



Amazon Web Services KMS External Key Store (XKS) and AWS PrivateLink

KeyControl Integration Guide

2026/11/06

© Entrust Corporation. All rights reserved.

Connect AWS KMS External Key Store (XKS) to KeyControl over AWS PrivateLink

This guide sets up AWS KMS to reach your KeyControl XKS proxy **privately** (no internet path from AWS KMS to KeyControl). Follow the steps in order. Region used: `us-east-1` (use your own Region everywhere).

```
AWS KMS --> VPC endpoint service --> Internal NLB --> KeyControl node 1 / node 2
             (PrivateLink)         (TCP 443)        (private IPs only)
```

The whole path stays on private networks. If KeyControl runs in your own data center, the VPC is connected to it with **AWS Direct Connect** (recommended).

Before you start

You need:

- A VPC with **two private subnets in two different Availability Zones**.
- Two KeyControl nodes (a cluster) with **private IP addresses** that the VPC can reach on TCP 443 (see "Recommended network design" below).
- A **public DNS domain you control** (example: `example.com`). You will use the name `xks.example.com`.
- A **publicly trusted TLS certificate** for `xks.example.com` (CN or SAN; a wildcard such as `*.example.com` is fine) from a CA on the AWS trusted CA list, installed on the KeyControl XKS proxy. A private, enterprise or self-signed certificate is **not accepted**, even with PrivateLink (see Step 6).
- The KeyControl nodes' security group (or your data center firewall) allows **TCP 443 from the VPC CIDR**.

Recommended network design

If KeyControl runs in your own data center, connect it to the VPC with **AWS Direct Connect**. A Site-to-Site VPN is an alternative.

- Keep the KeyControl nodes on **private IP addresses only**. Do not give them public IP addresses.
- Keep the load balancer **internal**. Neither KeyControl nor the load balancer should be reachable from the internet.
- Use the same private connection for administrators to open the KeyControl GUI.
- The VPC IP range must **not overlap** with the IP range of your data center.

Names used in this guide (change if you like):

Item	Name
Target group	`keycontrol-xks-443`
Network Load Balancer	`kc-xks-nlb`
Endpoint service	`xks-vpc-endpoint-service`
Private DNS name	`xks.example.com`
KMS external key store	`keycontrol-xks`

Step 1. Create the target group

EC2 > Target Groups > **Create target group**

1. Target type: **IP addresses** and enter the **private IP** of each KeyControl node. This is required when KeyControl is in your data center (over Direct Connect or VPN). Use **Instances** only if the nodes are EC2 instances in this VPC.
2. Name: `keycontrol-xks-443`. Protocol: **TCP**. Port: **443**. IP address type: **IPv4**. Select your VPC.
3. Health checks: protocol **HTTPS**, path `/`. Under Advanced, success codes `200-399`.
4. Register at least **two** KeyControl nodes on port **443** (by private IP, or by instance). If a load balancer in your data center fronts the nodes, you can register its virtual IP instead (see "Using a load balancer in your data center" below). Create the target group.

The "two Availability Zones" in this guide refer to the **load balancer subnets in the VPC** (Step 2). They do not apply to where the KeyControl nodes are; on-premises nodes have no AWS Availability Zone.

Step 2. Create the INTERNAL Network Load Balancer

EC2 > Load Balancers > **Create load balancer** > Network Load Balancer

1. Name: `kc-xks-nlb`.
2. Scheme: **Internal** (never Internet-facing). IP address type: **IPv4**.
3. VPC: your VPC. Mappings: select **both Availability Zones** and a private subnet in each.
4. Security groups: leave empty.
5. Listener: **TCP : 443** > forward to `keycontrol-xks-443`.
6. Create the load balancer. Open it > **Attributes** > Edit > turn on **Cross-zone load balancing** > Save.
7. Wait until the state is **Active** and the targets in `keycontrol-xks-443` show **Healthy**.

If a target is not healthy, fix that first (usually the node security group does not allow TCP 443 from the VPC CIDR).

Step 3. Create the VPC endpoint service

VPC > Endpoint services > **Create endpoint service**

1. Name: `xks-vpc-endpoint-service`.
2. Load balancer type: **Network**. Select `kc-xks-nlb`.
3. **Acceptance required: turn OFF** (leave the box unchecked). AWS KMS cannot connect if acceptance is required.
4. **Enable private DNS name**: on. Private DNS name: `xks.example.com`.
5. Supported IP address types: **IPv4**. Create.

Step 4. Allow AWS KMS to connect

Endpoint service > **Allow principals** tab > **Allow principals**

- Add: `cks.kms.us-east-1.amazonaws.com`

(Replace `us-east-1` with your Region.)

Step 5. Verify domain ownership

1. Open the endpoint service > **Details** and copy the **Domain verification name** and **Domain verification value**.
2. In your **public** DNS, create a **TXT** record: name = the verification name, value = the verification value.
3. Back on the endpoint service: **Actions > Verify domain ownership for private DNS name**.
4. Wait until **Domain verification status** shows **Verified**.

You do not need to create an A record for `xks.example.com`. AWS creates the private DNS entry for KMS automatically.

Step 6. Install the TLS certificate on KeyControl

A public certificate is required. PrivateLink makes the network path private, but AWS KMS still checks the KeyControl certificate in the same way as for a public endpoint:

- It must be issued by a CA on the AWS trusted CA list (publicly trusted CAs).
- Its CN or SAN must match the private DNS name (`xks.example.com`, or a wildcard that covers it).
- A self-signed, private or enterprise CA certificate is rejected, and the key store fails to connect with `XKS_PROXY_INVALID_TLS_CONFIGURATION`.

The certificate does not expose KeyControl to the internet, and `xks.example.com` does not need a public A record. When you request the certificate, use the CA's DNS validation (a TXT record), which most public CAs offer.

Install the same certificate on every KeyControl node (see the KeyControl documentation for certificate installation).

Step 7. Prepare the key set in KeyControl

In the KeyControl **Cloud Keys vault**:

1. Add your AWS **CSP account**.
2. Create a **Key Set** for AWS, tick **Enable External Key Store**.
3. Copy the values KeyControl shows: **Access Key ID**, **Secret Access Key** and **Proxy URI path**. Keep them safe; you need them in the next step.

Step 8. Create the external key store in AWS KMS

AWS KMS > Custom key stores > External key stores > **Create external key store**

Field	Value
Name	`keycontrol-xks`
Proxy connectivity	**VPC endpoint service**
VPC endpoint service name	`com.amazonaws.vpce.us-east-1.vpce-svc-xxxxxxxxxxxxxxxxx` (copy the service name from the endpoint service page)
Proxy URI endpoint	`https://xks.example.com`
Proxy URI path	the path shown in KeyControl (for example `/kms/xks/v1`)
Access key ID / Secret access key	from KeyControl (Step 7)

Click **Create**, then select the key store > **Actions > Connect**. Wait about 2 minutes until the state is **Connected**.

Step 9. Create a key and test it

1. In the KeyControl Cloud Keys vault, create a **CloudKey** in the key set you created (Region `us-east-1`). Make sure its status is **Enabled**.
2. Test from the AWS CLI v2 (replace `<key-id>` with the key ID from KMS):

```
CT=$(aws kms encrypt --region us-east-1 --key-id <key-id> \
  --plaintext "$(echo -n hello | base64)" --query CiphertextBlob --output text)

aws kms decrypt --region us-east-1 --key-id <key-id> --ciphertext-blob "$CT" \
  --query Plaintext --output text | base64 -d
```

The last command prints `hello`. The setup is working.

Using a load balancer in your data center (for example F5)

If KeyControl is on-premises, you can put a load balancer with one virtual IP (VIP) in front of the KeyControl nodes and register **only that VIP** as the IP target in Step 1. This is supported. The NLB accepts a target group with a single target, and AWS's advice to use two or more targets exists so that the proxy endpoint stays available. A VIP that fronts two or more KeyControl nodes provides that availability, if you cover the points below.

Topic	What to do
No single point of failure	The VIP is now the only target AWS sees. Use a high-availability load balancer pair with a floating VIP, and a resilient private link: two Direct Connect connections (ideally in different locations), or Direct Connect plus a VPN backup.
Health	Monitor every KeyControl node on the load balancer (HTTPS, for example `/`). The VIP must stop answering when no node is healthy, so the AWS health check shows the real state.
TLS	Use a **TCP pass-through (layer 4)** virtual server so KeyControl presents its certificate (Step 6). If the load balancer terminates TLS instead, it must present a publicly trusted certificate for `xks.example.com`.
Do not change requests	AWS signs every request (AWS Signature Version 4, which covers the host name and path). Do not rewrite the host header, path, query or body.
Return traffic	Requests arrive from the private IP addresses of the load balancer subnets in your VPC. Allow that range, and make sure replies return through the load balancer (use SNAT on the load balancer, or route replies back to it).
Addressing and routes	The VIP must be a private address (RFC 1918 or 100.64.0.0/10) that is reachable over Direct Connect from the VPC subnets of the NLB. The VPC range must not overlap your data center range. Keep **cross-zone load balancing** on.
Latency	AWS expects the proxy to answer within 250 ms, with a round-trip time of 35 ms or less between the AWS Region and the proxy. Count the load balancer and the Direct Connect path in that budget, and size the setup for up to 1,800 requests per second.

An alternative is to skip the load balancer: register the individual KeyControl node IP addresses as targets in Step 1, and the NLB spreads the requests.

Quick checklist

- [ ] NLB scheme is **Internal**, cross-zone load balancing is **on**, both targets **Healthy**
- [ ] Endpoint service: acceptance **not** required, private DNS name set, domain **Verified**
- [ ] Allow principal `cks.kms.<region>.amazonaws.com` added
- [ ] Publicly trusted certificate (CA on the AWS list) for `xks.example.com` installed on every KeyControl node
- [ ] KMS key store is **Connected**, CloudKey is **Enabled**, encrypt and decrypt succeed

Important: changing or reconnecting the key store

If you **disconnect** the external key store in KMS, KeyControl automatically **disables** its CloudKeys. Connecting again does **not** re-enable them. After any disconnect and reconnect, open the KeyControl Cloud Keys vault, select each CloudKey and choose **Actions > Enable CloudKey**. The key store can show **Connected** while the keys are still disabled, so always run the encrypt test.

AWS also does not let you change the load balancer behind an endpoint service while KMS is connected, so create the internal NLB correctly from the start.

Troubleshooting

Message or symptom	Cause	Fix
Create key store fails: `XksProxyVpcEndpointServiceNotFoundException`	KMS principal not allowed	Add `cks.kms.<region>.amazonaws.com` (Step 4)
Create key store fails: `XksProxyInvalidConfigurationException`	Acceptance required is on	Turn it off (Step 3)
Connect fails: domain not verified	TXT record missing or not propagated	Redo Step 5 until **Verified**
Connect fails: `XKS_PROXY_INVALID_TLS_CONFIGURATION`	Certificate name does not match `xks.example.com`, or the certificate is not from a CA on the AWS trusted list (for example self-signed or an enterprise CA)	Install a publicly trusted certificate for the exact private DNS name (Step 6)
Targets **Unhealthy**	Node security group or firewall blocks the NLB, or no route to the nodes	Allow TCP 443 from the VPC CIDR to the KeyControl nodes; check the Direct Connect or VPN routes
Encrypt: "cannot communicate with the external key store proxy"	NLB targets down or cross-zone off	Check targets are Healthy; turn on cross-zone load balancing
Encrypt: "external key ... is not enabled in the external key manager"	CloudKey is disabled in KeyControl	**Actions > Enable CloudKey** in KeyControl

Next: make KeyControl's own calls to AWS private

This guide makes **AWS KMS reach KeyControl** privately. KeyControl also calls the AWS APIs (KMS, IAM, EC2 and Systems Manager) when it manages the AWS account and its keys. To keep those calls off the internet too, follow the separate guide **Private connectivity from KeyControl to AWS (BYOK and HYOK)**.