



**ENTRUST**

# Entrust KeySafe 5

**KeyControl Compliance Manager Integration  
Guide**

2024-12-05

# Table of Contents

1. Introduction .....	1
1.1. Product configuration .....	1
2. Install and configure KCM and KeySafe 5.....	3
2.1. Install the KeyControl Compliance Manager.....	3
2.2. Install KeySafe 5.....	3
3. Integrate Entrust KeySafe 5 with the KeyControl Compliance Manager server .....	13
3.1. Prerequisites.....	13
3.2. Create the KCM tenant .....	13
3.3. Create the KeySafe 5 app link token key.....	15
3.4. Capture the KCM TLS Certificate chain and load in the KeySafe 5 server.....	16
3.5. Enable the connection between the KeySafe 5 and KCM .....	21
3.6. Validate the integration .....	28
4. Additional resources and related products.....	30
4.1. nShield as a Service.....	30
4.2. KeyControl .....	30
4.3. KeyControl as a Service.....	30
4.4. Entrust products.....	30
4.5. nShield product documentation .....	30

---

# Chapter 1. Introduction

This guide describes:

- The procedure to install and configure Entrust KeySafe 5.
- The procedure to install and configure Entrust KeyControl Compliance Manager.
- The procedure to integrate Entrust KeySafe 5 and Entrust KeyControl Compliance Manager.

When all of these procedures are performed, the combined solution facilitates regulatory compliance with specific policies and documentation templates on security objects protected in KeySafe 5.

## 1.1. Product configuration

Entrust has successfully tested Entrust KeySafe 5 with Entrust KeyControl Compliance Manager in the following configurations:

<b>Product</b>	<b>Version</b>
KeySafe 5	1.4.0
KeyControl Compliance Manager	10.3.1
Operating System	RedHat 9
Security World	13.6.3
nShield HSM hardware	Connect XC, nShield 5C
FIPS 140 Level 3	Yes

### 1.1.1. Supported nShield hardware and software versions

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

HSM	Security World Software	Firmware	Image	FIPS 140 Level 3
nShield 5c	13.6.3	13.4.5 (FIPS 140-2 Certified)	13.6.5	Yes
Connect XC	13.6.3	12.72.3 (FIPS 140-2 certified)	13.6.5	Yes

---

# Chapter 2. Install and configure KCM and KeySafe 5

## 2.1. Install the KeyControl Compliance Manager

The Entrust KeyControl Compliance Manager server is a software solution deployed from an OVA or ISO image. Entrust recommends that you read the Entrust KeyControl Compliance Manager [Installation and Upgrade Guide](#) online documentation to fully understand the KeyControl Compliance Manager server deployment.

To configure a KeyControl Compliance Manager cluster (active-active configuration is recommended), Entrust recommends the use of the OVA installation method, as described in the Entrust [KeyControl Compliance Manager OVA Installation](#) online documentation.



Although an active-active cluster is not a requirement, and a single KeyControl Compliance Manager node can be deployed to perform its functions, Entrust strongly recommends deploying the solution with a minimum of two nodes in an active-active cluster solution.

## 2.2. Install KeySafe 5

Please refer to the Online documentation for installation instructions:

- [KeySafe 5 v1.4 Quick Start Guide](#).
- [KeySafe 5 v1.4 Installation and Upgrade Guide](#).

### 2.2.1. Server Installation - A quick example

1. Update to the latest packages:

```
% sudo dnf update
```

2. Install Docker:

```
% sudo dnf install docker
```

### 3. Install Helm:

```
% sudo curl -L https://mirror.openshift.com/pub/openshift-v4/clients/helm/latest/helm-linux-amd64 -o /usr/local/bin/helm
% sudo chmod +x /usr/local/bin/helm
```

### 4. Open up the firewall:

```
% sudo firewall-cmd --zone=public --permanent --add-port="443/tcp" --add-port="5671/tcp"
% sudo firewall-cmd --reload
% sudo firewall-cmd --list-all
```

### 5. Transfer the `nshield-keySAFE5-1.4.0.tar.gz` to the server.

### 6. Create a directory for the installer file:

```
% mkdir keysafe5-install
```

### 7. Untar the KeySafe 5 `tar` file into this directory:

```
% tar -xf nshield-keySAFE5-1.4.0.tar.gz -C keysafe5-install
```

### 8. Change to the install directory:

```
% cd keysafe5-install
```

### 9. Disable authentication:

```
export DISABLE_AUTHENTICATION=yes
```



Refer to documentation for information on how to setup authentication.

### 10. Perform a dry-run of the installation:

```
% ./deploy.sh -n
```

#### Output:

```
helm not found. Will fetch helm
istioctl not found. Will fetch istioctl
kubectl not found. Will fetch kubectl
Pre-flight checks
Found firewalld:
  * configuration will need sudo as K3s firewall rules are applied.
  * To prevent firewall reconfiguration set FIREWALL_CONFIGURED=YES
```

```
Kubernetes installation not found. Will fetch and install K3s
  This will call sudo
istioctl unavailable – cannot determine the state of Istio
No MongoDB specified. Will install MongoDB
No RabbitMQ specified. Will install RabbitMQ
Will create local object storage in K3s
Will install a local Docker Registry
No authentication set
```



Resolve any issues before proceeding with the actual installation.

## 11. If no errors are reported, run the installation:

```
% ./deploy.sh -y
```

### Output:

```
Preparing to deploy KeySafe 5
helm not found. Will fetch helm
istioctl not found. Will fetch istioctl
kubectl not found. Will fetch kubectl
=====
===== Downloading Required Tools =====
=====
kubectl Acquisition          In Progress
kubectl Acquisition          Completed
Helm Acquisition             In Progress
Helm Acquisition             Completed
Istioctl Acquisition         In Progress
Istioctl Acquisition         Completed
Pre-flight checks
Found firewall:
* configuration will need sudo as K3s firewall rules are applied.
* To prevent firewall reconfiguration set FIREWALL_CONFIGURED=YES
Kubernetes installation not found. Will fetch and install K3s
  This will call sudo
Istio not installed. Will install Istio
No MongoDB specified. Will install MongoDB
No RabbitMQ specified. Will install RabbitMQ
Will create local object storage in K3s
Will install a local Docker Registry
No authentication set
=====
===== Deploying =====
=====
Apply K3s Firewall rules     In Progress
Apply K3s Firewall rules     Completed
K3s Installation              In Progress
K3s Installation              Completed
Local Docker Registry         In Progress
Local Docker Registry         Completed
Docker Images                 In Progress
Docker Images                 Completed
Istio Installation            In Progress
Istio Installation            Completed
CA Setup                       In Progress
CA Setup                       Completed
RabbitMQ Installation         In Progress
RabbitMQ Installation         Completed
```

```
MongoDB Installation      In Progress
MongoDB Installation      Completed
Object Storage Local PVC Installation In Progress
Object Storage Local PVC Installation Completed
Installing KeySafe 5      In Progress
Installing KeySafe 5      Completed
Updating RabbitMQ certificates In Progress
Updating RabbitMQ certificates Completed
Uninstall local Docker Registry In Progress
Uninstall local Docker Registry Completed
```

To use K3s you will need to set the environment variable KUBECONFIG  
This may be done by running:

```
export KUBECONFIG=/home/<user>/.kube/config
```

You may append this to your shell's configuration file:

```
KUBECONFIG=/home/<user>/.kube/config
export KUBECONFIG
```

You can list the pools through <https://<keysafe5-server-ip>/mgmt/v1/pools>

```
{
  "data": {
    "pools": []
  },
  "meta": {
    "page": {
      "next": "",
      "previous": ""
    }
  }
}
```

The ui is now available through <https://<keysafe5-server-ip>>

Deployment completed successfully.

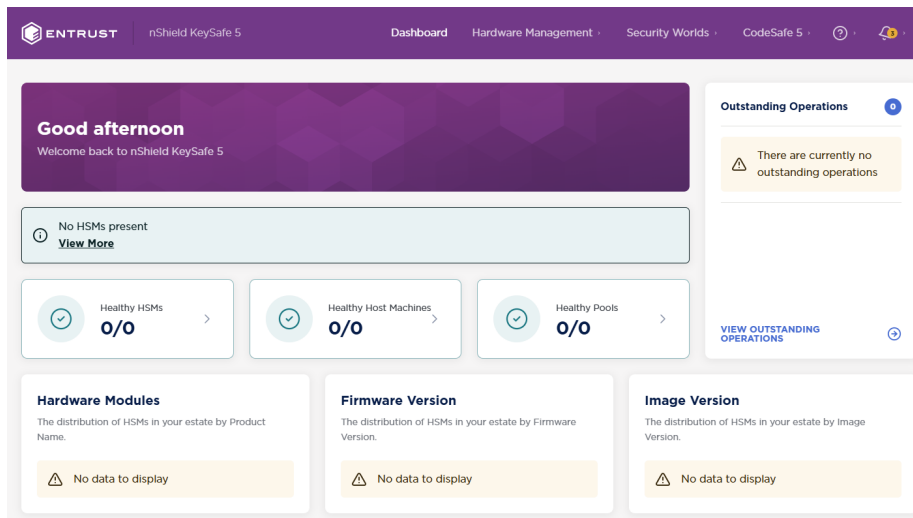
Generated TLS certificates are valid for 30 days. See the Installation Guide for how to update certificates.

## 12. Point your browser to the deployment URL:

```
https://<keysafe5-server-ip>
```

## 13. The KeySafe 5 dashboard appears:





14. The deploy script will also produce two archives for the agent configuration, one for Linux, one for Windows.

These files contain the agent configuration file and CA certificates for TLS authentication to RabbitMQ. The contents are used for configuring the KeySafe 5 client machines.

15. Configure `kubectl` access, which will be needed later during the integration:

```
% mkdir -p ${HOME}/.kube
% sudo /usr/local/bin/k3s kubectl config view --raw > ${HOME}/.kube/config
% chmod 600 ${HOME}/.kube/config
% export KUBECONFIG=${HOME}/.kube/config
```

16. Add the following line to the user's profile `~/.bash_profile` so KUBECONFIG is set every time:

```
export KUBECONFIG=${HOME}/.kube/config
```

## 2.2.2. Client installation - quick example

In this section we will go over a quick installation of the KeySafe 5 agent (client) on a RedHat 9 Linux server. For the purpose of this integration, we install the KeySafe 5 agent on a server that already has security world installed and configured with some keys. We demonstrate KeySafe 5 with the data from this server. In our scenario we have a RedHat Linux 9 server already setup and running and the integration picks up from the point where the KeySafe 5 agent is installed.

To configure the client machine to be managed and monitored by the KeySafe 5 server, we install the KeySafe 5 agent on this server:

1. Transfer the `nshield-keysafe5-1.4.0.tar.gz` file to the client server.
2. Create a directory for the installer:

```
% mkdir keysafe5-install
```

3. Untar the KeySafe 5 `tar` file into this directory:

```
% tar -xf nshield-keysafe5-1.4.0.tar.gz -C keysafe5-install
```

4. Change to the install directory:

```
% cd keysafe5-install
```

5. Create the installation directory:

```
% sudo mkdir -p /opt/nfast/keysafe5
```

6. Transfer the `agent-config.tar.gz` that `deploy.sh` generated to the client server:

```
% scp <keysafe5-server-ip>:/home/<USER>/keysafe5-install/agent-config.tar.gz .
```

7. Untar the agent into `/opt/nfast/keysafe5`:

```
% sudo tar -xf keysafe5-agent/keysafe5-1.4.0-Linux-keysafe5-agent.tar.gz -C /
```

8. Untar the agent configuration, `agent-config.tar.gz`:

```
% sudo tar -xf agent-config.tar.gz -C /opt/nfast/keysafe5/
```

9. Generate a TLS private key for this KeySafe 5 agent.:

```
% /opt/nfast/keysafe5/bin/ks5agenttls -keypath=/opt/nfast/keysafe5/conf/messagebus/tls/tls.key -keygen
```

#### Output:

```
Will use nShield HSM as source of randomness for private key data
Private key has been generated and saved to /opt/nfast/keysafe5/conf/messagebus/tls/tls.key

When configuring message bus TLS for this KeySafe 5 agent, the key should be saved to
/opt/nfast/keysafe5/conf/messagebus/tls/tls.key with file permissions and ownership as documented in the
KeySafe 5 Installation Guide
```

- 
10. Generate a TLS CSR for this KeySafe 5 agent:

```
% /opt/nfast/keysafe5/bin/ks5agenttls -keypath=/opt/nfast/keysafe5/conf/messagebus/tls/tls.key -csrgen
```

Output:

```
CSR has been generated and saved to ks5agent_openssl-redhat-9.csr
```

11. Copy the CSR file to the KeySafe 5 server **keysafe5-install** directory:

```
% scp ks5agent_openssl-redhat-9.csr <keysafe5-server-ip>:/home/<USER>/keysafe5-install/.
```

12. Go back to the KeySafe 5 server:

13. Run **agentcert.sh** to create a client TLS certificate for the KeySafe 5 client (agent) using the CSR from the client and the CA created by the deploy script:

```
% cd /home/<user>/keysafe5-install
% agentcert.sh ks5agent_openssl-redhat-9.csr 365
```

```
Certificate generated to ks5agent_openssl-redhat-9.crt. Valid for 365 days
CA Certificate available at /home/<user>/keysafe5-install/internalCA/cacert.pem
```

```
RabbitMQ will need to be configured to allow access for the user 'ks5agent_openssl-redhat-9':
export RUN_RABBIT="kubectl -n rabbitns exec rabbit-chart-rabbitmq-0 -c rabbitmq -- "
${RUN_RABBIT} rabbitmqctl add_user ks5agent_openssl-redhat-9 ephemeralpw
${RUN_RABBIT} rabbitmqctl set_permissions -p nshieldvhost ks5agent_openssl-redhat-9 '.*' '.*' '.*'
${RUN_RABBIT} rabbitmqctl cclear_password ks5agent_openssl-redhat-9
```

14. In the KeySafe 5 server, Run the commands requested to configure RabbitMQ:

```
% export RUN_RABBIT="kubectl -n rabbitns exec rabbit-chart-rabbitmq-0 -c rabbitmq -- "
% ${RUN_RABBIT} rabbitmqctl add_user ks5agent_openssl-redhat-9 ephemeralpw
```

Output:

```
Adding user "ks5agent_openssl-redhat-9" ...
Done. Don't forget to grant the user permissions to some virtual hosts! See 'rabbitmqctl help
set_permissions' to learn more.
```

```
% ${RUN_RABBIT} rabbitmqctl set_permissions -p nshieldvhost ks5agent_openssl-redhat-9 '.*' '.*' '.*'
```

Output:

```
Setting permissions for user "ks5agent_openssl-redhat-9" in vhost "nshieldvhost" ...
```

```
% ${RUN_RABBIT} rabbitmqctl clear_password ks5agent_openssl-redhat-9
```

#### Output:

```
Clearing password for user "ks5agent_openssl-redhat-9" ...
```

15. In the KeySafe 5 server, copy the client TLS certificate output from the `agentcert.sh` command to the client server with the KeySafe 5 agent:

```
% scp ks5agent_openssl-redhat-9.crt <KEYSAFE5_CLIENT_IP>:/home/<USER>/keysafe5-install/.
```

16. In the KeySafe 5 server, also copy over the CA certificate `internalCA/cacert.pem`:

```
% scp internalCA/cacert.pem <KEYSAFE5_CLIENT_IP>:/home/<USER>/keysafe5-install/.
```

17. Go back to the KeySafe 5 client machine.

18. Copy `ks5agent_openssl-redhat-9.crt` to `/opt/nfast/keysafe5/conf/messagebus/tls/tls.crt`:

```
% sudo cp /home/<USER>/keysafe5-install/ks5agent_openssl-redhat-9.crt  
/opt/nfast/keysafe5/conf/messagebus/tls/tls.crt
```

19. Copy the `cacert.pem` to `/opt/nfast/keysafe5/conf/messagebus/tls/ca.crt`:

```
% sudo cp /home/<USER>/keysafe5-install/cacert.pem /opt/nfast/keysafe5/conf/messagebus/tls/ca.crt
```

20. Edit the `/opt/nfast/keysafe5/conf/config.yaml` file:

Set the URL to `<KEYSAFE5_CLIENT_IP>:5671/nshieldvhost` and `message_bus` to `amqp`

```
message_bus:  
  # What technology to use for the message bus.  
  # Supported Values:  
  # - amqp  
  # - nats  
  # The default is nats  
  type: amqp
```

21. If the hardserver is already running, use the KeySafe 5 install script to start the KeySafe 5 agent without restarting the hardserver:

```
% sudo /opt/nfast/keysafe5/sbin/install
```

Output:

```
-- Running install fragment keysafe5-agent
Checking for user 'keysafe5d'
Creating keysafe5d user.
useradd: warning: the home directory /opt/nfast already exists.
useradd: Not copying any file from skel directory into it.
Checking user 'keysafe5d' is in correct group 'nfast'
users created correctly
Adding 'keysafe5d' user to 'nfastadmin' group
Enforcing permissions on existing KeySafe5 agent configuration file
Setting group ownership of '/opt/nfast/keysafe5/conf/config.yaml' to 'nfastadmin' group
Enforcing permissions on KeySafe5 agent message bus configuration files
Installing startup scripts for 'keysafe5-agent'.
Enabling the systemd service unit
Adding and enabling a systemd unit
Created symlink /etc/systemd/system/multi-user.target.wants/nc_keysafe5-agent.service →
/etc/systemd/system/nc_keysafe5-agent.service.
Starting nCipher 'keysafe5-agent' server process.
---- Installation complete ----
```

Otherwise use the nShield install script that starts all the services:

```
% sudo /opt/nfast/sbin/install
```

If you have to start or stop the **keysafe5** agent, use:

```
% sudo /opt/nfast/scripts/init.d/keysafe5-agent [start/stop]
```

## 22. View the KeySafe 5 agent log:

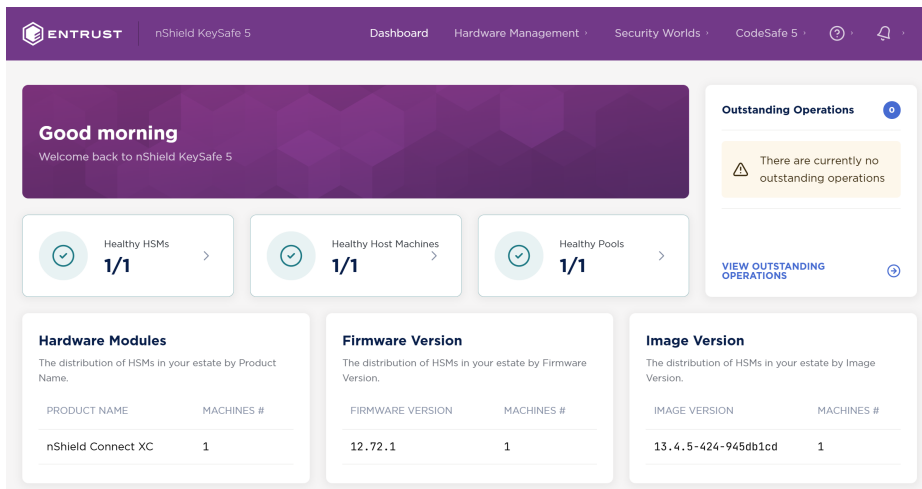
```
% sudo tail -50 /opt/nfast/log/keysafe5-agent.log
```

Output:

```
{"component":"KEYSAFE5-AGENT","level":"info","msg":"Starting agent with config: Hostname:openssl-redhat-9,
Version:1.4.0-b69fc133, MessageBus:{type: amqp, URL: amqps://<keysafe5-server-ip>:5671/nshieldvhost, tls:
true}, LoggerConfig:{level:Info, format:JSON, file.enabled:true, file.path:/opt/nfast/log/keysafe5-
agent.log}, UpdateInterval:1m0s, HealthInterval:1m0s, RecoveryInterval:5s, KmdataNetworkMount:false,
KmdataPollInterval:1s, CodeSafeUpdateInterval:3m0sCodeSafeCachePeriod:1h0m0s","pid":94643,"time":"2024-10-
30 11:28:34.204"}
{"component":"KEYSAFE5-AGENT","level":"info","msg":"Recovery necessary","pid":94643,"time":"2024-10-30
11:28:34.205"}
{"component":"KEYSAFE5-AGENT","level":"info","msg":"Started watching for changes in
/opt/nfast/kmdata/local","pid":94643,"time":"2024-10-30 11:28:34.255"}
{"component":"KEYSAFE5-AGENT","level":"info","msg":"Published HSM Data update for module: 1, esn: 5xxx-
xxxx-xxxx (466 B)","pid":94643,"time":"2024-10-30 11:28:34.259"}
{"component":"KEYSAFE5-AGENT","level":"info","msg":"Starting nCore handler on 5xxx-xxxx-xxxx_ncore_request
for module: 1, esn: 5xxx-xxxx-xxxx","pid":94643,"time":"2024-10-30 11:28:34.259"}
{"component":"KEYSAFE5-AGENT","level":"info","msg":"Starting feature certificate handler for openssl-
redhat-9","pid":94643,"time":"2024-10-30 11:28:34.269"}
{"component":"KEYSAFE5-AGENT","level":"info","msg":"Starting platform handler for openssl-redhat-
```

```
9", "pid": 94643, "time": "2024-10-30 11:28:34.269"}
{"component": "KEYSAFE5-AGENT", "level": "info", "msg": "Published host[openssl-redhat-9] update (20 KiB), containing hsm: [5xxx-xxxx-xxxx]", "pid": 94643, "time": "2024-10-30 11:28:34.274"}
{"component": "KEYSAFE5-AGENT", "level": "info", "msg": "Published kmdata update for world hknso 0e41xxxxxxxx6c2315xxxx451xxx. keys=7 softcards=1 cardsets=2 cards=10 modulecerts=2 (54 KiB)", "pid": 94643, "time": "2024-10-30 11:28:34.488"}
{"component": "KEYSAFE5-AGENT", "level": "info", "msg": "Starting kmdata handler for openssl-redhat-9", "pid": 94643, "time": "2024-10-30 11:28:34.488"}
{"component": "KEYSAFE5-AGENT", "level": "info", "msg": "No recovery necessary. hardserver running = true", "pid": 94643, "time": "2024-10-30 11:28:39.270"}
```

23. Once deployed you should see the HSM and Security World information from the KeySafe 5 client populated into the KeySafe 5 server:



---

# Chapter 3. Integrate Entrust KeySafe 5 with the KeyControl Compliance Manager server

## 3.1. Prerequisites

- KeyControl Compliance Manager server has been deployed and configured ([Install and configure KCM and KeySafe 5](#)).
- KeySafe 5 has been deployed and configured ([Install and configure KCM and KeySafe 5](#)).
- A KeySafe 5 client has been deployed and configured with the HSM ([Install and configure KCM and KeySafe 5](#)).

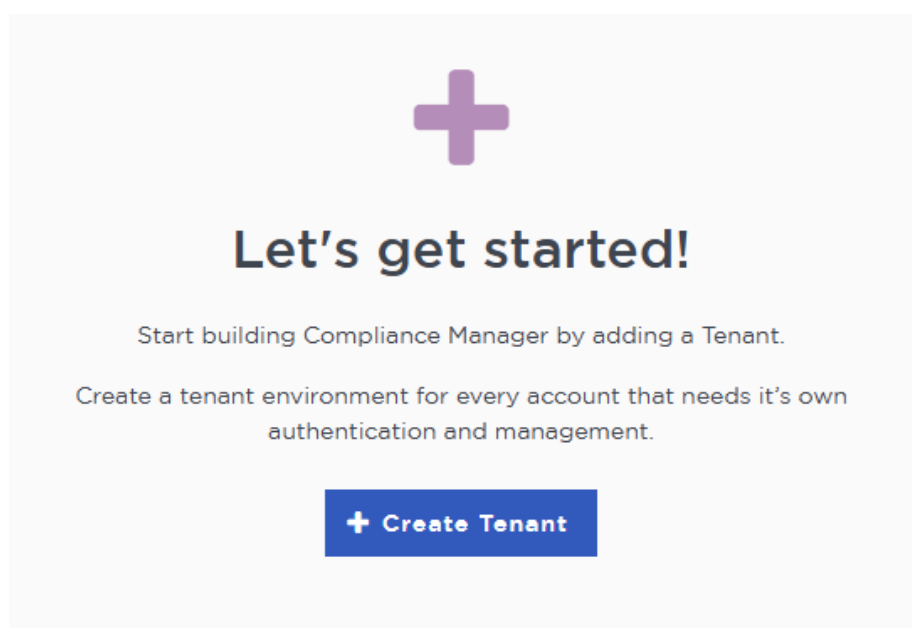
## 3.2. Create the KCM tenant

To be able to use the KeyControl Compliance Manager you must create a tenant that can be used for the integration.

1. Point your browser to the KeyControl Compliance Manager (KCM) URL:  
<https://<kcm-server-ip>/kcm>.

Sign in using the **secroot** credentials setup during the KCM installation.

2. Select the **Create Tenant** button.



3. fill out the **Create Tenant** form:

### Tenants

Each tenant has unique authentication and management

---

#### Create Tenant

A tenant will be an instance of Compliance Manager with unique authentication and management.

**Name\***

**Description**

---

**Email Notifications** OFF

**SMTP needs to be configured to turn on email notifications**

Use email to communicate with Tenant Administrators, including their temporary passwords. Turning off email notifications means you will see and need to give temporary passwords to the Tenant Admin.

---

**Administrator**

Invite an individual to have complete access and control over this Tenant. They will be responsible for inviting additional members.

**Admin Name\***

**Admin Email\***

Email will be used as the User Name when logging into the Tenant.

---

**Create Tenant** Cancel


4. Select **Create Tenant**.


5. When the tenant creation completes, a message is displayed:

## ✔ Successfully Created KeySafe5!

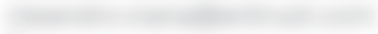
The Tenant has been successfully created. The following information will need to be given to the Tenant Administrator:


**Tenant URL**




 [Copy](#)


**User Name**



 [Copy](#)

**Temporary Password**



 [Copy](#)

**Close**



A temporary password is emailed to the administrator's email address. This is the password for signing in for the first time to the tenant space in KCM. In a closed-gap environment where email is not available, the password for the user is displayed when you first create the vault, then it can be copied and sent to the user.

6. Select **Close**.

The new tenant is displayed in the Vault dashboard.

7. To view the details on the tenant, select **View Details** when you mouse over the tenant.

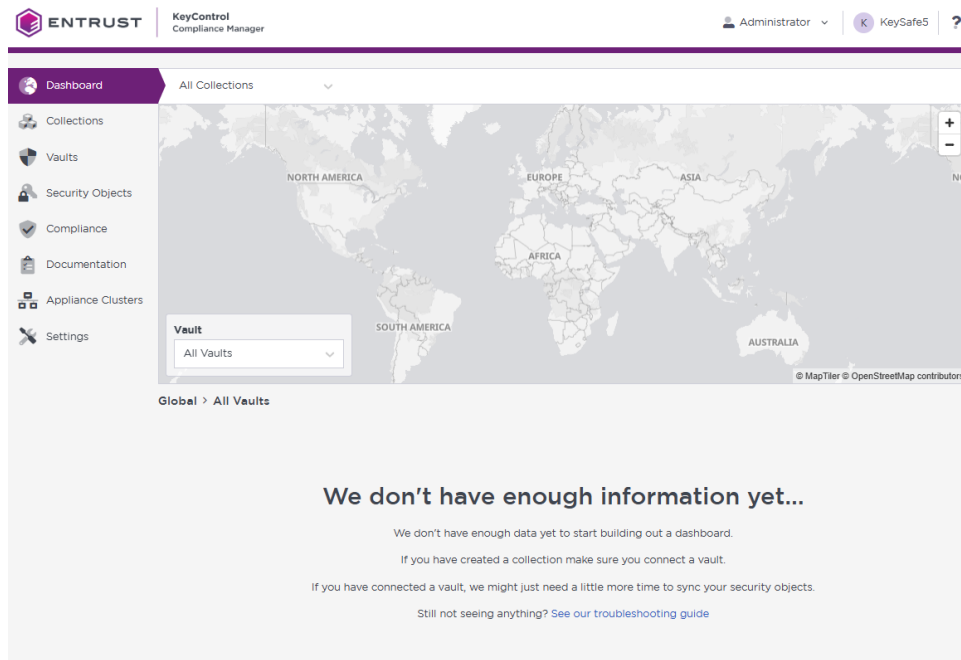
8. Select the **URL** to access the tenant's page.

```
https://<kcm-server-ip>/login/kcm/75f226ec-4730-42eb-85b3-157fc8b3467a/
```

9. Sign in with the password that was copied when you created the tenant or the password that was emailed to you.

10. The system makes you change the temporary password and to sign back in.

The KeyControl Compliance Manager Dashboard is displayed:



### 3.3. Create the KeySafe 5 app link token key

To connect KeySafe 5 and KCM, you need an app link token key, which you have to create on the KeySafe 5 client server.

1. Sign in to the KeySafe 5 client server.
2. Generate the key.

In this example, we create the key in the HSM, using module protection:

```
% sudo /opt/nfast/bin/generatekey simple module=1 protect=module type=AES size=256 ident=cmapplinktokenprot plainname=applinktokenprot nvrाम=no
```

Output:

```
key generation parameters:
operation  Operation to perform      generate
application Application                 simple
protect    Protected by               module
verify     Verify security of key    yes
type       Key type                   AES
size       Key size                   256
ident      Key identifier             cmapplinktokenprot
plainname  Key name                   applinktokenprot
nvrाम      Blob in NVRAM (needs ACS) no

Key successfully generated.
Path to key: /opt/nfast/kmdata/local/key_simple_cmapplinktokenprot
```

If you are using a FIPS Level 3 security world, you will need to provide an OCS card for FIPS authorization during the key creation.

### 3.4. Capture the KCM TLS Certificate chain and load in the KeySafe 5 server

To connect KeySafe 5 and KCM, you have to capture the KCM TLS certificate chain and load in the KeySafe 5 server.

On the KeySafe 5 server, create a secret with the KCM TLS certificate chain.

1. Change to the install `keysafe5-install` folder:

```
% cd keysafe5-install
```

2. Capture the KCM TLS certificate chain (Issuer + Root) certificates, for example using OpenSSL, and store it in `compliance-ca.pem`:

```
% openssl s_client -connect <kcm-server-ip>:443 -showcerts
```

Output:

```

CONNECTED(00000003)
Can't use SSL_get_servername
depth=1 C = US, O = Hytrust Inc., CN = KeyControl Compliance Manager Certificate Authority
verify error:num=19:self-signed certificate in certificate chain
verify return:1
depth=1 C = US, O = Hytrust Inc., CN = KeyControl Compliance Manager Certificate Authority
verify return:1
depth=0 C = US, O = HyTrust Inc., CN = kcm-1031.interop.local
verify return:1
---
Certificate chain
 0 s:C = US, O = HyTrust Inc., CN = kcm-1031.interop.local
  i:C = US, O = Hytrust Inc., CN = KeyControl Compliance Manager Certificate Authority
  a:PKEY: rsaEncryption, 2048 (bit); sigalg: RSA-SHA256
  v:NotBefore: Jun  1 00:00:00 2011 GMT; NotAfter: Dec 31 23:59:59 2049 GMT
-----BEGIN CERTIFICATE-----
MII
...
gFFAjK0g=
-----END CERTIFICATE-----
 1 s:C = US, O = Hytrust Inc., CN = KeyControl Compliance Manager Certificate Authority
  i:C = US, O = Hytrust Inc., CN = KeyControl Compliance Manager Certificate Authority
  a:PKEY: rsaEncryption, 2048 (bit); sigalg: RSA-SHA256
  v:NotBefore: Jun  1 00:00:00 2011 GMT; NotAfter: Dec 31 23:59:59 2049 GMT
-----BEGIN CERTIFICATE-----
MIIEG
...
8jYM6yT+w=
-----END CERTIFICATE-----
---
Server certificate
subject=C = US, O = HyTrust Inc., CN = kcm-1031.interop.local
issuer=C = US, O = Hytrust Inc., CN = KeyControl Compliance Manager Certificate Authority
---
No client certificate CA names sent
Peer signing digest: SHA512
Peer signature type: RSA
Server Temp Key: ECDH, prime256v1, 256 bits
---
SSL handshake has read 2718 bytes and written 421 bytes
Verification error: self-signed certificate in certificate chain
---
New, TLSv1.2, Cipher is ECDHE-RSA-AES256-GCM-SHA384
Server public key is 2048 bit
Secure Renegotiation IS supported
Compression: NONE
Expansion: NONE
No ALPN negotiated
SSL-Session:
    Protocol : TLSv1.2
    Cipher   : ECDHE-RSA-AES256-GCM-SHA384
    Session-ID: 165B25EF141F15205AABFA039C4E7CEE81EFFDDF63C5EA6058E2BC479F5CB18
    Session-ID-ctx:
    Master-Key:
E9D4E199127778CC3118AF5681CB3E33131E15219A045B9BD48BCDC88EC59F4E9C1FB03D49CA64039F093E14A1506050
    PSK identity: None
    PSK identity hint: None
    SRP username: None
    TLS session ticket lifetime hint: 300 (seconds)
    TLS session ticket:
0000 - 34 91 0d 87 a6 91 b5 a7-da ba e1 b0 4d f4 1c 10  4.....M...
0010 - ef 7a 1b cf 80 42 ff 5f-0c 8a 68 24 38 7c ec 76  .z...B...h$8|.v
0020 - d2 66 c6 a0 63 e7 2d 28-91 c9 b3 07 c8 c6 2c 81  .f..c.-(...,..
0030 - 20 18 15 cc 81 8e fb f9-25 62 b8 7d 7e 54 7d 53  .....%b.)~T}S
0040 - 0a 6f 22 3f f6 5d 91 fc-c6 a0 41 32 cd 5a 84 e8  .o"?.]...A2.Z..
0050 - d6 12 cc 8e a5 69 ac f4-e0 1d d0 ae 01 c7 fe 9e  .....i.....

```

```

0060 - 4d 4f a1 80 bd a4 75 11-db 55 71 35 cb f0 87 ab MO....u..Uq5....
0070 - a4 41 2d f0 2a e5 13 9b-b9 52 0a 10 fd 5e 71 2e .A-.*....R...^q.
0080 - 39 8f 88 57 45 9d b9 da-8b 0d 7f 05 a4 40 e7 ad 9..WE.....@..
0090 - f1 bd 67 bf 80 74 55 74-8a 17 64 07 88 03 62 cf ..g..tUt..d...b.
00a0 - 60 53 06 fb dd 50 fc 2c-18 8b 99 db 14 14 94 dd `S...P.,.....
00b0 - 55 0d 71 5a 64 73 89 98-a2 76 02 3a cd 1a 45 d6 U.qZds...v:...E.

Start Time: 1730470418
Timeout : 7200 (sec)
Verify return code: 19 (self-signed certificate in certificate chain)
Extended master secret: no
---
closed
    
```

You want to use the server CA certificate which in this case is the second certificate in the output above. Save this certificate in a file named **compliance-ca.pem**.

```

-----BEGIN CERTIFICATE-----
MIIE
...
8jYM6fvyT+w=
-----END CERTIFICATE-----
    
```

3. Create a Kubernetes configmap for the KCM TLS certificate:

```
% kubectl -n nshieldkeysafe5 create configmap compliance-ca --from-file=ca.pem=/path/to/compliance-ca.pem
```

Output:

```
configmap/compliance-ca created
```

4. Generate the KeySafe 5 back-end values and store them into a file named **keysafe5-backend-values.yaml**:

```
% helm -n nshieldkeysafe5 get values --all --output yaml keysafe5-backend > keysafe5-backend-values.yaml
% cat keysafe5-backend-values.yaml
```

Output:

```

appLabel: keysafe5-backend-app
cache:
  itemTTL: 168h
  peers: keysafe5-headless:3322
  replicaCount: 3
codesafe_mgmt:
  dbName: codesafe-mgmt-db
  image: localhost:5000/keysafe5/codesafe-mgmt:1.4.0
livenessProbe:
  failureThreshold: 3
  initialDelaySeconds: 5
  periodSeconds: 60
    
```

```
    successThreshold: 1
  pullPolicy: IfNotPresent
  readinessProbe:
    failureThreshold: 3
    initialDelaySeconds: 5
    periodSeconds: 10
    successThreshold: 1
  database:
    mongo:
      auth:
        authDatabase: authdb
        existingSecret: ""
        type: tls
      connectTimeout: 30s
      hosts: mongo-chart-mongodb-0.mongo-chart-mongodb-headless.mongons.svc.cluster.local:27017,mongo-chart-
mongodb-1.mongo-chart-mongodb-headless.mongons.svc.cluster.local:27017
      maxPoolSize: 100
      minPoolSize: 1
      replicaSet: rs0
      selectionTimeout: 30s
      socketTimeout: 30s
      tls:
        cipherSuites:
          - ECDHE-ECDSA-AES128-GCM-SHA256
          - ECDHE-RSA-AES128-GCM-SHA256
          - ECDHE-ECDSA-AES256-GCM-SHA384
          - ECDHE-RSA-AES256-GCM-SHA384
          - ECDHE-ECDSA-CHACHA20-POLY1305
          - ECDHE-RSA-CHACHA20-POLY1305
        enabled: true
        existingSecret: mongodb-demo-client-certificates
        minProtocolVersion: TLSV1_2
      timeout: 30s
      type: mongo
  global:
    imagePullSecrets: []
  health:
    allowedClockSkew: 2m
    internalHealthTimeoutPeriod: 10s
    internalHealthUpdatePeriod: 30s
    livenessFailurePeriod: 5m
  hsm_mgmt:
    dbName: hsm-mgmt-db
    image: localhost:5000/keysafe5/hsm-mgmt:1.4.0
    livenessProbe:
      failureThreshold: 3
      initialDelaySeconds: 5
      periodSeconds: 60
      successThreshold: 1
    pullPolicy: IfNotPresent
    readinessProbe:
      failureThreshold: 3
      initialDelaySeconds: 5
      periodSeconds: 10
      successThreshold: 1
  httpServer:
    cleanupTimeout: 30s
    maxHeaderBytes: "1048576"
    readTimeout: 5m
    writeTimeout: 8m
  integrations:
    kcm:
      caConfigMap: ""
  logging:
    format: JSON
    level: INFO
  messageBus:
```

```

URL: rabbit-chart-rabbitmq.rabbitns.svc.cluster.local:5671/nshieldvhost
auth:
  existingSecret: ""
  type: tls
tls:
  cipherSuites:
    - ECDHE-ECDSA-AES256-GCM-SHA384
    - ECDHE-RSA-AES256-GCM-SHA384
    - ECDHE-ECDSA-AES128-GCM-SHA256
    - ECDHE-RSA-AES128-GCM-SHA256
    - ECDHE-ECDSA-CHACHA20-POLY1305
    - ECDHE-RSA-CHACHA20-POLY1305
  enabled: true
  existingSecret: rabbit-client-secret-20241029141648
  minProtocolVersion: TLSV1_2
  type: amqp
objectStore:
  pvc: data-nshield-keysafe5
podSecurityContext:
  fsGroup: 1001
  runAsGroup: 1001
  runAsUser: 1001
replicaCount: 3
sw_mgmt:
  dbName: sw-mgmt-db
  image: localhost:5000/keysafe5/sw-mgmt:1.4.0
livenessProbe:
  failureThreshold: 3
  initialDelaySeconds: 5
  periodSeconds: 60
  successThreshold: 1
pullPolicy: IfNotPresent
readinessProbe:
  failureThreshold: 3
  initialDelaySeconds: 5
  periodSeconds: 10
  successThreshold: 1

```

## 5. Upgrade the KeySafe 5 backend:

Ensure that the `nshield-keysafe5-backend-1.X.X.tgz` is correct version.

```
% helm upgrade --install keysafe5-backend --namespace=nshieldkeysafe5 --values keysafe5-backend-values.yaml
--set integrations.kcm.caConfigMap=compliance-ca helm-charts/nshield-keysafe5-backend-1.4.0.tgz
```

### Output:

```

Release "keysafe5-backend" has been upgraded. Happy Helming!
NAME: keysafe5-backend
LAST DEPLOYED: Fri Nov 1 10:42:21 2024
NAMESPACE: nshieldkeysafe5
STATUS: deployed
REVISION: 3
TEST SUITE: None
NOTES:
nShield KeySafe 5 backend services

The installed nshield-keysafe5-backend release is named keysafe5-backend in namespace nshieldkeysafe5.

To view configuration values used for this install:
helm --namespace nshieldkeysafe5 get values keysafe5-backend

```

### Configuring External Access

This helm chart installs the KeySafe 5 services into a Kubernetes cluster but does not configure external access to the services. To access the services you will need to configure ingress to your cluster and routing of requests to the `keysafe5-headless` Kubernetes Service.

To configure an Istio Ingress Controller, use the KeySafe 5 Istio Helm chart:

1. Install the KeySafe 5 Istio helm chart. This chart contains pre-defined routes to the ClusterIP services described above (see chart README.md for configuration options)  
> helm install keysafe5-istio helm-keysafe5-istio/

2. Determine the external IP address of the Istio Ingress Gateway

```
> INGRESS_IP=$(kubectl get svc -n istio-system -l app=istio-ingressgateway -o jsonpath='{.items[0].status.loadBalancer.ingress[0].ip}')
```

The backend services would then be accessible at:

```
    HSM Management: https://$INGRESS_IP/mgmt/v1/hsms
    Host Management: https://$INGRESS_IP/mgmt/v1/hosts
    Pool Management: https://$INGRESS_IP/mgmt/v1/pools
    Feature Certificate Management: https://$INGRESS_IP/mgmt/v1/feature-certificates
    Security World Management: https://$INGRESS_IP/mgmt/v1/worlds
    CodeSafe Management: https://$INGRESS_IP/codesafe/v1
```

To configure other Kubernetes Ingress Controllers you must configure routing to the `keysafe5-headless` service in `nshieldkeysafe5` namespace, and any authentication or authorization to access the services.

Alternatively, for local testing:

To access the HSM/Pool Management API:

```
kubectl port-forward --namespace nshieldkeysafe5 svc/keysafe5-headless 18080:18080
curl -X GET http://127.0.0.1:18080/mgmt/v1/hsms
curl -X GET http://127.0.0.1:18080/mgmt/v1/hosts
curl -X GET http://127.0.0.1:18080/mgmt/v1/pools
curl -X GET http://127.0.0.1:18080/mgmt/v1/feature-certificates
```

To access the Security World Management API:

```
kubectl port-forward --namespace nshieldkeysafe5 svc/keysafe5-headless 18081:18081
curl -X GET http://127.0.0.1:18081/mgmt/v1/worlds
```

To access the CodeSafe Management API:

```
kubectl port-forward --namespace nshieldkeysafe5 svc/keysafe5-headless 18082:18082
curl -X GET http://127.0.0.1:18082/codesafe/v1/machines
```

## 3.5. Enable the connection between the KeySafe 5 and KCM

The connection between KeySafe 5 and KCM is based on a Security World basis. This allows a direct mapping one-to-one between the security world and nShield and a vault in KCM. That will allow in KCM for a customer to apply specific policies and documentation template on a vault by vault (security world by security world) basis. This allows for each security world to have its own compliance policy.

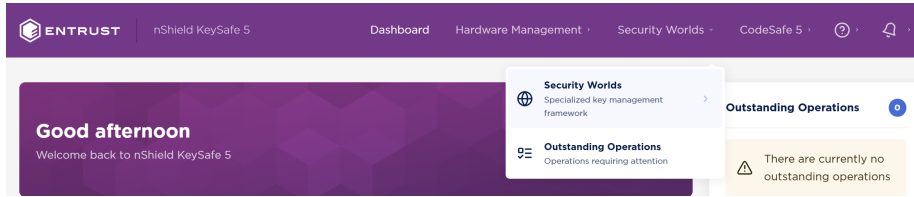
1. Make sure you have the following information from the KeyControl Compliance Manager:

- **KeyControl Compliance Manager ID**
- **KeyControl Compliance Manager Token**

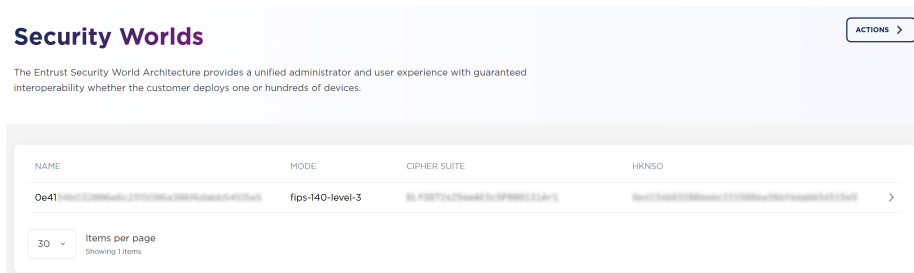
2. Point your browser to the KeySafe 5 deployment URL:

```
https://<keysafe5-server-ip>
```

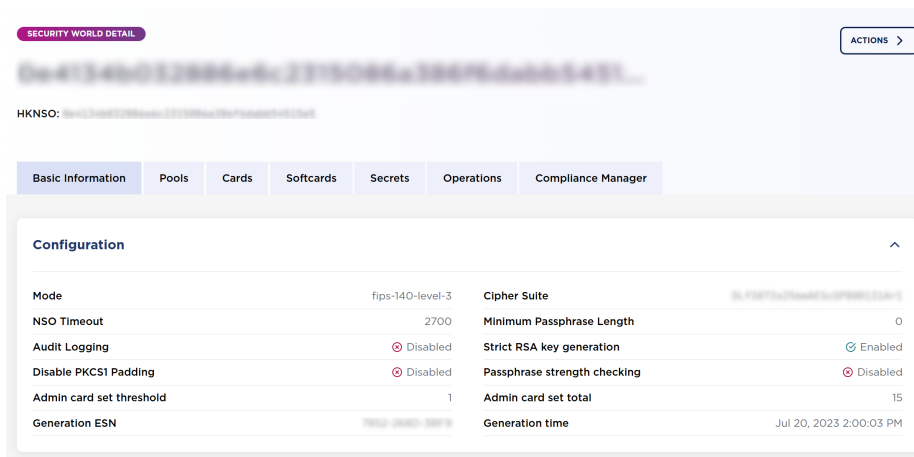
3. Select **Security Worlds** in the top bar, then select the **Security Worlds** menu option.



4. The list of security worlds is displayed:

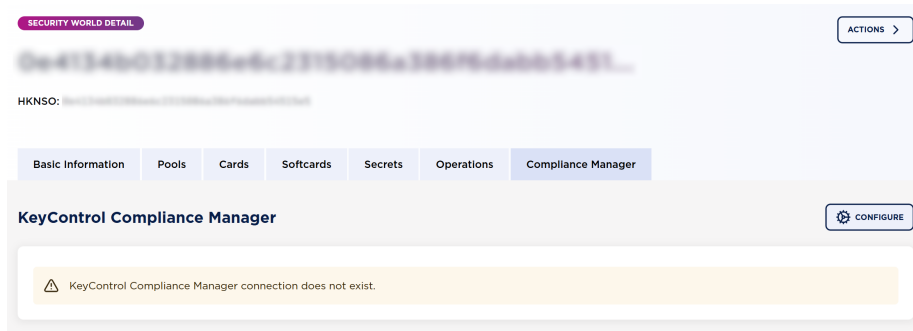


5. Select the security world listed that you will use to connect to Entrust KeyControl Compliance Manager.



6. Select **Compliance Manager >> Configure** to configure the connection with KCM.





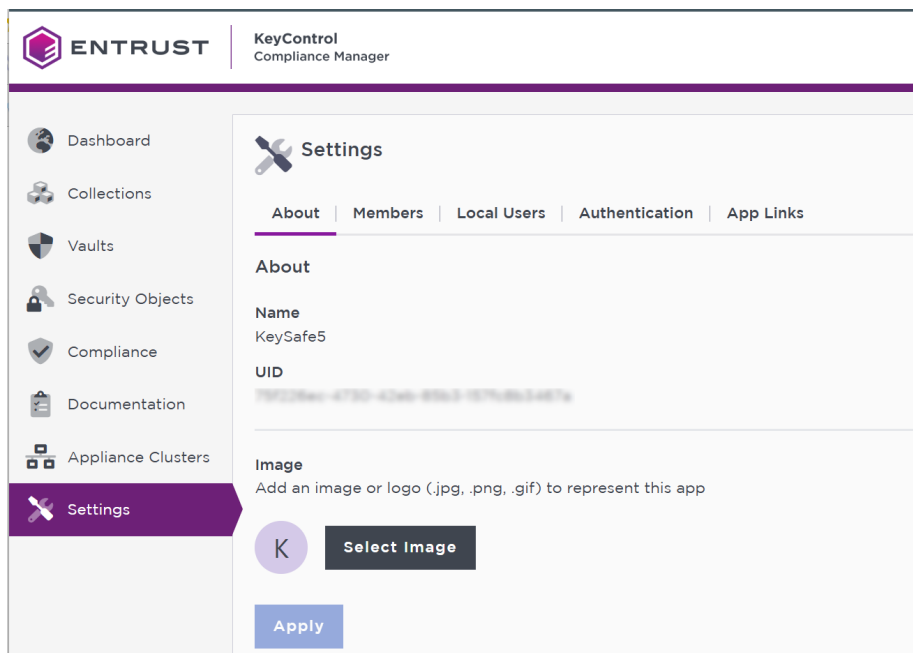
7. Enter the following information:

- **KeyControl Compliance Manager ID**
- **KeyControl Compliance Manager Token**

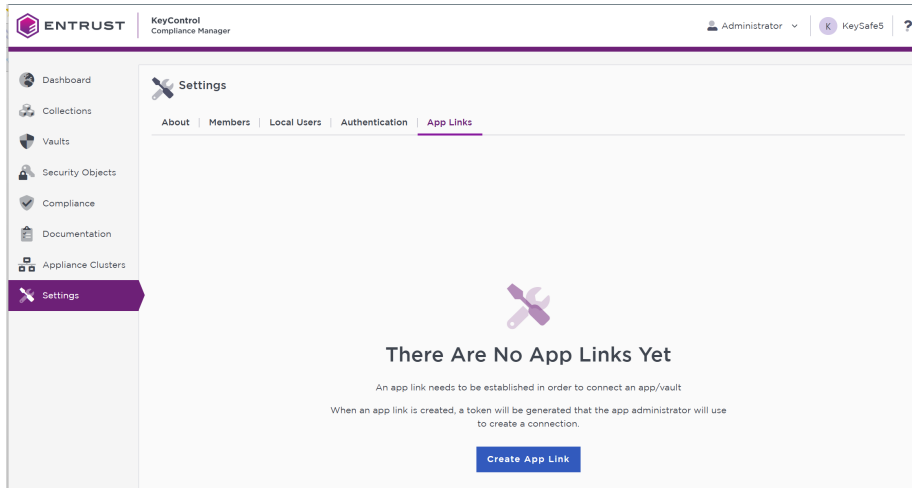
8. Navigate to the KeyControl Compliance Manager tenant URL created earlier and sign in:

`https://<kcm-server-ip>/login/kcm/75f226ec-4730-42eb-85b3-157fc8b3467a/`

9. In the main page, select **Settings** in the left pane.



10. In **Settings**, select **App Links**.



11. Select **Create App Link** and enter the information about the app:

### Create App Link

**Name \***  
Give this app link a name to easily identify where it will be used.

**Description**

Max 1024 characters

[Cancel](#) [Create](#)

12. Select **Create**.

This generates the token and the ID that KeySafe 5 needs.

---

## Copy Token

The following information will need to be added to the app that wants to connect to this Compliance Manager.

Make sure to copy this token and treat like a password. You want to keep it secure. **You will not be able to see this token again.**

### Token

Mz...FENw==

 Copy

### Compliance Manager ID

10.194.148.192

 Copy

Close

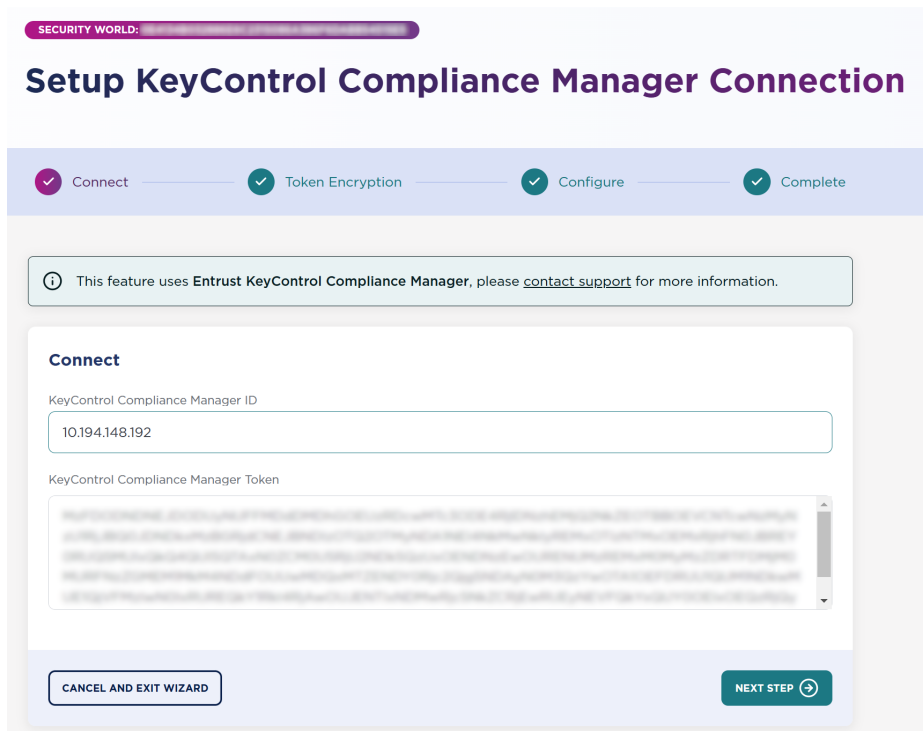
- Copy the **Token** and paste it back in the **Setup KeyControl Compliance Manager Connection** page in KeySafe 5:

Mz...FENw==

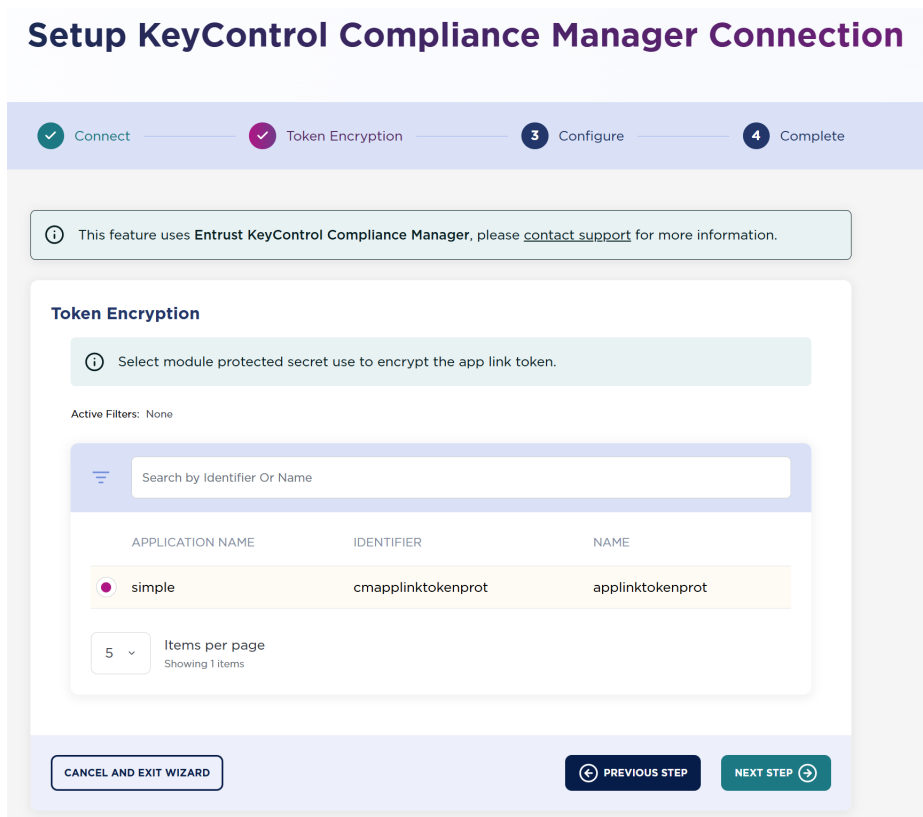
- Copy the **Compliance Manager ID** and paste it back in the **Setup KeyControl Compliance Manager Connection** page in KeySafe 5:

10.194.XXX.XXX

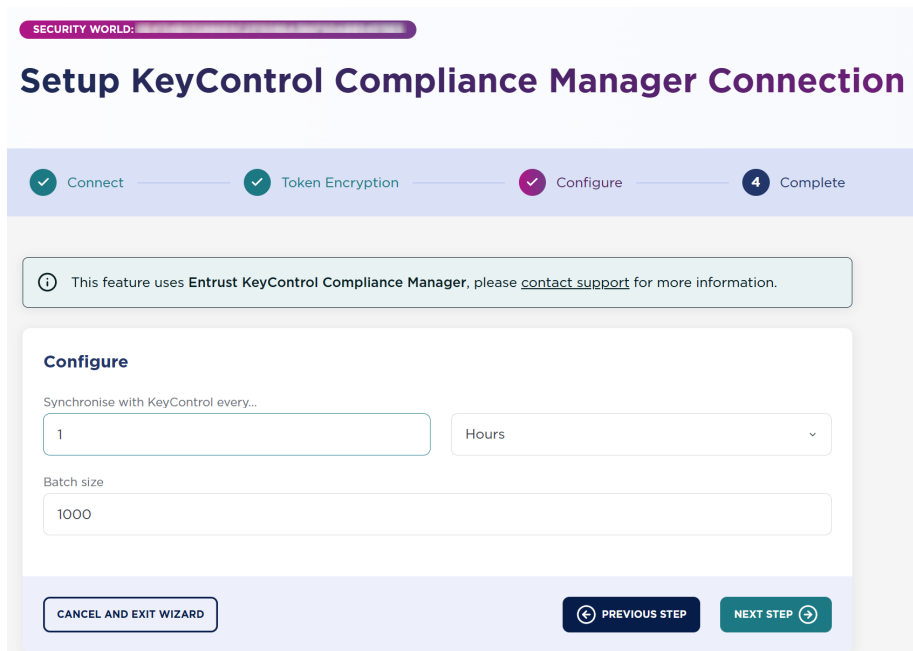
- Back in the **Setup KeyControl Compliance Manager Connection** page in KeySafe 5, then select **Next Step**.



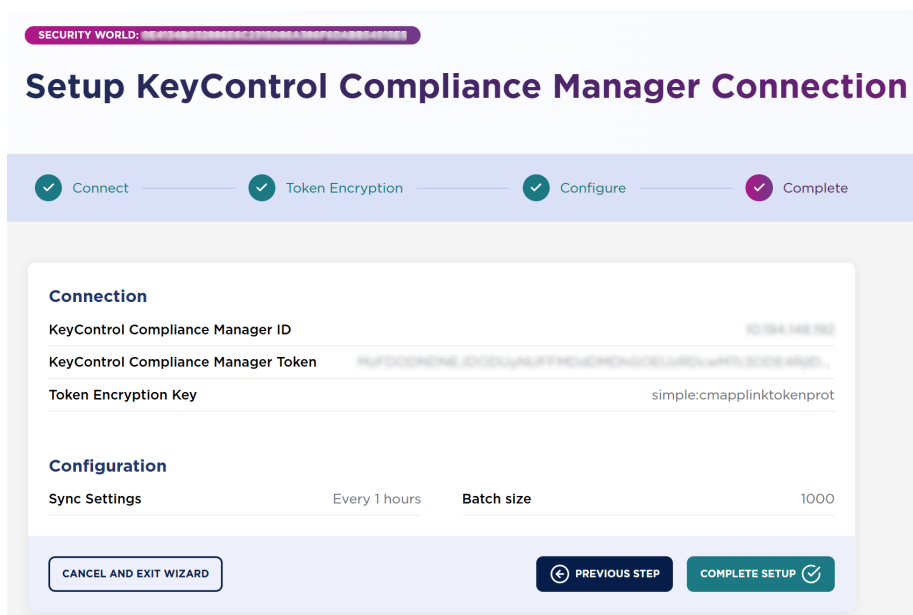
- 16. In **Token Encryption**, select the **App Link Token Key** that you created earlier, then select **Next Step**.



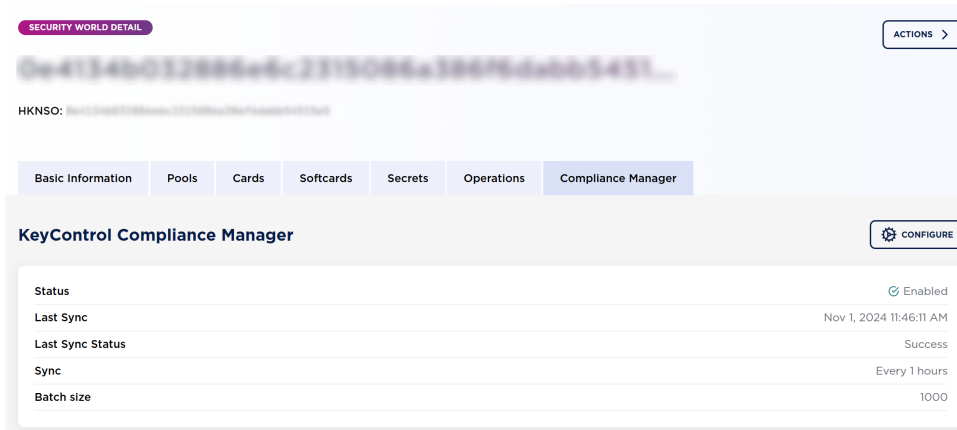
- 17. In **Configure**, select the synchronization time and the size, then select **Next Step**.



18. In **Complete**, review the configuration settings, then select **Complete Setup**.



The system should connect to KeyControl Compliance Manager and display the connection status.



Now a vault has been created in KCM and the secrets metadata in keysafe5 has been pushed out to the KCM vault. The display above shows the time of the last sync and when the next sync will occur.

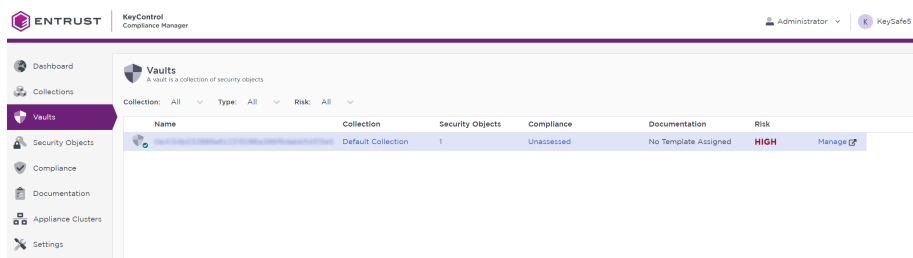
### 3.6. Validate the integration

The secrets have been pushed to the KeyControl Compliance Manager. We can check them there.

1. Navigate to the KeyControl Compliance Manager tenant URL that you created earlier and sign in:

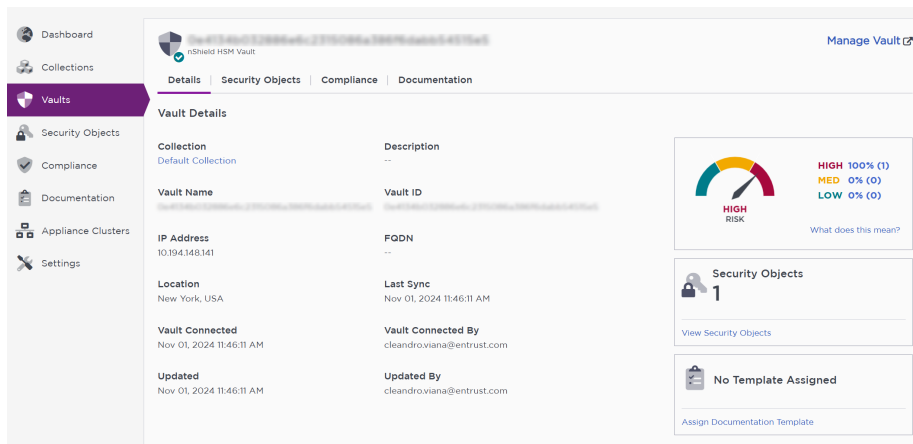
```
https://<kcm-server-ip>/login/kcm/75f226ec-xxxx-xxxx-xxxx-xxxxxxxxxx/
```

2. Select **Vaults** on the left pane to list the vaults and you should see one for the world we selected in KeySafe 5.

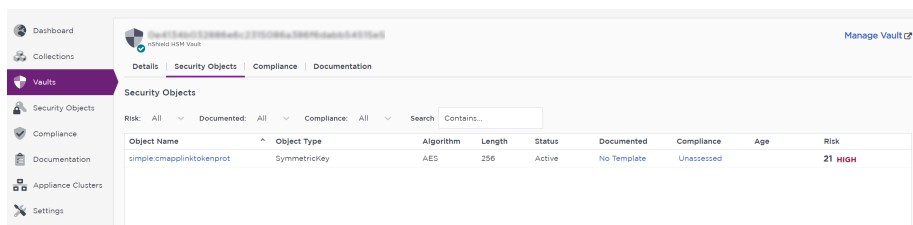


3. Select the vault for the security world.

You should see the vault details, when it was connected, the last time it was sync and so on.



4. Select the **Security Objects** tab to view the secret key we created earlier in the world.



Now that you have the security objects in KCM, you can apply specific policies and documentation template on a vault by vault (security world by security world) basis. This way each security world has its own compliance policy.

For information on how to use the compliance policies, see the KeyControl Compliance Manager online documentation.

## Chapter 4. Additional resources and related products

4.1. nShield as a Service

4.2. KeyControl

4.3. KeyControl as a Service

4.4. Entrust products

4.5. nShield product documentation