminfo -l

Keys with module protection:
 key_pkcs11_uab9d4256896a7dc8fad4966f9081bd873dab12262 'nshield-xks-hsm-key'
```

   rocs

```
$ rocs
`rocs' key recovery tool
```

```
Useful commands: `help', `help intro', `quit'.
rocs> list keys
  No. Name                App    Protected by
    1 nshield-xks-hsm-key pkcs11 module
rocs> quit
```

Chapter 6. Deploy the AWS KMS XKS proxy

6.1. Clone the AWS samples Git repository

The AWS Samples GitHub repository provides essential resources for AWS KMS External Key Store (XKS) integrations, including the XKS proxy API specification. In addition, it contains a reference implementation of an XKS proxy and a test client that can be used to validate compliance with the API specification. For more information, see the AWS Samples repository: https://github.com/aws/aws-kms-xksproxy-api-spec.

1. Install Git on the AWS EC2 instance:

```
$ sudo dnf install -y git
Amazon Linux 2023 repository                                                    69 MB/s |
69 MB     00:00
Amazon Linux 2023 Kernel Livepatch repository                                  427 kB/s |
55 kB     00:00
...
    Complete!
```

2. Create a directory for Git repositories:

```
$ sudo mkdir /opt/git
```

3. Change to the newly created directory and clone the AWS Samples repository:

```
$ cd /opt/git

$ sudo git clone https://github.com/aws-samples/aws-kms-xks-proxy/
Cloning into 'aws-kms-xks-proxy'...
remote: Enumerating objects: 451, done.
remote: Counting objects: 100% (180/180), done.
remote: Compressing objects: 100% (108/108), done.
remote: Total 451 (delta 105), reused 104 (delta 72), pack-reused 271 (from 1)
Receiving objects: 100% (451/451), 248.80 KiB | 16.59 MiB/s, done.
Resolving deltas: 100% (228/228), done.
```

6.2. Configure the XKS proxy

The configuration file is located at `/opt/git/aws-kms-xks-proxy/xks-axum/configuration/settings_nshield.toml`.

1. Edit the `[server]` section as shown below. Set `ip` to the private IP of the AWS EC2 instance. Do not use the public IP. Set `port` to 443. Leave all other settings at their default values.

   For example:

   ```
   [server]
   ip = "10.0.7.200"
   port = 443
   region = "us-east-1"
   service = "kms-xks-proxy"
   ```

2. Edit the `[security]` section as shown below:

   ```
   [security]
   is_sigv4_auth_enabled = true
   is_tls_enabled = true
   is_mtls_enabled = false
   ```

3. Edit the `[external_key_stores]` section as shown below. Specify values for `sigv4_access_key_id` and `sigv4_secret_access_key`. These values are shared secrets used to authenticate requests between AWS KMS and the nShield XKS proxy. They are not AWS IAM access keys and must not be derived from IAM user credentials. Generate these values as shown in the following table.

   For example:

Parameter	Value
uri_path_prefix	"/nshield/xks"
sigv4_access_key_id	Generate yourself or `run tr -dc 'A-Z2-7' </dev/urandom`
sigv4_secret_access_key	Generate yourself or run `openssl rand -base64 32`
xks_key_id_set	"plainname" of the PKCS #11 key generated in Generate a PKCS #11 key in the nShield HSM

   `xks_key_id_set` can contain multiple keys. When specifying multiple keys, separate them with commas.

   ```
   $ tr -dc 'A-Z2-7' </dev/urandom | head -c 24 ; echo
   U4B.....................

   $ openssl rand -base64 32
   Ja4.....................................
   ```

```
[[external_key_stores]]
uri_path_prefix = "/nshield/xks"
sigv4_access_key_id = "U4B....................."
sigv4_secret_access_key = "Ja4......................................."
xks_key_id_set = ["nshield-xks-hsm-key"]
```

4. Delete any other `[external_key_stores]` sections.
5. Copy the key generated in Obtain a publicly trusted TLS certificate to `/opt/git/aws-kms-xks-proxy/xks-axum/tls`.

   For example:

```
$ ls -al /opt/git/aws-kms-xks-proxy/xks-axum/tls/nshield-xks-private.key
-rw-------. 1 root root 3268 Jul  9 15:42 /opt/git/aws-kms-xks-proxy/xks-axum/tls/nshield-xks-private.key
```

6. Copy the signed certificate obtained in Obtain a publicly trusted TLS certificate to `/opt/git/aws-kms-xks-proxy/xks-axum/tls`.

   For example:

```
$ ls -al /opt/git/aws-kms-xks-proxy/xks-axum/tls/nshield-xks_entrust_com.pem
-rwx------. 1 root root 7505 Jul  9 16:28 /opt/git/aws-kms-xks-proxy/xks-axum/tls/nshield-
xks_entrust_com.pem
```

7. Edit the `[tls]` section as shown below. Set `tls_cert_pem` and `tls_key_pem` to the paths of the certificate and private key files copied in the previous steps.

```
[tls]
tls_cert_pem = "tls/nshield-xks_entrust_com.pem"
tls_key_pem = "tls/nshield-xks-private.key"
mtls_client_ca_pem = ""
mtls_client_dns_name = ""
```

8. Leave all other settings at their default values, and then save the file.

6.3. Start the nShield XKS proxy

This section demonstrates one method of building and deploying the XKS proxy using Rust Cargo. Alternative build and deployment methods are available. Select the approach that best aligns with your organization's requirements, operational practices, and toolset.

1. Install Rust and Cargo on the AWS EC2 instance.

```
$ sudo dnf install -y rust cargo
Last metadata expiration check: 3:50:49 ago on Tue Jul  7 14:42:39 2026.
Dependencies resolved.
```

```
...
Complete!
```

2. Build the XKS proxy using Cargo. This process may take several minutes to complete, depending on the EC2 instance type and available system resources.

```
cd /opt/git/aws-kms-xks-proxy/xks-axum

$ sudo cargo build --release
Updating crates.io index
...
   Compiling toml v0.7.8
   Compiling axum-macros v0.3.8
   Compiling xks-proxy v3.1.2 (/opt/git/aws-kms-xks-proxy/xks-axum)
    Finished `release` profile [optimized] target(s) in 4m 29s
```

3. Create the following executable script:

```
$ cat nshield-xks-run
cd /opt/git/aws-kms-xks-proxy/xks-axum
XKS_PROXY_SETTINGS_TOML=configuration/settings_nshield.toml cargo run

$ sudo chmod +x ~/nshield-xks-run
```

4. Run the XKS proxy executable. After it starts successfully, the XKS proxy displays initialization and configuration information in the console, indicating that it is ready to accept requests.

```
$ nohup sudo ./nshield-xks-run > xks.log 2>&1 &
[1] 1619760

$ cat xks.log
    Finished `dev` profile [unoptimized + debuginfo] target(s) in 3m 58s
     Running `target/debug/xks-proxy`
2026-07-09T18:44:33.743964Z  INFO main xks_proxy: 276: Tracing level="DEBUG" is_file_writer_enabled=true
2026-07-09T18:44:33.744161Z  INFO main xks_proxy: 278: Tracing file rotation_kind="HOURLY"
...
```

The nShield XKS proxy is now running and ready to process AWS KMS cryptographic operations.

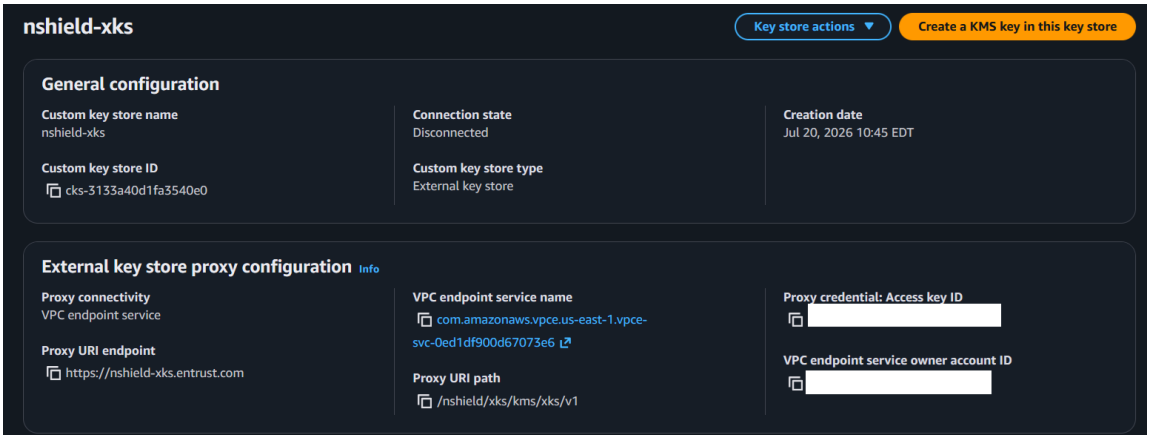
Chapter 7. Create an external key store and keys

7.1. Create an external key store

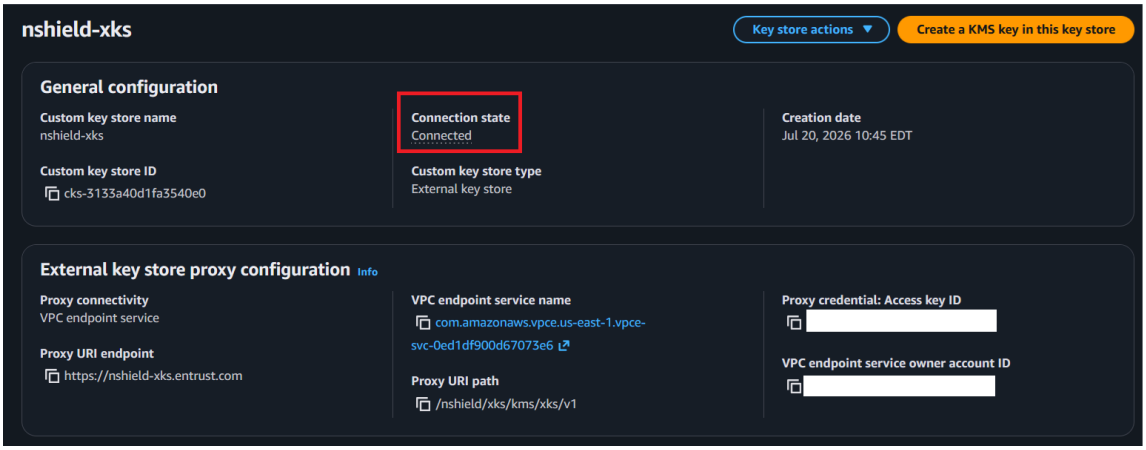
1. Sign in to the AWS management console and navigate to **Key Management Service (KMS)**.
2. In KMS, go to **Custom key stores** > **External key stores**.
3. Select **Create external key store**.
4. Enter the information as shown below, then select **Create external key store**.

Parameter	Value
Proxy connectivity	`VPC endpoint service`
VPC endpoint service name	Name from VPC Endpoint Service
Proxy URL endpoint	URL containing the FQDN, for example, `https://nshield-xks.entrust.com`
Proxy URI path prefix	`/nshield/xks`
Proxy credential: Access key ID	sigv4_access_key_id from Configure XKS PRoxy
Proxy credential: Secret access key	sigv4_secret_access_key from Configure XKS Proxy

The following example shows a newly created external key store. Notice the **Connection state** is **Disconnected**.



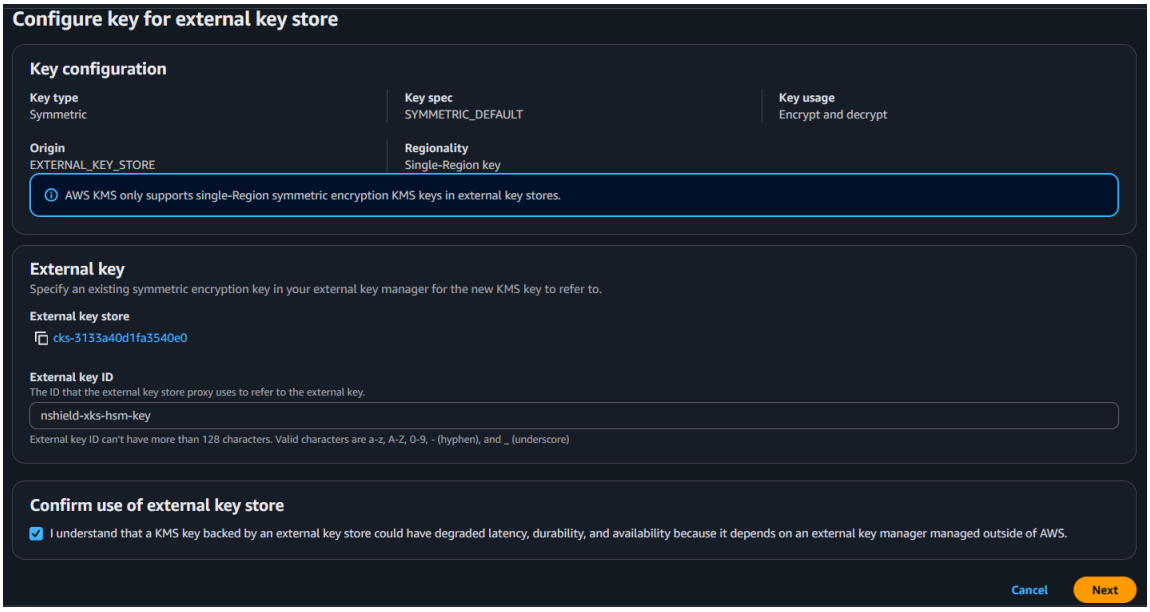
5. Connect the newly created external key store. From the **Key store actions** menu, select **Connect**. After the connection process completes, verify the updated **Connection state**.



After you create the External key store, you can use it to securely manage and store keys in your AWS environment.

7.2. Create a key in an external key store

1. Sign in to the AWS management console and navigate to **Key Management Service (KMS)**.
2. In KMS, select the key store created in section Create an external key store.
3. Select **Create a KMS key in this key store**.
4. In the **Configure key for external key store** window, complete the following fields and then select **Next**:
    - In the **External key ID** field, enter the name of the HSM key generated in section Generate a PKCS #11 key in the nShield HSM.
    - Select the **Confirm use of external key store** checkbox.

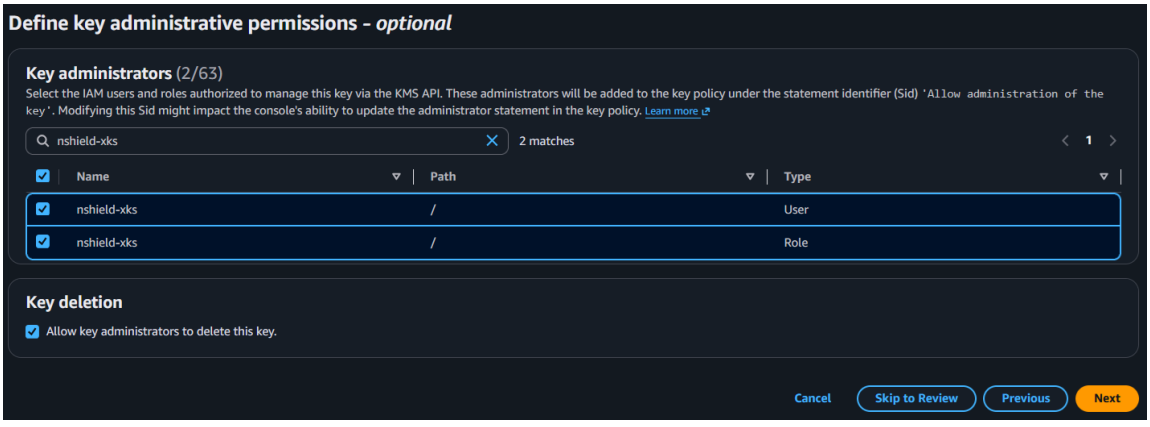


5. In the **Add label** window, enter a name and description for the KMS key. Select **Next** to continue.
6. In the **Define key administrative permissions** window, select the IAM user or users designated as administrators.
   In the **Define key usage permissions** window, select the IAM user or users designated as key users.

   This example uses a single key administrator and a single key user. However, AWS KMS supports assigning multiple IAM users and roles to each permission set.

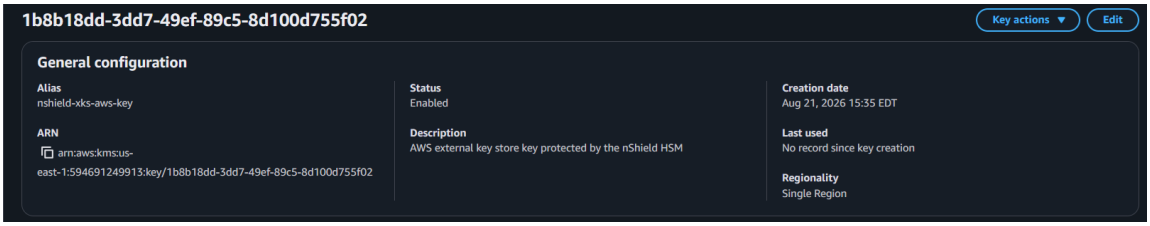
   Select **Next** to continue.

   For example:



7. In the **Edit key policy - optional** window, select **Next**.
8. In the **Review** window, select **Finish**.

   The key is created and the key information is displayed:


1b8b18dd-3dd7-49ef-89c5-8d100d755f02
Key actions
Edit
General configuration
Alias
nshield-xks-aws-key
ARN
arn:aws:kms:us-east-1:594691249913:key/1b8b18dd-3dd7-49ef-89c5-8d100d755f02
Status
Enabled
Description
AWS external key store key protected by the nShield HSM
Creation date
Aug 21, 2026 15:35 EDT
Last used
No record since key creation
Regionality
Single Region

Chapter 8. Connect to the external key store

Clients access AWS KMS keys in an external key store by using IAM user or role credentials (access key ID and secret access key) and specifying the Amazon Resource Name (ARN) of the KMS key.

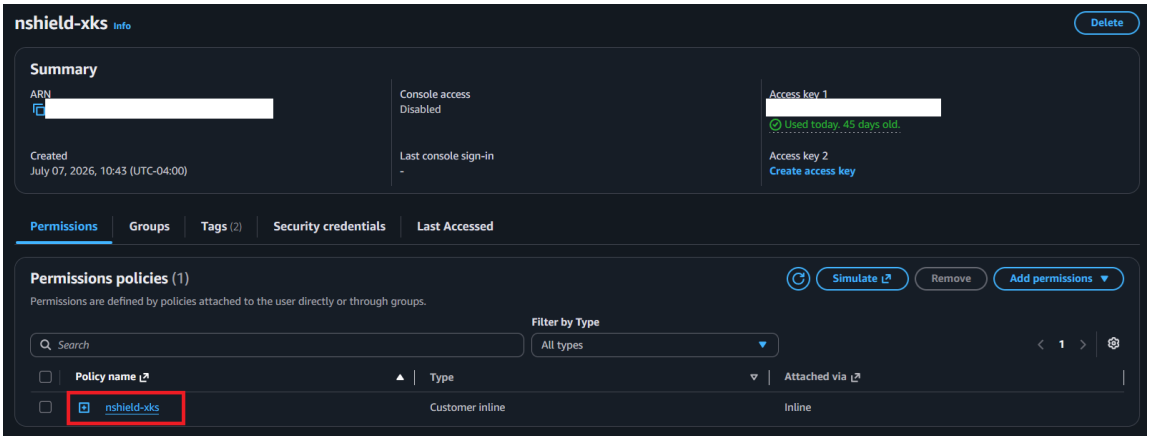
Therefore, the IAM policy associated with the user or role must grant the permissions required to perform cryptographic operations on the KMS key. Additionally, the KMS key policy must allow the user or role to access the key.

1. Sign in to the AWS Management Console and navigate to **Identity and Access Management (IAM)**.
2. Select your IAM user.
3. Select **Add permissions** → **Create inline policy**.
4. Select the **JSON** tab.
5. Copy the policy shown below and paste it into the field on the JSON tab and then select **Next** to continue.
    - Replace the ARN in the `"Resource"` element with the ARN of your AWS KMS key.
    - To grant permissions to multiple keys, add additional key ARNs to the `"Resource"` element, separated by commas.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "UseDelineaKmsKey",
            "Effect": "Allow",
            "Action": [
                "kms:Encrypt",
                "kms:Decrypt",
                "kms:GenerateDataKey",
                "kms:GenerateDataKeyWithoutPlaintext",
                "kms:DescribeKey"
            ],
            "Resource": [
                "arn:aws:kms:us-east-1:594663549913:key/1b8b18dd-3dd7-79ef-87c5-8d169d755f02"
            ]
        }
    ]
}
```

6. Enter a policy name and then select **Create policy** to continue.

   The policy is created and displayed, for example:



This completes the configuration of Amazon Web Services KMS External Key Store (XKS) nShield® HSM integration. Applications can connect to the external key store by using the IAM user credentials and specifying the ARN of the key in the external key store.

Chapter 9. Additional resources and related products

9.1. nShield HSMs

9.2. nShield as a Service

9.3. Entrust products

9.4. nShield product documentation