KeySafe 5

KeySafe 5 v1.5.0 Installation and Upgrade Guide

13 November 2025

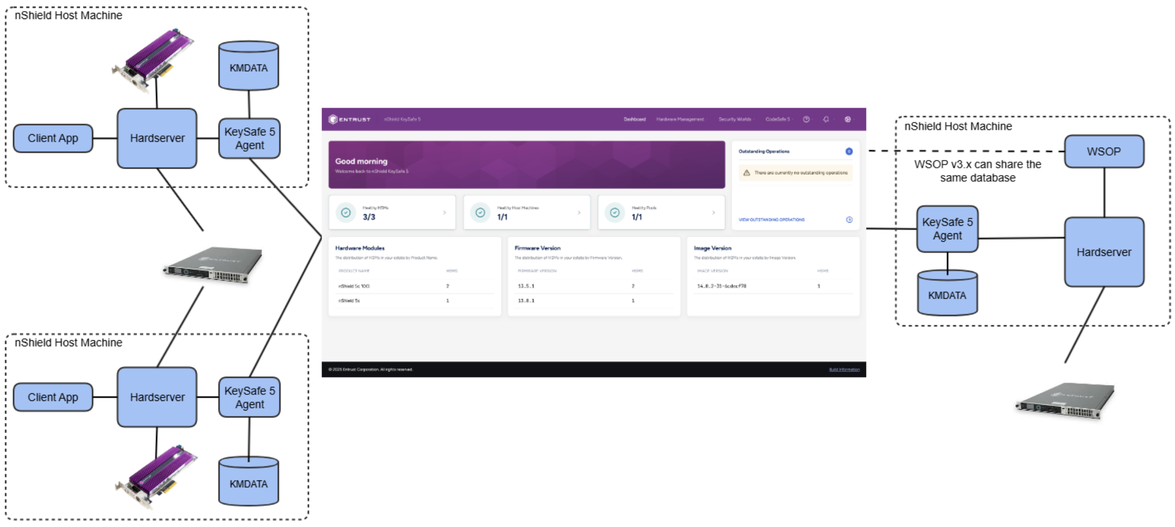
Table of Contents

1. Introduction

KeySafe 5 provides a centralized means to securely manage a distributed nShield HSM estate, including the creation and management of Security Worlds and associated resources (Softcards & Card Sets).

KeySafe 5 provides this capability in two forms: HTTP REST APIs for HSM Management and Security World management, and a graphical user interface. Only authenticated clients are permitted access to the service, providing assurance that your HSM and Security World data remain usable only by clients that are permitted access.

Typical KeySafe 5 deployment:



KeySafe 5 should be deployed as a Kubernetes application to manage a large estate of HSMs.

For each nShield client machine that you want to manage using this platform, you must install a KeySafe 5 agent binary alongside the existing nShield hardserver. A KeySafe 5 agent is installed on the nShield Connect for nShield Connect images released with Security World v13.4 and later software.

2. Release Package

The release package is provided in `.tar.gz` format and has the following contents.

2.1. OpenAPI specifications

The API specification documents for the RESTful web services follow v3.0 of the OpenAPI specification.

- `api/agent-mgmt.yml` defines the Agent Management API
- `api/codesafe-mgmt.yml` defines the CodeSafe Management API
- `api/hsm-mgmt.yml` defines the HSM Management API
- `api/sw-mgmt.yml` defines the Security World Management API

2.2. KeySafe 5 Service Deployment

KeySafe 5 can be installed as a background service on a Unix or a Windows machine using the provided installers. See KeySafe 5 Service Deployment for details on configuring and installing the Service deployment.

- `keysafe5-service/keysafe5-server-1.5.0-Linux.tar.gz` is the Linux KeySafe 5 Service installer.
- `keysafe5-service/keysafe5-server-1.5.0-windows.msi` is the Windows KeySafe 5 Ser vice installer.

2.3. KeySafe 5 Kubernetes Deployment

KeySafe 5 can be installed to a Kubernetes cluster using the provided Helm Charts. See KeySafe 5 Kubernetes Deployment for details on configuring and installing the Kubernetes deployment.

2.3.1. Helm charts

The KeySafe 5 Kubernetes-based deployment consists of the following Helm charts:

- `keysafe5-k8s/helm-charts/nshield-keysafe5-backend-1.5.0.tgz`

  This installs the backend API services (Agent Management, CodeSafe Management, HSM Management and Security World Management).

- `keysafe5-k8s/helm-charts/nshield-keysafe5-ui-1.5.0.tgz`

  This installs the graphical user interface for KeySafe 5.

- `keysafe5-k8s/helm-charts/nshield-keysafe5-istio-1.5.0.tgz`

  This configures an existing Istio Ingress Gateway to allow external access (routing and authentication) to the services deployed by the previous two KeySafe 5 Helm charts.

- `keysafe5-k8s/helm-charts/bitnami-mongodb-17.0.0.tgz`

  This installs Bitnami packaged MongoDB database server for use by the KeySafe 5 back end services.

This organisation enables you to deploy the backend services only, if you do not need the UI, or the UI only, if you want to point it at some existing backend services already running elsewhere.

You can also use a different Kubernetes Ingress other than Istio if desired.

For more information on configuring and installing the Helm chart, see Hardening The Deployment .

2.3.2. Docker images

The Docker images are provided as tar archives. You can load them into a local Docker image registry using the `docker load` command, then push to a private container registry.

For example:

```
docker load < keysafe5-k8s/docker-images/hsm-mgmt.tar
Loaded image: hsm-mgmt:1.5.0
docker tag hsm-mgmt:1.5.0 private.registry.local/keysafe5/hsm-mgmt:1.5.0
docker login private.registry.local
docker push private.registry.local/keysafe5/hsm-mgmt:1.5.0
```

The Docker images provided are:

- `keysafe5-k8s/docker-images/agent-mgmt.tar` is the KeySafe 5 Agent Management ser vice
- `keysafe5-k8s/docker-images/codesafe-mgmt.tar` is the KeySafe 5 CodeSafe Manage-ment service
- `keysafe5-k8s/docker-images/hsm-mgmt.tar` is the KeySafe 5 HSM Management ser-vice
- `keysafe5-k8s/docker-images/sw-mgmt.tar` is the KeySafe 5 Security World Manage-

ment service

- `keysafe5-k8s/docker-images/ui.tar` is the KeySafe 5 user interface
- `keysafe5-k8s/docker-images/mongodb.tar` is the Bitnami packaged MongoDB database server container
- `keysafe5-k8s/docker-images/nginx.tar` is the Bitnami packaged NGINX container (used as an init container during Bitnami mongodb Helm Chart install)

These Docker images are intended to be deployed via the provided Helm charts. See the Helm chart configuration for details of how to configure and run each image.

2.4. KeySafe 5 agent installers

You can use the Linux and Windows KeySafe 5 agent installers provided to install the KeySafe 5 agent on nShield client machines. See Installation Steps for details on configuring and installing the agent.

- `keysafe5-agent/keysafe5-1.5.0-Linux-keysafe5-agent.tar.gz` is the Linux KeySafe 5 Agent installer.
- `keysafe5-agent/keysafe5-agent.msi` is the Windows KeySafe 5 Agent installer.

3. Getting Started

KeySafe 5 may either be installed as a background service on a Unix or a Windows machine using the KeySafe 5 Service Deployment or installed to a Kubernetes cluster using the provided Helm Charts as part of the KeySafe 5 Kubernetes Deployment.

- For installing the KeySafe 5 Service Deployment, see Service Deployment Installation Steps
- For a quick-start, non-production, installation of the KeySafe 5 Kubernetes Deployment, see Kubernetes Deployment Demo Deploy Script.
- For a production installation of the KeySafe 5 Kubernetes Deployment, see Kubernetes Deployment Manual Install Steps.

4. Security Guidance

Your nShield HSM protects the confidentiality and integrity of your Security World keys. KeySafe 5 allows an authorized client to remotely configure and manage an estate of nShield HSMs. All network traffic between KeySafe 5 and clients using the WebUI, the REST API, or both, passes through a secure channel. This TLS based secure channel is set up using token-based client authentication. The administrator of the KeySafe 5 system must remain diligent concerning the entities who are given access to the system and the secure configuration of the system.

Entrust recommends the following security-related actions for KeySafe 5 deployments:

- Ensure that log levels are set appropriately for your environment.

  More verbose log levels might expose information that is useful for auditing users of KeySafe 5, but the log information also reveals which REST API operations were performed. While this log information might be useful for diagnostics, it could also be considered sensitive and should be suitably protected when stored.

- Rotate the logs regularly. The log files could grow quickly if left unattended for a long time. The system administrator is responsible for log rotation.
- Verify the integrity of the KeySafe 5 tar file before executing it. You can verify the integrity of this file with the hash provided with the software download.
- Suitably protect the network environment of KeySafe 5 to maintain its availability, for example using firewalls and intrusion detection and prevention systems.
- Ensure that the KeySafe 5 platform's system clock is set accurately and only authorized system administrators can modify it so that the platform correctly interprets certificate and token lifetimes.
- Ensure that only authorized system administrators have access to the KeySafe 5 system, and only trusted software is run on the platform hosting KeySafe 5.
- Take standard virus prevention and detection measures on the platform hosting KeySafe 5.
- The system administrator should consider whether threats in the KeySafe 5 deployment environment would justify the encryption of the sensitive configuration data held in Kubernetes secrets, see Kubernetes documentation.

4.1. Customer security responsibilities

There are a number of third-party components that are required for correct KeySafe 5 oper ation, but which are not provided with KeySafe 5. These are considered the responsibility of the customer/operator.

It is the responsibility of the customer to:

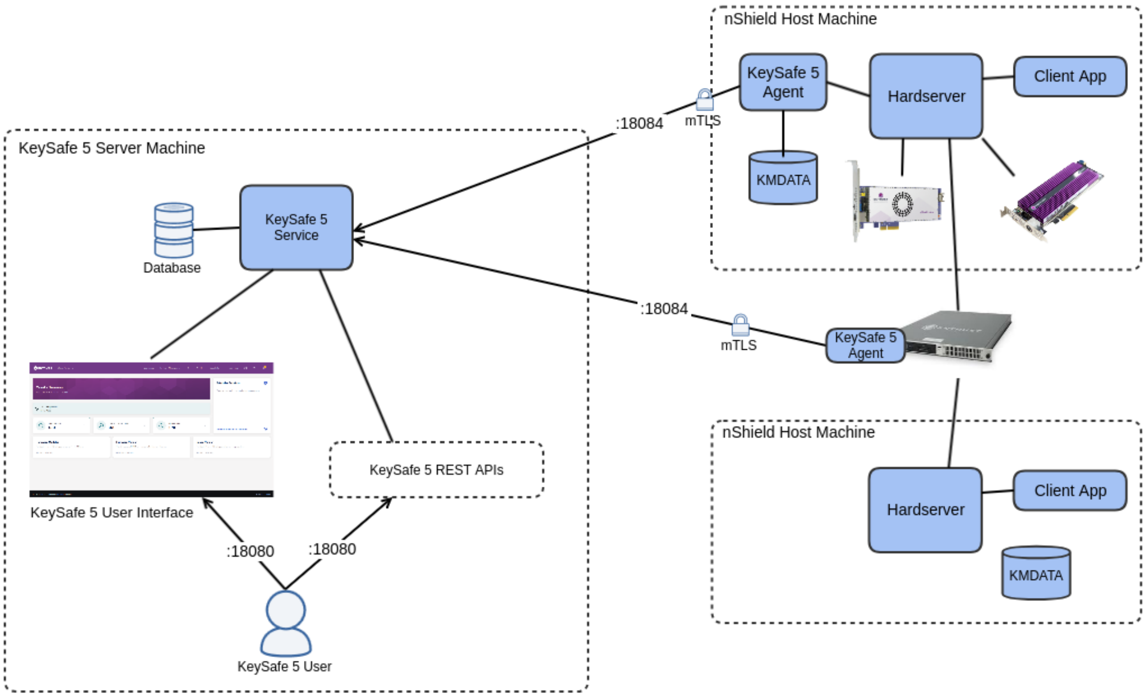
- Ensure that the Web Browser contains all the latest security updates from the Web Browser provider.
- Ensure that only authenticated users, that are trusted not to perform malicious actions, are given access to the KeySafe 5 system
- Ensure the integrity of any components that is downloaded from an external source. For example, by verifying the downloaded component using trusted 'hash fingerprints' or signatures.
- Ensure that a component is updated when an impacting CVE is published for the component.
- Ensure that all components are configured in a secure fashion and deployed in a secure environment.
- Ensure that all third-party components are configured in a secure fashion. For example, all third-party components should use mutually-authenticated secure channels to com municate with the KeySafe 5.
- Ensure that all certificates and keys, used for securing the communications between the third-party components and KeySafe 5, are uncompromised and of sufficient security strength.
- Ensure that the permissions required to access any sensitive configuration items are sufficient. For example, the permissions to access and manipulate a third-party compo nent's server certificates and their associated private key should only be provided to authorised and trusted administrators.
- Ensure that the external identity provider that is providing the bearer token used to authenticate the KeySafe 5 user implements a bearer token with a short lifetime. That is, the bearer token is reissued regularly, as this will mitigate the impact of a compromised bearer token, which would allowing unapproved access to KeySafe 5 for a prolonged period.
- Ensure that a threat analysis of the KeySafe 5 deployed environment has been performed, and that the results of this analysis justify any changes of KeySafe 5 default configuration.

5. KeySafe 5 Service Deployment

The KeySafe 5 Service Deployment installs KeySafe 5 (REST APIs and User Interface) as a service running on a Unix or a Windows machine.

KeySafe 5 Agent's may then be configured to connect to the KeySafe 5 server running on that machine, and optionally, you may configure external access to the REST APIs and WebUI via the KeySafe 5 Server configuration file.

Example KeySafe 5 Service deployment:



- To expose the KeySafe 5 WebUI and REST APIs externally from the KeySafe 5 server machine, you must configure the KeySafe 5 Server host and port, and configure any firewalls to allow inbound TCP traffic to the KeySafe 5 server on the configured port (18080 by default).
- To allow KeySafe 5 Agents installed on either a network-attached HSM or on a nShield host machine to communicate with the KeySafe 5 Service Deployment, you must allow inbound TCP traffic to the KeySafe 5 server on the configured agent communications port (18084 by default).
- KeySafe 5 Agents are required on nShield host machine's if you want to use KeySafe 5 to manage a HSM or Security World that is local to that machine.

5.1. Prerequisites

Ensure the following prerequisites are met before installing KeySafe 5 Service.

5.1.1. Software Prerequisites

- A supported Security World installation. Please see the release notes for details.

5.1.2. Network Prerequisites

- The following network configuration:
  - Inbound TCP port 18084 must be open for KeySafe 5 Agent communication.
  - Inbound TCP port 18080 must be open if you want to access the WebUI/API exter nally.

    Entrust recommends enabling authentication before exposing the WebUI/API externally.

5.2. Installation Steps

The following steps will install KeySafe 5 Service.

KeySafe 5 Service is shipped with a working configuration out of the box and will detect if existing certificates are in place, if no certificates are found then the installer will create self-signed ones with a 30-day validity. To regenerate self-signed certificates with a different validity, see keysafe5-server-admin Utility.

After installation KeySafe 5 Service will be running and available at `https://127.0.0.1:18080`.

Before installation ensure that all the prerequisites outlined in KeySafe 5 Service Prerequisites are met for the intended deployment.

By default, this KeySafe 5 central platform deployment will only be able to communicate with version 1.5 or later KeySafe 5 Agents. If you want your deployment to be able to communicate with legacy (1.4 or earlier) KeySafe 5 Agents then you must set `agent_comms.compatibilityMode` to `true` in KeySafe 5 Service Configuration.

5.2.1. Linux

1. Extract the KeySafe 5 Service package to the root of the filesystem. The install pack-

age can be found in the `keysafe5-service` directory of the KeySafe 5 release package.

This unpacks the KeySafe 5 Service binaries and associated scripts into the `/opt/nfast/` directory.

```
sudo tar -C / -xf /path/to/keysafe5-server-1.5.0-Linux.tar.gz
```

2. Configure the KeySafe 5 Service as described in KeySafe 5 Service Configuration and KeySafe 5 Service Certificates.

   This step is optional, KeySafe 5 Service is shipped with a working configuration out of the box.

3. Run the KeySafe 5 Service install script:

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

5.2.1.1. Managing The Service

To stop, start, or restart the KeySafe 5 service on Linux, use `/opt/nfast/scripts/init.d/keysafe5-server`. For example:

```
sudo /opt/nfast/scripts/init.d/keysafe5-server restart
```

5.2.2. Windows

1. Double-click on `keysafe5-server-1.5.0-windows.msi`. The install package can be found in the `keysafe5-service` directory of the KeySafe 5 release package.
2. Configure the KeySafe 5 Service as described in KeySafe 5 Service Configuration and KeySafe 5 Service Certificates.

   This step is optional, KeySafe 5 Service is shipped with a working configuration out of the box.

5.2.2.1. Managing The Service

To stop, start, or restart the KeySafe 5 service on Windows, use the standard Windows Services facility.

5.3. Upgrade Steps

This chapter details how to update an existing KeySafe 5 install to the latest version.

KeySafe 5 Service Deployment was introduced in KeySafe 5 v1.5.0. You may upgrade a 1.4 KeySafe 5 Local installation to the KeySafe 5 Service Deployment and keep the existing database contents.

5.3.1. Upgrade from 1.4 KeySafe 5 Local

> Upgrading from KeySafe 5 Local will allow you to keep the existing data base contents but does not migrate the service configuration. You will need to manually configure KeySafe 5 Service Deployment as detailed in Configuration Items and then update KeySafe 5 Agents.

1. Stop any running KeySafe 5 Local instance on the machine.
2. Optionally, backup `%NFAST_DATA_HOME%\databases` directory.
3. Uninstall KeySafe 5 Local as per the KeySafe 5 v1.4 Installation Guide.
4. Install KeySafe 5 Service Deployment. The install package can be found in the `keysafe5-service` directory of the KeySafe 5 release package.

   On Linux:

   a. Run `sudo tar -C / -xf /path/to/keysafe5-server-1.5.0-Linux.tar.gz` to unpack the KeySafe 5 Service binaries and associated scripts into the `/opt/nfast/` directory.
   b. Run `sudo /opt/nfast/bin/keysafe5-server-admin init` to initialise new certificates to use for KeySafe 5 to use for communicating with the Agent Communications Interface. See Certificate Details.
   c. Run `sudo /opt/nfast/keysafe5/server/sbin/install` to install KeySafe 5 Server.

   On Windows:

   a. Double-click on keysafe5-server-1.5.0-windows.msi and follow the Installation Wiz ard.
   b. From an Administrator command prompt, run `%NFAST_HOME%\bin\keysafe5-server-admin init -y` to initialise new certificates to use for KeySafe 5 to use for communicating with the Agent Communications Interface. See Certificate Details.
   c. Restart KeySafe 5 Service using the standard Windows Services facility.
5. Open the KeySafe 5 WebUI (by default at `https://127.0.0.1:18080`) and check you can see any existing KeySafe 5 data. Resources may be marked as unhealthy until the corresponding KeySafe 5 Agent's are upgraded and configured to communicate with

the updated Agent Communications Interface.

6. Upgrade and generate new Agent communication certificates for each KeySafe 5 Agent. To upgrade KeySafe 5 Agents, see Agent Upgrade.

5.4. Configuration Items

The KeySafe 5 Service configuration file is located at `%NFAST_DATA_HOME%/keysafe5/server/config/config.yaml`.

The install contains an example configuration file at `%NFAST_DATA_HOME%/keysafe5/server/config/config.yaml.example` which can be used to revert back to original configuration if needed.

Unless configured otherwise, `%NFAST_DATA_HOME%` is located at `/opt/nfast` on Linux and `%ProgramData%\nCipher` on Windows.

Time durations are a sequence of decimal numbers, each with optional fraction and a unit suffix, such as "300ms", "1.5h" or "2h45m". Valid time units are "ns", "us" (or "µs"), "ms", "s", "m", "h". For example, `30s` configures a time interval of `30 seconds`.

Configuration Key	Description	Example Value
`server.host`	Host used for serving the WebUI and API. Entrust recommends keeping this value as `127.0.0.1`, to restrict external connections, until authentication has been configured.	`127.0.0.1`
`server.port`	Port used for serving the WebUI and API. If this port is not available, KeySafe 5 will fail to start.	`18080`
`server.read_timeout`	Period of time before timing out reading a request.	`5m`
`server.write_timeout`	Period of time before timing out writing a response. This should be at least as long as you'd expect the slowest nShield request in your environment to take (e.g. the amount of time to write a card when creating a Security World)	`8m`
`server.cleanup_timeout`	Amount of time to wait after each request for the next request before timing out.	`30s`
`server.max_header_bytes`	Maximum number of bytes to read while parsing the request header's keys and values	`1048576`
`server.tls.min_protocol_version`	Minimum TLS protocol version allowed. Valid values: `TLSV1_0`, `TLSV1_1`, `TLSV1_2`, `TLSV1_3`.	`TLSV1_2`

Configuration Key	Description	Example Value
`server.tls.cipher_suites`	Allowed cipher suites. The default provided here is the list of recommended cipher suites. TLSv1.3 cipher suites are currently not configurable. See Supported TLS Cipher Suites.	`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`
`ui.refresh_rate`	How often the WebUI will poll the backend. Set 0 to disable auto refresh in the WebUI.	`30s`
`agent_comms.host`	Host used for communication with KeySafe 5 Agents.	`0.0.0.0`
`agent_comms.port`	Port used for communication with KeySafe 5 Agents. If this port is not available, KeySafe 5 will fail to start.	`18084`
`agent_comms.compatibilityMode`	Enable message bus server compatibility mode. If false, this KeySafe 5 Server will only be able to commu nicate with KeySafe 5 v1.5, or newer, Agents	`false`
`agent_comms.tls.min_protocol_version`	Minimum TLS protocol version allowed. Valid values: `TLSV1_0`, `TLSV1_1`, `TLSV1_2`, `TLSV1_3`.	`TLSV1_2`
`agent_comms.tls.cipher_suites`	Allowed cipher suites. The default provided here is the list of recommended cipher suites. TLSv1.3 cipher suites are currently not configurable. See Supported TLS Cipher Suites.	`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`
`agent_comms.tls.ocsp.enabled`	Enable OCSP checks	`true`

Configuration Key	Description	Example Value
`agent_comms.tls.ocsp.`	OCSP Stapling Mode - [auto, always, never]. `auto` staples a status, only if 'status_request' is set in the certificate. `always` enforces OCSP stapling for certificates even if 'status_request' is not set in the certificate. `never` disables OCSP stapling even if the certificate has Must-Staple flag	`auto`
`agent_comms.tls.ocsp.override_url`	HTTP URL used to get OCSP staples. Overrides the OCSP Responder URI set in certificates. Example: https://1.2.3.4:5000	``
`agent_comms.tls.ocsp.cache_enabled`	Cache OCSP staples to local file storage.	`true`
`auth.type`	Authentication type applied to the WebUI/API interface. Valid values: `none`, `oauth_oidc`. Entrust recommends configuring this section before entering production.	`none`
`auth.oauth_oidc.issuers`	Listing of OIDC/OAuth2 issuers configured, each of the following items are per issuer. Please refer to your Identity Provider's documentation for details, these items are usually returned from its .well-known/openid-configuration endpoint.	``
`auth.oauth_oidc.issuers.name`	Name for the issuer to be displayed in the WebUI.	`Entrust IDaaS`
`auth.oauth_oidc.issuers.issuer`	Identity of the issuer This MUST match the 'iss' payload in any issued JWT by the issuer	`https://example.idp.com`
`auth.oauth_oidc.issuers.jwks_uri`	URL of the issuers public key set to validate signature of the JWT. Can only set one of `jwk_url` or `offline_jwks`.	`https://example.idp.com/jwks`
`auth.oauth_oidc.issuers.offline_jwks`	JWKs of public keys to validate signature of the JWT Can only set one of `jwk_url` or `offline_jwks`.	`'{"keys":[…]}'`
`auth.oauth_oidc.issuers.jwks_cache_refresh`	Period of time that the JWKs will be refreshed Will be the largest of either the Cache-Control response header, the Expires header or this value. Not used if `offline_jwks` is set.	`15m`
`auth.oauth_oidc.issuers.audiences`	List of JWT audiences that are allowed access. A JWT containing any of these audiences will be accepted.	`https://example.audience.com`
`auth.oauth_oidc.issuers.client_id`	ID of the application to request a JWT for.	`33118f7c-2be5-40eb-bf45-60ba091596e3`
`auth.oauth_oidc.issuers.response_type`	Which grant type to execute during authentication.	`code`

Configuration Key	Description	Example Value
`auth.oauth_oidc.issuers.scope`	List of scopes to request.	`profile`, `openid`, `offline_access`
`auth.oauth_oidc.issuers.logout_redirect_uri`	URL that the issuer will redirect to on successful logout.	https://keysafe5.server.com
`auth.oauth_oidc.issuers.authorization_endpoint`	URL of the issuer to request authentication.	https://example.idp.com/authorize
`auth.oauth_oidc.issuers.token_endpoint`	URL of the issuer to obtain a token.	https://example.idp.com/token
`auth.oauth_oidc.issuers.userinfo_endpoint`	URL of the issuer to obtain user information.	https://example.idp.com/userinfo
`auth.oauth_oidc.issuers.end_session_endpoint`	URL of the issuer to end the session.	https://example.idp.com/endsession
`logging.level`	Minimum severity level of log statements to output. Valid values: `trace`, `debug`, `info`, `warning`, `error`. The default is to output at `info` level and above.	`info`
`logging.format`	Format of the log statements. Valid values: `json`, `logfmt`. The default is to output in `json` format.	`json`
`logging.file.enabled`	To enable log output to file, set to `true`. The default is to output to file (`true`).	`true`
`logging.file.path`	The absolute path of the directory to which logs should be written. The default is `/opt/nfast/log` on Linux and `%ProgramData%\nCipher\Log Files` on Windows.	`/opt/nfast/log`
`database.type`	Type of database to use for KeySafe 5. Valid values: [sqlite]	`sqlite`
`database.timeout`	Timeout for database requests.	`30s`
`database.sqlite.database_directory`	Absolute path of the directory in which KeySafe 5 will store its database files. KeySafe 5 must have permission to read and write to this directory. If not specified, it defaults to $NFAST_KMDATA/databases	`/opt/nfast/kmdata/databases`
`health.update_period`	Period of time between health checks.	`30s`
`health.timeout_period`	Time before a running health check should fail.	`10s`
`health.liveness_failure_period`	Period of time before a liveness check is marked as fail ing.	`5m`

Configuration Key	Description	Example Value
`health.allowed_clock_skew`	Maximum amount of time a clock on a KeySafe 5 agent can differ from this service before the host clockSkew health check fails.	`2m`
`filestore`	Absolute path of the directory in which KeySafe 5 will store large files. These may be gigabytes in size. KeySafe 5 must have permission to read and write to this directory. If not specified, it defaults to $NFAST_KEYSAFE5/server/filestore	`%NFAST_DATA_HOME%/keysafe5/server/filestore`

5.5. Certificate Details

In KeySafe 5 Service, certificates are used to secure communications. The following sections provide details about the different types of certificates used in KeySafe 5.

5.5.1. WebUI/API Interface Certificates

Certificates are used to secure the API and WebUI interface. These certificates are found in the `%NFAST_DATA_HOME%/keysafe5/server/https` directory.

Unless configured otherwise, `%NFAST_DATA_HOME%` is located at `/opt/nfast` on Linux and `%ProgramData%\nCipher` on Windows.

File Name	Description
server.crt	The TLS certificate for the WebUI/API interface.
server.key	The private key for the WebUI/API interface.

On initial installation, a self-signed certificate is created. It is recommended to replace this with a certificate from a trusted Certificate Authority (CA). To replace the HTTPS certificate, follow these steps:

1. Optional - Replace the private key located at `%NFAST_DATA_HOME%/keysafe5/server/https/server.key` with one that meets your requirements.
2. Generate a Certificate Signing Request (CSR) using the private key located at `%NFAST_DATA_HOME%/keysafe5/server/https/server.key`.
3. Submit the CSR to a trusted CA to obtain a signed certificate.
4. Replace the existing `server.crt` file with the new signed certificate.

5. Ensure the new certificate is in PEM format and includes the full certificate chain if required by your CA.
6. Restart the KeySafe 5 service to apply the changes.
7. Verify the WebUI/API interface is functioning correctly with the new certificate.

5.5.2. Agent Communication Certificates

TLS certificates are used to secure communication between the KeySafe 5 Service and KeySafe 5 Agents. These certificates are found in multiple directories under the `%NFAST_-DATA_HOME%/keysafe5/server/tls` directory.

Unless configured otherwise, `%NFAST_DATA_HOME%` is located at `/opt/nfast` on Linux and `%ProgramData%\nCipher` on Windows.

5.5.2.1. Server Certificates

The server certificates are used by KeySafe 5 Service to secure the communication to KeySafe 5 Agents. The CA certificate is used to verify that connecting KeySafe 5 Agents have permission to connect. These certificates are found in the `%NFAST_-DATA_HOME%/keysafe5/server/tls/server` directory.

File Name	Description
server.crt	The TLS certificate for the KeySafe 5 Agent communication interface.
server.key	The private key for the KeySafe 5 Agent communication interface.
ca.crt	The CA certificate used to sign KeySafe 5 Agent certificates. KeySafe 5 Agents not signed by this CA will not be able to connect to the KeySafe 5 Service.

On initial installation, a self-signed CA and server certificate are created. It is recommended to replace these with certificates from a trusted Certificate Authority (CA). To replace the TLS certificates, follow these steps:

1. Optional - Replace the private key located at `%NFAST_-DATA_HOME%/keysafe5/server/tls/server/server.key` with one that meets your requirements.
2. Generate a Certificate Signing Request (CSR) using the private key located at `%NFAST_-DATA_HOME%/keysafe5/server/tls/server/server.key`.
3. Submit the CSR to a trusted CA to obtain a signed certificate.
4. Replace the existing `server.crt` file with the new signed certificate.

5. Obtain the CA certificate from your trusted CA and replace the existing `ca.crt` file with it. This is used to ensure that only KeySafe 5 Agents signed by this CA can connect to the KeySafe 5 Service.
6. Ensure the new certificates are in PEM format and include the full certificate chain if required by your CA.
7. Restart the KeySafe 5 service to apply the changes.
8. Any KeySafe 5 Agents that were connected to the KeySafe 5 Service will need to be updated with new certificates signed by the new CA.

5.5.2.2. Client Certificates

Client certificates are used by KeySafe 5 Server internally to authenticate the connection to the Agent Communication interface. These certificates are found in the `%NFAST_DATA_HOME%/keysafe5/server/tls/clients` directory.

The TLS certificate that KeySafe 5 uses for connection to the Agent Communications interface must contain `keysafe5-backend-services` in the certificate's Distinguished Name so that the Agent Communications interface can properly limit permissions for this certificate. If the certificate's DistinguishedName does not contain `keysafe5-backend-services` then KeySafe 5 Service will be unable to connect to the Agent Communication interface.

File Name	Description
tls.crt	The TLS certificate for the KeySafe 5 Service to authenticate to the Agent Communication interface. This certificate is signed by the CA used to sign KeySafe 5 Agent certificates.
tls.key	The private key for the KeySafe 5 Service to authenticate to the Agent Communication interface.
ca.crt	The CA certificate used to sign the Agent Communication Server certificates.

On initial installation, a self-signed certificate is created. If you are replacing the Agent Com munication Server certificates and/or the CA used to sign the KeySafe 5 Agent certificates, it is necessary to also replace these with certificates signed by the same CA.

5.5.2.3. Certificate Authority (CA)

The CA certificate is used to sign KeySafe 5 Agent certificates. This certificate is found in the `%NFAST_DATA_HOME%/keysafe5/server/tls/ca` directory.

This directory is created during initial installation if existing certificates are not detected. It is created and managed by the `keysafe5-server-admin` utility. If you wish to use your own CA, you can remove this directory.

File Name	Description
ca.key	The private key for the CA used to sign KeySafe 5 Agent certificates.
ca.crt	The CA certificate used to sign KeySafe 5 Agent certificates.

Please see the `keysafe5-server-admin` utility section below for details on how to reinitialize this CA and sign KeySafe 5 Agent certificates.

5.5.2.4. keysafe5-server-admin Utility

The `keysafe5-server-admin` tool is installed with KeySafe 5 Service to aid management of the certificate infrastructure used to secure communication between the KeySafe 5 Service and KeySafe 5 agents.

`keysafe5-server-admin` is invoked on initial install if existing certificates are not detected to ensure a working instance post installation.

`keysafe5-server-admin` can be used to initialise a self-signed CA, update CA and server certificates, and sign KeySafe 5 Agent CSR requests.

By default, generated certificates are valid for 30 days.

If you use `keysafe5-server-admin` to update any certificates, you must then restart KeySafe 5 Server to apply the updated certificates.

Updating the CA certificate will require any KeySafe 5 Agent certificates signed by the old CA certificate to be re-signed with the new CA certificate.

5.5.2.4.1. Initialise TLS certificates for KeySafe 5

```
keysafe5-server-admin init [-y|-n] [--ca DAYS] [--server DAYS] [ADDRESSES...]
```

- `-y` : Always overwrite an existing configuration.
- `-n` : Always preserve the existing configuration.
- `--ca DAYS` : (Optional) The number of days that the CA certificate will be valid for

(default 30).

- `--server DAYS` : (Optional) The number of days that the server certificates will be valid for (default 30).
- `[ADDRESSES]` : (Optional) Comma-separated list of IP addresses to include in the generated server certificate. If not provided, all local and loopback IPv4 and IPv6 addresses are added automatically.

Example:

```
keysafe5-server-admin init 127.0.0.1,172.26.0.1,192.168.0.1,::1
```

5.5.2.4.2. Update the CA certificate

Update the Certificate Authority certificate. The certificate will be valid for 30 days unless specified otherwise.

```
keysafe5-server-admin ca [DAYS]
```

- `[DAYS]` : (Optional) The number of days that the signed certificate will be valid for (default 30).

Example:

```
keysafe5-server-admin ca 365
```

5.5.2.4.3. Update the server certificates

Update the server TLS certificate for the KeySafe 5 Agent communication interface and the client TLS certificate for the KeySafe 5 Service to authenticate to the Agent Communication interface. The certificates will be valid for 30 days unless specified otherwise.

```
keysafe5-server-admin server [DAYS] [ADDRESSES...]
```

- `[DAYS]` : (Optional) The number of days that the server certificates will be valid for (default 30).
- `[ADDRESSES]` : (Optional) Comma-separated list of IP addresses to include in the generated server certificate. If not provided, all local and loopback IPv4 and IPv6 addresses are added automatically.

Example:

```
keysafe5-server-admin server 60 127.0.0.1,172.26.0.1,192.168.0.1,::1
```

5.5.2.4.4. Sign a CSR for a KeySafe 5 Agent.

```
keysafe5-server-admin sign PATH_TO_CSR [DAYS] [TLS_DIRECTORY]
```

- `PATH_TO_CSR` : Path to the Certificate Signing Request file.
- `[DAYS]` : (Optional) The number of days that the signed certificate will be valid for (default 30).
- `[TLS_DIRECTORY]` : (Optional) Directory to save the signed certificates.

Example:

```
keysafe5-server-admin sign demo.csr 365
```

5.6. Backup Details

Entrust recommends that you back up the following directories as part of your routine nShield backup schedules.

Unless configured otherwise, `%NFAST_DATA_HOME%` is located at `/opt/nfast` on Linux and `%ProgramData%\nCipher` on Windows.

Directory	Contents
`%NFAST_DATA_HOME%/kmdata/databases`	KeySafe 5 Server Database files on Linux.
`%NFAST_DATA_HOME%/Key Management Data/databases`	KeySafe 5 Server Database files on Windows.
`%NFAST_DATA_HOME%/keysafe5/server/config`	KeySafe 5 Server configuration files.
`%NFAST_DATA_HOME%/keysafe5/server/https`	HTTPS certificates and key files which secure the WebUI and API interface.
`%NFAST_DATA_HOME%/keysafe5/server/filestore`	Large files uploaded to and used in KeySafe 5. Linked to items in databases.
`%NFAST_DATA_HOME%/keysafe5/server/tls`	TLS certificates and key files which secure the KeySafe 5 agent communication interface.

5.7. Troubleshooting

If the KeySafe 5 Service fails to start, ensure that all KeySafe 5 Service Prerequisites are met and that install is correctly configured, configuration information can be found here.

If KeySafe 5 Service still fails to start, see below for instructions on accessing the logs.

5.7.1. Logs

5.7.1.1. Linux

The KeySafe 5 Service log files are located in the `/opt/nfast/log` directory, unless configured otherwise in the configuration file, and their filenames are prefixed with `keysafe5-server`.

5.7.1.2. Windows

The KeySafe 5 Service log files are located in the `%NFAST_LOGDIR%` directory, unless configured otherwise in the configuration file, and their filenames are prefixed with `keysafe5-server`.

The KeySafe 5 Service actions are emitted to the Windows event log under the `nShield-KeySafe5Service` source identifier.

You can use the `nshieldeventlog` utility, which is shipped with the nShield Security World Software, to extract these log entries and output them to the console or a text file.

```
nshieldeventlog.exe --source=nShieldKeySafe5Service
```

As required, specify the following parameters.

- `-c` | `--count`: The number of records read from the event log.

  The default is `10000`

- `-f` | `--file`: The output filename.

See the nShield Security World Software documentation for more information on the `nshieldeventlog` utility.

5.8. Uninstall Steps

Before uninstalling the KeySafe 5 Service, Entrust recommends that you back up any configuration files and certificates from the installation, for more information see KeySafe 5 Service Backup.

5.8.1. Linux

To remove the KeySafe 5 Service from a Linux host run the KeySafe 5 uninstaller:

```
sudo /opt/nfast/keysafe5/server/sbin/install -u
```

Then proceed to remove the following files and directories:

- `/opt/nfast/keysafe5/server`
- `/opt/nfast/sbin/keysafe5-server`
- `/opt/nfast/bin/keysafe5-server-admin`
- `/opt/nfast/log/keysafe5-server*`
- `/opt/nfast/lib/versions/keysafe5-server-atv.txt`
- `/opt/nfast/scripts/install.d/14keysafe5-server`
- `/opt/nfast/kmdata/databases/agent-mgmt.sqlite`
- `/opt/nfast/kmdata/databases/codesafe-mgmt.sqlite`
- `/opt/nfast/kmdata/databases/hsm-mgmt.sqlite`
- `/opt/nfast/kmdata/databases/sw-mgmt.sqlite`

The log files will be located in a different location if you have changed the default value of `logging.file.path` in the configuration file.

If required, you can also remove the `keysafe5serviced` user that was created as part of the installation.

5.8.2. Windows

To remove the KeySafe 5 Service from a Windows host:

1. Stop KeySafe 5 Service using **Windows Service Manager**.
2. Open the **Control Panel** and select **Programs and Features**.
3. Select the **nShield KeySafe 5 Service** package.
4. Select **Uninstall** and follow the on-screen instructions.

Then proceed to remove the following files and directories:

- `%NFAST_DATA_HOME%\keysafe5\service`
- `%NFAST_DATA_HOME%\Log Files\keysafe5-server*`
- `%NFAST_DATA_HOME%\Key Management Data\databases\agent-mgmt.sqlite`
- `%NFAST_DATA_HOME%\Key Management Data\databases\codesafe-mgmt.sqlite`
- `%NFAST_DATA_HOME%\Key Management Data\databases\hsm-mgmt.sqlite`
- `%NFAST_DATA_HOME%\Key Management Data\databases\sw-mgmt.sqlite`

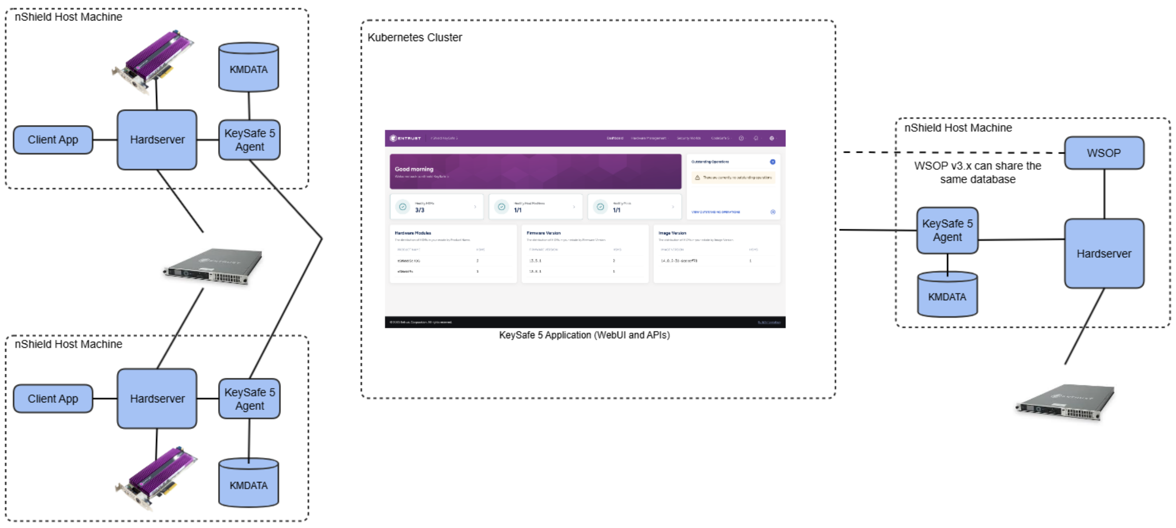
The log files will be located in a different location if you have changed the default value of `logging.file.path` in the configuration file.

6. KeySafe 5 Kubernetes Deployment

The KeySafe 5 Kubernetes Deployment installs KeySafe 5 (REST APIs and User Interface) as an application running in a Kubernetes cluster.

KeySafe 5 Agent's may then be configured to connect to the KeySafe 5 server.

Example KeySafe 5 Kubernetes deployment:



6.1. Prerequisites

The following table contains the required software version that this release of KeySafe 5 has been tested on and any minimum version requirements.

Software	Minimum version	Tested version
Kubernetes	1.31	1.33
MongoDB	7.0.14	8.0.13

In addition to the set of required software, this release of KeySafe 5 requires:

- A location for storing large objects (for example CodeSafe machines)
- An external identity provider that supports OIDC and OAuth2 for user and machine authentication.

6.1.1. Optional Software

KeySafe 5 is shipped with a Helm chart that configures an Istio Ingress Gateway to provide external access to the KeySafe 5 application running in Kubernetes. Other Ingress Gateways can be used, see Configure a custom ingress provider.

Software	Minimum version	Tested version
Istio	1.26	1.26

6.1.2. Hardware Requirements

Entrust recommends the following hardware specification to ensure smooth running of KeySafe 5.

- CPU: 4 minimum (8 recommended)
- RAM: 8GB minimum
- Disk Storage: 64GB minimum
  - For optimal performance Entrust recommends use of an SSD.

Requirements will vary depending on the size of the nShield estate being managed and if services, such as MongoDB, are being hosted on the same machine as the Kubernetes cluster or externally.

6.1.3. Kubernetes cluster

KeySafe 5 has been tested on Kubernetes version 1.33.

6.1.3.1. Using namespaces

When deploying an application to a Kubernetes cluster that is shared with many users spread across multiple teams, Entrust recommends using Namespaces to isolate groups of resources.

To create a namespace for the KeySafe 5 application:

```
kubectl create namespace nshieldkeysafe5
namespace/nshieldkeysafe5 created
```

To set the namespace for a current request, use the `--namespace` flag. For example:

```
helm install --namespace=nshieldkeysafe5 my-release helm-keysafe5-backend/
kubectl --namespace=nshieldkeysafe5 get pods
```

If you are using Istio in your Kubernetes cluster, beyond acting as an API Gateway for

KeySafe 5, you might also want to configure Istio injection for this Kubernetes namespace to take advantage of other Istio features. This step is not required for KeySafe 5 to function.

```
kubectl label namespace nshieldkeysafe5 istio-injection=enabled
namespace/nshieldkeysafe5 labeled
```

6.1.4. MongoDB

MongoDB is the persistent data store for the KeySafe 5 application data. Any sensitive Security World data stored in the database is stored in standard nShield encrypted blobs. You should restrict access to this database in the same way that you would normally restrict access to the `Key Management Data` directory on an nShield client machine.

> The MongoDB used must be a Replica Set.

If you have an existing MongoDB database you can configure the application to use this, otherwise you must securely deploy a MongoDB instance.

> Entrust recommend that you configure MongoDB with authentication enabled and TLS enabled and with RBAC configured. The database user for KeySafe 5 should be given the minimum capabilities required, see Database: User Roles.

6.1.5. Large Object Storage

A location for storing objects that are too large for a traditional database, such as CodeSafe machines, is required. This storage can be located either within the Kubernetes cluster, or externally.

To configure this storage you must specify a Persistent Volume Claim (PVC) for the `helm-keysafe5-backend` Helm Chart to use via the `objectStore.pvc` Chart parameter. If a Kubernetes namespace has been created for the KeySafe 5 application, then the PVC must be in the same namespace as the application. The PVC may use any type of storage supported by Persistent Volumes in your Kubernetes Cluster (for example, NFS). See Persistent Volumes for supported storage types.

To set the user and group IDs used by the KeySafe 5 application when accessing the object storage, configure the `podSecurityContext.runAsUser`, `podSecurityContext.runAsGroup` and `podSecurityContext.fsGroup` Chart parameters.

You should ensure that the size of the configured storage is sufficient to meet the application's needs. For this release of KeySafe 5 that should include:

- All CodeSafe 5 machine images that will be managed by KeySafe 5.
- All HSM upgrade images that will be managed by KeySafe 5.

6.1.5.1. NFS Object Storage Configuration

To use an NFS for object storage you must know the NFS server address and the path of the directory being exported from the NFS server.

Create a Persistent Volume containing the configuration of your NFS.

```
cat << EOF | kubectl -n nshieldkeysafe5 apply -f -
  apiVersion: v1
  kind: PersistentVolume
  metadata:
    name: nfs-pv
    labels:
      application: keysafe5
  spec:
    capacity:
      storage: 2Gi
    volumeMode: Filesystem
    accessModes:
      - ReadWriteMany
    persistentVolumeReclaimPolicy: Recycle
    storageClassName: nfs
    nfs:
      path: ${NFS_PATH}
      server: ${NFS_IP}
EOF
```

Create a Persistent Volume Claim to use that Persistent Volume.

```
cat << EOF | kubectl -n nshieldkeysafe5 apply -f -
  apiVersion: v1
  kind: PersistentVolumeClaim
  metadata:
    name: data-nshield-keysafe5
  spec:
    storageClassName: nfs
    accessModes:
      - ReadWriteMany
    resources:
      requests:
        storage: 2Gi
    selector:
      matchLabels:
        application: keysafe5
EOF
```

6.1.6. External identity provider (IdP)

You need an external identity provider that supports OIDC and OAuth2 to provide user and machine authentication to KeySafe 5. This is required to gain access to the WebUI and authenticate commands sent to the backend services.

At the IdP, Entrust typically expects the following to be configured:

- A single OIDC public client application

  This provides user identity information and an `id_token` for use by the WebUI.

- Multiple OAuth2 private client application

  This provides machine-to-machine credentials. Typically you would want an instance of this per application identity required to limit the sharing of the `client_secret` value.

For more information, refer to Client Types.

6.1.6.1. OIDC public client

The OIDC public client application provides user identity information and an `id_token` for use by the WebUI in its calls to the backend services. It is a public client due to the WebUI being a client side application, and as such cannot be trusted with the `client_secret` like a server side application would be.

Entrust recommends the following settings:

Setting	Value
Grant Types	Authorization Code with PKCE
	Refresh Token
Authorization Code PKCE Code Challenge Method	S256
Scopes	openid

For more information, refer to Authorization Code flow with PKCE.

6.1.6.2. OAuth2 private client

The OAUTH2 private client provides client credentialing for machine-to-machine authentication by applications that do not hold user identification. This requires the use of the `client_secret` value, which must be securely held.

You would typically want a separate private client instance for each application for which you provide access, resulting in a separate `client_id` and `client_secret` for each application. This eases management of the `client_secret` by reducing the number of applications that have knowledge of it. It also provides easy identification of which application is doing what at the KeySafe 5 end.

Entrust recommend the following settings:

Setting	Value
Grant Type	Client Credentials

For more information, refer to Client Credentials.

6.2. Deploy Script

The included deploy script (`keysafe5-k8s/deploy.sh`) provides a quick start installer for installing a Kubernetes deployment of KeySafe 5 on a Linux machine.

6.2.1. Overview and prerequisites

The following steps provide a quick-start guide to installing KeySafe 5 and its dependencies using the provided deploy script for evaluation purposes. Please refer to Manual Install Steps for full installation instructions.

The script is designed to be run on UNIX/Linux based systems by a non-root user. The script may call `sudo` as required.

These steps install KeySafe 5 and its dependencies. The included deploy script (`deploy.sh`) will provide a substitute installation for the various dependencies using self-signed temporary certificates. They are only suitable for evaluation purposes, and should *not* be used for production environments.

Please see Hardening The Deployment for steps to harden the deployment. Entrust recommends these steps as a minimum and that additional hardening may be required dependent on your own requirements.

A production deployment will have as a minimum the following:

- Maintained and patched versions of all the dependencies
- A secure CA with TLS v1.3 support for certificates. The deploy script can provide a local insecure CA.
- A secure Kubernetes installation. The deploy script can install K3s locally.

- A secure MongoDB database. The deploy script can provide a replicated MongoDB with X.509 authentication running in Kubernetes.
- A secure means of large object storage. The deploy script can provide local object storage within the Kubernetes cluster or be config ured for using an NFS for object storage.
- HTTPS secured by a trusted certificate for the KeySafe 5 endpoints. The deploy script will enable HTTPS connections with a self-signed insecure certificate.
- Require authentication to access KeySafe 5. OIDC & OAUTH2 are currently supported in KeySafe 5. The deploy script will not set up authenticated access.
- Time synchronization between the central platform and all agents.

The script requires a local installation of Docker or Podman. When using `podman` on Red Hat Enterprise Linux, you should install the `podman-docker` package to provide the Docker alias.

The user executing the deploy script must be able to successfully execute `docker info`. If this is not the case, please consult the appropriate documentation for your platform, Docker Documentation or Podman Documentation.

This release includes Docker images that need to be pushed to a Docker registry. If you have a private registry you may push the images from a different machine.

6.2.2. Hardware Requirements

See Hardware Requirements.

6.2.3. Unpack the release

```
mkdir keysafe5-1.5.0
tar -xf nshield-keysafe5-1.5.0.tar.gz -C keysafe5-1.5.0
cd keysafe5-1.5.0/keysafe5-k8s
```

The user executing the `deploy.sh` script must have permission to read and write files within the `keysafe5-1.5.0` directory. This will automatically be the case if the user extracting the release package is the same user that executes the deploy script.

If there is an existing infrastructure that you may like to use when installing KeySafe 5 via the deployment script, continue with Existing infrastructure. Otherwise skip to Authentication and proceed from there.

6.2.4. Existing infrastructure

6.2.4.1. Docker images

If you have a private registry you may push the images to it like so:

```
# Load the Docker images to your local Docker
docker load < docker-images/agent-mgmt.tar
docker load < docker-images/codesafe-mgmt.tar
docker load < docker-images/hsm-mgmt.tar
docker load < docker-images/mongodb.tar
docker load < docker-images/nginx.tar
docker load < docker-images/sw-mgmt.tar
docker load < docker-images/ui.tar
```

There is no need to load the mongodb or nginx images if an external MongoDB instance is being used.

```
# Define the private registry location
export DOCKER_REGISTRY=private.registry.local/my_space/keysafe5

# Tag the Docker images for a private registry
docker tag agent-mgmt:1.5.0 $DOCKER_REGISTRY/keysafe5/agent-mgmt:1.5.0
docker tag codesafe-mgmt:1.5.0 $DOCKER_REGISTRY/keysafe5/codesafe-mgmt:1.5.0
docker tag hsm-mgmt:1.5.0 $DOCKER_REGISTRY/keysafe5/hsm-mgmt:1.5.0
docker tag mgmt-ui:1.5.0 $DOCKER_REGISTRY/keysafe5/mgmt-ui:1.5.0
docker tag sw-mgmt:1.5.0 $DOCKER_REGISTRY/keysafe5/sw-mgmt:1.5.0
docker tag bitnami/mongodb:8.0.13-debian-12-r0-2025-09-15 $DOCKER_REGISTRY/keysafe5/bitnami/mongodb:8.0.13-
debian-12-r0-2025-09-15
docker tag bitnami/nginx:1.29.1-debian-12-r0-2025-09-15 $DOCKER_REGISTRY/keysafe5/bitnami/nginx:1.29.1-debian-12-
r0-2025-09-15

# Log in to ensure pushes succeed
docker login private.registry.local

# And push
docker push $DOCKER_REGISTRY/keysafe5/agent-mgmt:1.5.0
docker push $DOCKER_REGISTRY/keysafe5/codesafe-mgmt:1.5.0
docker push $DOCKER_REGISTRY/keysafe5/hsm-mgmt:1.5.0
docker push $DOCKER_REGISTRY/keysafe5/mgmt-ui:1.5.0
docker push $DOCKER_REGISTRY/keysafe5/sw-mgmt:1.5.0
docker push $DOCKER_REGISTRY/keysafe5/bitnami/mongodb:8.0.13-debian-12-r0-2025-09-15
docker push $DOCKER_REGISTRY/keysafe5/bitnami/nginx:1.29.1-debian-12-r0-2025-09-15
```

By setting the `DOCKER_REGISTRY` environment variable the deploy script will pull images from that registry. Otherwise the deploy script will set up a local insecure Docker registry. In this case, ensure that Docker is installed.

6.2.4.2. Kubernetes

If you have a Kubernetes cluster available, ensure that `kubectl` points to it, and that `kubectl get pods -A` returns a list of pods. Otherwise the deploy script will install K3s locally to `/usr/local/bin` and create a `${HOME}/.kube/config` to point to it. Kubernetes tools use the `KUBECONFIG` environment variable for the location of its configuration file, but when unset this defaults to `${HOME}/.kube/config`. For this setting to persist it needs to be added to your shell's configuration file.

6.2.4.3. Object Storage

If you do not have an existing Kubernetes cluster, or if your cluster contains only 1 worker node, the deploy script will use local storage on the single worker node.

If you would like to use an NFS for large object storage, set the environment variable `NFS_IP` to the NFS server address, and `NFS_PATH` to the path of the directory being exported from the NFS server.

To set the user and group IDs used by the KeySafe 5 application when accessing the object storage, configure the `podSecurityContext.runAsUser`, `podSecurityContext.runAsGroup` and `podSecurityContext.fsGroup` Chart parameters. To do this, specify the environment variable `KEYSAFE_BACKEND_CHART_EXTRA_ARGS`. For example, `KEYSAFE_BACKEND_CHART_EXTRA_ARGS="--set podSecurityContext.runAsUser=2000 --set podSecurityContext.runAsGroup=3000"`.

6.2.4.4. Istio

If you have Istio installed, ensure that `istioctl` is on your path. Otherwise the deploy script will download a local copy of `istioctl`, and install Istio as required.

6.2.4.5. MongoDB

Entrust recommends that you use your standard secure MongoDB Replica Set installation.

If you have an existing MongoDB deployment:

- Set the environment variable `MONGODB` to a backslash-comma separated list of servers, along with their port numbers in the form: `mongo-1.example.com:27017\,mongo-2.example.com:27017` The backslash should be visible when running `echo $MONGODB`. A quick tip: using single-quotes `'` will prevent the bash command line acting on the backslash you have typed.
- You will also need to create a Kubernetes generic secret in the `nshieldkeysafe5` namespace with `ca.crt`, `tls.crt`, and `tls.key` for a user that has readWrite roles on the data

bases: `agent-mgmt-db`, `codesafe-mgmt-db`, `hsm-mgmt-db` and `sw-mgmt-db`. Set `MONGO_SECRETS` to the name of this generic secret.

If you do not have an existing MongoDB deployment, the deploy script will set one up on the Kubernetes cluster along with the secrets for both the server and backend services.

MongoDB 5.0 and newer requires use of the AVX instruction set for processors. For more information, see MongoDB Production Notes

6.2.5. Authentication

To disable OIDC authentication, set the environment variable `DISABLE_AUTHENTICATION` to `yes`, and you may move on to Install KeySafe 5.

To configure authentication for Istio, the environment variable `AUTH_ISSUER_URL` needs to point at the issuer URL. Additionally, either `AUTH_JWKS` (for the payload) or `AUTH_JWKS_URL` (for the URL) also needs to be set. `AUTH_AUDIENCES` should be a comma-delimited list. The deploy script will automatically add the fully qualified domain name for the host to this list if not already present.

For UI authentication the deploy script requires an `OIDCProviders.json` file. Its location should be set in the environment variable `OIDC_PROVIDERS_FILE_LOCATION`.

Further details on configuring authentication for KeySafe 5 can be found in the Helm Chart Installation section of the *KeySafe 5 Installation Guide*.

6.2.6. Legacy KeySafe 5 agent support

By default, this KeySafe 5 central platform deployment will only be able to communicate with version 1.5 or later KeySafe 5 Agents. If you want your deployment to be able to communicate with legacy (1.4 or earlier) KeySafe 5 Agents then you must set the environment variable `AGENT_COMPATIBILITY` to `1`.

6.2.7. Install KeySafe 5

It is now possible to run the deploy script.

Do not run the deploy script under `sudo`. If `sudo` permissions are required, `sudo` will be called by the script and you will be prompted for your credentials.

The deploy script must be run from inside the directory to which it is extracted. Running with the -n flag will perform a set of pre-flight checks and show what will happen then exit without taking any action.

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

To disable authentication set the environment variable DISABLE_AUTHENTICATION to yes. Otherwise you may follow the instructions in the Authentication section.

You may now perform the deployment with the -y flag.

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

The script will take a few minutes to run, showing what actions are taking place. You may be prompted for your password by sudo, for example when installing K3s.

The script will create a local insecure Certificate Authority to be used by the agentcert.sh and updateinternalcerts.sh scripts. This directory should be preserved to allow this.

The script will also produce two archives, agent-config.tar.gz (for Unix) and agent-config.zip (for Windows), that contains the agent configuration file. The contents are used for configuring nShield client machines below.

6.2.7.1. K3s

If the deploy.sh script has installed K3s, you should configure kubectl access for the current user account by running:

```
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
```

This step is necessary for any user account on this machine that will be required to administrate the KeySafe 5 central platform.

You may append the export KUBECONFIG=${HOME}/.kube/config to your shell's configuration file.

6.2.7.2. Configure nShield Client Machines

In summary, to configure your nShield client machine to be managed and monitored by this deployment:

1. Install the KeySafe 5 agent on the nShield client machine containing the relevant Security World or HSMs.
2. Extract the `agent-config.tar.gz` or `agent-config.zip` archive on the nShield client machine alongside the KeySafe 5 agent.
3. Generate a unique private key and client CSR using the provided `ks5agenttls` for each individual KeySafe 5 agent.
4. Copy the client CSR to the central platform and sign it with the CA.
5. Copy the certificate to the nShield client machine, and place it alongside the KeySafe 5 agent configuration.

The steps to install and configure vary depending on the client. See Agent Configuration for more details.

6.2.8. Uninstall

If Kubernetes was not provided and K3s was installed by the deploy script, you may simply uninstall K3s which will clear up all the installed helm charts.

```
/usr/local/bin/k3s-uninstall.sh
```

This will request `sudo` permissions.

If a private Docker Registry was not provided, the deploy script will have created a local one and it will be removed when the script finishes. Should this fail, you may uninstall it manually by running:

```
docker stop registry
docker rm registry
```

If a Kubernetes installation was provided then the helm charts will need to be uninstalled individually.

```
helm --namespace nshieldkeysafe5 uninstall keysafe5-istio
helm --namespace nshieldkeysafe5 uninstall keysafe5-backend
helm --namespace nshieldkeysafe5 uninstall keysafe5-ui
```

If an existing MongoDB installation was not provided, then the deploy script will have installed a MongoDB helm chart that should be uninstalled.

```
helm --namespace mongons uninstall mongo-chart
```

If Istio was installed by the deploy script, it may be uninstalled by running:

```
keysafe5-1.5.0/istioctl uninstall --purge
```

To uninstall the KeySafe 5 agent, run the KeySafe 5 uninstaller:

```
sudo /opt/nfast/keysafe5/sbin/install -u
```

Finally secrets and pvc's can be deleted

```
kubectl --namespace=nshieldkeysafe5 delete agentcomms-server-certificates
kubectl --namespace=nshieldkeysafe5 delete agentcomms-client-certificates
kubectl --namespace=nshieldkeysafe5 delete pvc data-nshield-keysafe5
```

6.3. Manual Install Steps

The following steps provide a step-by-step guide to installing KeySafe 5 and its dependencies into an existing Kubernetes cluster.

An alternative to this guide is the Deploy Script which provides a scripted means of installing KeySafe 5.

These steps install KeySafe 5 and its dependencies. They should be followed to set up a demo environment for evaluation purposes and should *not* be used for production environments.

Please see Hardening The Deployment for steps to harden the deployment. Entrust recommends these steps as a minimum and that additional hardening may be required dependent on your own requirements.

A production deployment will have as a minimum the following:

- Maintained and patched versions of all the dependencies.
- A secure CA with TLS v1.3 support for certificates. The deploy script can provide a local insecure CA.
- A secure Kubernetes installation. The deploy script can install K3s locally.
- A secure MongoDB database. The deploy script can provide a replicated MongoDB with X.509 authentication running in Kubernetes.
- A secure means of large object storage. The deploy script can provide local object storage within the Kubernetes cluster or be config ured for using an NFS for object storage.

- HTTPS secured by a trusted certificate for the KeySafe 5 end-points. The deploy script will enable HTTPS connections with a self-signed insecure certificate.
- Require authentication to access KeySafe 5. OIDC & OAUTH2 are currently supported in KeySafe 5. The deploy script will not set up authenticated access.

This set of commands are an example of how to install KeySafe 5. They may need modification to suit your environment.

6.3.1. Unpack the release

```
mkdir ~/keysafe5-1.5.0
tar -xf nshield-keysafe5-1.5.0.tar.gz -C ~/keysafe5-1.5.0
cd ~/keysafe5-1.5.0/keysafe5-k8s
```

6.3.2. Docker images

The Docker images need to be loaded onto a Docker registry that each node in your Kubernetes cluster can pull the images from.

1. Load the Docker images to your local Docker, for example:

```
docker load < docker-images/agent-mgmt.tar
docker load < docker-images/codesafe-mgmt.tar
docker load < docker-images/hsm-mgmt.tar
docker load < docker-images/mongodb.tar
docker load < docker-images/nginx.tar
docker load < docker-images/sw-mgmt.tar
docker load < docker-images/ui.tar
```

2. Set the `DOCKER_REGISTRY` variable to the registry in use, for example:

```
export DOCKER_REGISTRY=localhost:5000
```

If you are using a single-machine Kubernetes distribution like K3s, you may be able to create a simple unauthenticated local private Docker registry by following the instructions in Distribution Registry. However this registry is only accessible by setting the name to `localhost` which will not work for other configurations.

3. Log in to the registry to ensure that you can push to it:

```
docker login $DOCKER_REGISTRY
```

4. Tag the Docker images for the registry, for example:

```
docker tag agent-mgmt:1.5.0 $DOCKER_REGISTRY/keysafe5/agent-mgmt:1.5.0
docker tag codesafe-mgmt:1.5.0 $DOCKER_REGISTRY/keysafe5/codesafe-mgmt:1.5.0
docker tag hsm-mgmt:1.5.0 $DOCKER_REGISTRY/keysafe5/hsm-mgmt:1.5.0
docker tag mgmt-ui:1.5.0 $DOCKER_REGISTRY/keysafe5/mgmt-ui:1.5.0
docker tag sw-mgmt:1.5.0 $DOCKER_REGISTRY/keysafe5/sw-mgmt:1.5.0
docker tag bitnami/mongodb:8.0.13-debian-12-r0-2025-09-15 $DOCKER_REGISTRY/keysafe5/bitnami/mongodb:8.0.13-
debian-12-r0-2025-09-15
docker tag bitnami/nginx:1.29.1-debian-12-r0-2025-09-15 $DOCKER_REGISTRY/keysafe5/bitnami/nginx:1.29.1-
debian-12-r0-2025-09-15
```

5. Push the KeySafe 5 images to the registry, for example:

```
docker push $DOCKER_REGISTRY/keysafe5/agent-mgmt:1.5.0
docker push $DOCKER_REGISTRY/keysafe5/hsm-mgmt:1.5.0
docker push $DOCKER_REGISTRY/keysafe5/codesafe-mgmt:1.5.0
docker push $DOCKER_REGISTRY/keysafe5/mgmt-ui:1.5.0
docker push $DOCKER_REGISTRY/keysafe5/sw-mgmt:1.5.0
docker push $DOCKER_REGISTRY/keysafe5/bitnami/mongodb:8.0.13-debian-12-r0-2025-09-15
docker push $DOCKER_REGISTRY/keysafe5/bitnami/nginx:1.29.1-debian-12-r0-2025-09-15
```

6.3.3. Set up a Certificate Authority

You should use your existing CA for a production system. This is simply used as an example for the purposes of having a working demo system.

Either OpenSSL 3.0 or OpenSSL 1.1.1 may be used to create the CA, and the CA may be cre ated in a directory of your choosing. In these examples, `/home/user/keysafe5-1.5.0/keysafe5-k8s/internalCA` is the example directory used. In that directory, create the file `internalCA.conf` with the contents:

```
[ ca ]
default_ca      = CA_default                         # The default ca section

[ CA_default ]

dir             = /home/user/keysafe5-{prod_vrsn}/keysafe5-k8s/internalCA  # The directory of the CA
database        = $dir/index.txt                     # index file.
new_certs_dir   = $dir/newcerts                      # new certs dir

certificate     = $dir/cacert.pem                    # The CA cert
serial          = $dir/serial                        # serial no file
#rand_serial    = yes                                # for random serial#'s
private_key     = $dir/private/cakey.pem             # CA private key
RANDFILE        = $dir/private/.rand                 # random number file

default_days    = 15                                 # how long to certify for
default_crl_days= 5                                  # how long before next CRL
default_md      = sha256                             # Message Digest
policy          = test_root_ca_policy
```

```
x509_extensions = certificate_extensions
unique_subject  = no
# This copy_extensions setting should not be used in a production system.
# It is simply used to simplify the demo system.
copy_extensions = copy

[ test_root_ca_policy ]
commonName = supplied
stateOrProvinceName = optional
countryName = optional
emailAddress = optional
organizationName = optional
organizationalUnitName = optional
domainComponent = optional

[ certificate_extensions ]
basicConstraints = CA:false

[ req ]
default_bits       = 4096
default_md         = sha256
prompt             = yes
distinguished_name = root_ca_distinguished_name
x509_extensions    = root_ca_extensions

[ root_ca_distinguished_name ]
commonName = hostname

[ root_ca_extensions ]
basicConstraints       = CA:true
keyUsage               = keyCertSign, cRLSign
subjectKeyIdentifier   = hash
authorityKeyIdentifier = keyid:always,issuer
basicConstraints       = critical,CA:true
```

Remember to update the `dir` value to the directory in which the `internalCA.conf` and the other CA files will be stored. The certificates generated, unless overridden on the command line, will be valid for 15 days as specified in `default_days`.

To generate the long-term CA key and random number source, create a directory called `private`, then place them in that directory:

```
mkdir ~/keysafe5-{prod_vrsn}/keysafe5-k8s/internalCA/private
openssl genrsa -out ~/keysafe5-{prod_vrsn}/keysafe5-k8s/internalCA/private/cakey.pem 4096
openssl rand -out ~/keysafe5-{prod_vrsn}/keysafe5-k8s/internalCA/private/.rand 1024
```

The CA needs a self-signed certificate; as this is a short-term demo it will be valid for 90 days:

```
cd ~/keysafe5-install/
openssl req -x509 -new -nodes \
  -key internalCA/private/cakey.pem \
  -subj "/CN=internalCA" -days 90 \
  -out internalCA/cacert.pem \
  -config internalCA/internalCA.conf
cp internalCA/cacert.pem ca.crt
```

And finally, to finish off the configuration:

```
mkdir internalCA/newcerts
echo 01 > internalCA/serial
touch internalCA/index.txt
```

6.3.4. Install and set up the supporting software

6.3.4.1. Kubernetes namespace

Create a namespace in Kubernetes for KeySafe 5 installation.

```
kubectl create namespace nshieldkeysafe5
```

6.3.4.2. Istio

These instructions assume that only Istio will be used for ingress, and no other ingress controller is installed.

If Istio is not already installed, you may install a version aligned with the software version of `istioctl` with:

```
istioctl install -y
```

6.3.4.3. MongoDB

Entrust recommends that you use your standard secure MongoDB Replica Set installation. This is just an example, and not production-ready.

6.3.5. TLS Secrets

By default the Bitnami MongoDB chart will create its own CA, and generate TLS keys for each of its servers from this CA. As we have an existing CA to use, we will pass the private key and certificate to MongoDB to use as its CA.

First, we need to create a TLS key and certificate for securing communications between the backend services and MongoDB. Here is an example:

```
export MONGOUSER="ks5-mongo-user"
openssl genrsa -out $MONGOUSER.key 4096
openssl req -new -key $MONGOUSER.key -out $MONGOUSER.csr \
  -subj "/CN=${MONGOUSER}" \
  -addext "keyUsage=digitalSignature" \
  -addext "extendedKeyUsage=clientAuth" \
  -addext "subjectAltName=DNS:${MONGOUSER}"
openssl ca -config ~/keysafe5-1.5.0/keysafe5-k8s/internalCA/internalCA.conf \
  -out ${MONGOUSER}.crt -in ${MONGOUSER}.csr -batch
rm ${MONGOUSER}.csr

kubectl create secret generic ks5-mongotls \
  --namespace nshieldkeysafe5 \
  --from-file=ca.crt \
  --from-file=tls.crt=${MONGOUSER}.crt \
  --from-file=tls.key=${MONGOUSER}.key
```

We will repeat the process to make the remaining keys, certificates and Kubernetes secrets.

Start by creating the following keys and certificate pairs:

- mongodb-arbiter
- mongodb-replica-0
- mongodb-replica-1

Now we use these to create the Kubernetes secrets:

```
kubectl create namespace mongons
kubectl create secret generic mongodb-certs-arbiter-0 \
  --namespace mongons \
  --from-file=ca.crt=mongodb-ca-cert \
  --from-file=tls.crt=mongodb-arbiter.crt \
  --from-file=tls.key=mongodb-arbiter.key

kubectl create secret generic mongodb-certs-0 \
    --namespace mongons \
    --from-file=ca.crt=mongodb-ca-cert \
    --from-file=tls.crt=mongodb-replica-0.crt \
    --from-file=tls.key=mongodb-replica-0.key

kubectl create secret generic mongodb-certs-1 \
    --namespace mongons \
    --from-file=ca.crt=mongodb-ca-cert \
    --from-file=tls.crt=mongodb-replica-1.crt \
    --from-file=tls.key=mongodb-replica-1.key
```

Now we install the MongoDB chart, using the CA secrets from above. This may take a few minutes.

```
helm install mongo-chart \
  --namespace=mongons \
  --set image.registry=$DOCKER_REGISTRY \
  --set image.tag=8.0.13-debian-12-r0-2025-09-15 \
  --set image.repository=keysafe5/bitnami/mongodb \
  --set tls.image.registry=$DOCKER_REGISTRY \
  --set tls.image.tag=1.29.1-debian-12-r0-2025-09-15 \
  --set tls.image.repository=keysafe5/bitnami/nginx \
  --set architecture=replicaset \
  --set auth.enabled=true \
  --set auth.usernames={dummyuser} \
  --set auth.passwords={dummypassword} \
  --set auth.databases={authdb} \
  --set tls.enabled=true \
  --set tls.mTLS.enabled=true \
  --set tls.autoGenerated=false \
  --set 'tls.replicaset.existingSecrets={mongodb-certs-0, mongodb-certs-1}' \
  --set tls.arbiter.existingSecret=mongodb-certs-arbiter-0 \
  --set global.security.allowInsecureImages=true \
  --wait --timeout 15m \
  helm-charts/bitnami-mongodb-17.0.0.tgz
```

There will be a message listing the MongoDB server addresses. Save the addresses to environment variables for use later.

```
export MONGO1=mongo-chart-mongodb-0.mongo-chart-mongodb-headless.mongons.svc.cluster.local:27017
export MONGO2=mongo-chart-mongodb-1.mongo-chart-mongodb-headless.mongons.svc.cluster.local:27017
export MONGODB=${MONGO1},${MONGO2}
```

The database tables and access for `MONGOUSER` also need to be set up. For this we will pass commands to mongosh running on the database server itself as the root user.

```
export MONGODB_ROOT_PASSWORD=$(kubectl get secret --namespace mongons \
  mongo-chart-mongodb -o jsonpath="{.data.mongodb-root-password}" \
  | base64 --decode)
export MONGO_TLSUSER=$(openssl x509 -in $MONGOUSER.crt -subject -noout | cut -f2- -d= | tr -d '[:space:]')
echo $MONGO_TLSUSER
```

Make a note of the `MONGO_TLSUSER` as you will need it shortly.

We need to start the mongosh command prompt.

```
kubectl -n mongons exec \
  --stdin=true mongo-chart-mongodb-0 0 -- \
  mongosh admin --tls \
  --tlsCAFile /certs/mongodb-ca-cert \
  --tlsCertificateKeyFile /certs/mongodb.pem \
  --host 127.0.0.1 \
  --authenticationDatabase admin \
  -u root -p $MONGODB_ROOT_PASSWORD
```

At the command prompt enter these database commands to create the tables.

```
db.createRole(
  {
    role: "hsm-mgmt-db-user",
    privileges: [
        {
          "resource": {"db": "hsm-mgmt-db", "collection": ""},
          "actions": ["createIndex", "find", "insert", "remove", "update"]
        },
      ],
    roles: []
  }
)
db.createRole(
  {
    role: "sw-mgmt-db-user",
    privileges: [
        {
          "resource": {"db": "sw-mgmt-db", "collection": ""},
          "actions": ["createIndex", "dropCollection", "find", "insert", "remove", "update"]
        },
      ],
    roles: []
  }
)
db.createRole(
  {
    role: "codesafe-mgmt-db-user",
    privileges: [
        {
          "resource": {"db": "codesafe-mgmt-db", "collection": ""},
          "actions": ["createIndex", "find", "insert", "remove", "update"]
        },
      ],
    roles: []
  }
)
db.createRole(
  {
    role: "agent-mgmt-db-user",
    privileges: [
        {
          "resource": {"db": "agent-mgmt-db", "collection": ""},
          "actions": ["createIndex", "dropCollection", "find", "insert", "remove", "update"]
        },
      ],
    roles: []
  }
)
```

We now need to create the user with access to the database. Replace `CN=ks5-mongo-user` with the value you got for `MONGO_TLSUSER`, and enter it into the prompt.

```
use $external
x509_user = {
  user : "CN=ks5-mongo-user",
  roles : [
    {"role": "agent-mgmt-db-user", "db": "admin" },
    {"role": "codesafe-mgmt-db-user", "db": "admin" },
    {"role": "hsm-mgmt-db-user", "db": "admin" },
    {"role": "sw-mgmt-db-user", "db": "admin" },
  ]
}
db.createUser(x509_user)
```

Type `exit` to exit the mongosh prompt.

6.3.5.1. Object Storage

For large object storage, create a Persistent Volume Claim, in the `nshieldkeysafe5` Kubernetes namespace (the same namespace that we will deploy the application to).

6.3.5.1.1. Cluster-local Object Storage

If your Kubernetes cluster only has 1 worker node, you can choose to use local storage.

```
cat << EOF | kubectl -n nshieldkeysafe5 apply -f -
  apiVersion: v1
  kind: PersistentVolumeClaim
  metadata:
    name: data-nshield-keysafe5
  spec:
    accessModes:
      - ReadWriteOnce
    storageClassName: local-path
    resources:
      requests:
        storage: 2Gi
EOF
```

6.3.5.1.2. NFS Object Storage

If your Kubernetes cluster has more than 1 worker node, you must use a type of storage that supports distributed access, such as NFS. For details on creating a PVC for NFS object storage, please see NFS Object Storage Configuration.

6.3.6. Install KeySafe 5

Bringing all the secrets and URLs created above, install KeySafe 5.

The commands below assume that a login is not required to pull from the Docker Registry.

By default, this KeySafe 5 central platform deployment will only be able to communicate with version 1.5 or later KeySafe 5 Agents.

If you want your deployment to be able to communicate with legacy (1.4 or earlier) KeySafe 5 Agents then you must include the configuration `--set messageBus.compatibilityMode=true` when running the `helm install` command.

```
# Get Ingress IP address
export INGRESS_IP=$(kubectl --namespace istio-system get svc -l app=istio-ingressgateway -o
jsonpath='{.items[0].status.loadBalancer.ingress[0].ip}')

# Install the KeySafe 5 backend services
helm install keysafe5-backend \
  --namespace=nshieldkeysafe5 \
  --set agent_mgmt.image=$DOCKER_REGISTRY/keysafe5/agent-mgmt:1.5.0 \
  --set codesafe_mgmt.image=$DOCKER_REGISTRY/keysafe5/codesafe-mgmt:1.5.0 \
  --set hsm_mgmt.image=$DOCKER_REGISTRY/keysafe5/hsm-mgmt:1.5.0 \
  --set sw_mgmt.image=$DOCKER_REGISTRY/keysafe5/sw-mgmt:1.5.0 \
  --set database.type=mongo \
  --set database.mongo.hosts="$MONGO1\,$MONGO2" \
  --set database.mongo.replicaSet=rs0 \
  --set database.mongo.auth.type=tls \
  --set database.mongo.auth.authDatabase=authdb \
  --set database.mongo.tls.enabled=true \
  --set database.mongo.tls.existingSecret=ks5-mongotls \
  --set messageBus.auth.type=tls \
  --set messageBus.tls.enabled=true \
  --set messageBus.tls.serverTLS.existingSecret=agentcomms-server-certificates \
  --set messageBus.tls.existingSecret=agentcomms-client-certificates \
  --set objectStore.pvc=data-nshield-keysafe5 \
  --wait --timeout 10m \
  helm-charts/nshield-keysafe5-backend-1.5.0.tgz

# Install the KeySafe 5 WebUI
helm install keysafe5-ui \
  --namespace=nshieldkeysafe5 \
  --set ui.image=$DOCKER_REGISTRY/keysafe5/mgmt-ui:1.5.0 \
  --set svcEndpoint="https://${INGRESS_IP}" \
  --set authMethod=none \
  --wait --timeout 10m \
  helm-charts/nshield-keysafe5-ui-1.5.0.tgz

# Create the TLS secret for the Istio Ingress Gateway
openssl genrsa -out istio.key 4096
openssl req -new -key istio.key -out istio.csr \
  -subj "/CN=${HOSTNAME}" \
  -addext "keyUsage=digitalSignature" \
  -addext "extendedKeyUsage=serverAuth" \
  -addext "subjectAltName=DNS:${HOSTNAME},IP:${INGRESS_IP}"
openssl ca -config ~/keysafe5-1.5.0/keysafe5-k8s/internalCA/internalCA.conf \
  -out istio.crt -in istio.csr -batch
kubectl -n istio-system create secret tls \
  keysafe5-server-credential --cert=istio.crt --key=istio.key

# Configure Istio Ingress Gateway for KeySafe 5
helm install keysafe5-istio \
  --namespace=nshieldkeysafe5 \
  --set tls.existingSecret=keysafe5-server-credential \
  --set requireAuthn=false \
  --wait --timeout 1m \
  helm-charts/nshield-keysafe5-istio-1.5.0.tgz
```

6.3.7. Access KeySafe 5

You can now access KeySafe 5 at `https://${INGRESS_IP}`.

For example, you could send curl requests as demonstrated below.

```
curl -X GET --cacert ca.crt https://${INGRESS_IP}/mgmt/v1/hsms | jq
curl -X GET --cacert ca.crt https://${INGRESS_IP}/mgmt/v1/hosts | jq
curl -X GET --cacert ca.crt https://${INGRESS_IP}/mgmt/v1/pools | jq
curl -X GET --cacert ca.crt https://${INGRESS_IP}/mgmt/v1/feature-certificates | jq
curl -X GET --cacert ca.crt https://${INGRESS_IP}/mgmt/v1/worlds | jq
curl -X GET --cacert ca.crt https://${INGRESS_IP}/codesafe/v1/images | jq
curl -X GET --cacert ca.crt https://${INGRESS_IP}/codesafe/v1/certificates | jq
```

You can access the Management UI in a web browser at `https://${INGRESS_IP}`.

6.3.8. Configure KeySafe 5 Agent machines

To configure a host machine to be managed and monitored by this deployment, run the KeySafe 5 agent binary on the KeySafe 5 Agent machine containing the relevant Security World or HSMs.

After copying over the agent tar file, extract it and start configuring:

```
sudo tar -C / -xf keysafe5-1.5.0-Linux-keysafe5-agent.tar.gz
export KS5CONF=/opt/nfast/keysafe5/conf
sudo cp $KS5CONF/config.yaml.example $KS5CONF/config.yaml
```

Create the messagebus/tls directory and copy the `ca.crt` file copied from the `keysafe5-1.5.0` directory on the demo machine into it.

```
mkdir -p $KS5CONF/messagebus/tls
cp ca.crt $KS5CONF/messagebus/tls/
```

Create the private key and a certificate signing request (CSR) for this specific KeySafe 5 agent.

```
sudo /opt/nfast/keysafe5/bin/ks5agenttls --keypath=$KS5CONF/messagebus/tls/tls.key --keygen
sudo /opt/nfast/keysafe5/bin/ks5agenttls --keypath=$KS5CONF/messagebus/tls/tls.key --csrgen
```

For this installation we copy the CSR to the demo machine, into the keysafe5-1.5.0 directory, then sign it using OpenSSL.

```
openssl ca -config ~/keysafe5-1.5.0/keysafe5-k8s/internalCA/internalCA.conf \
  -in ks5_demohost.csr \
  -out ks5_demohost.crt -batch
```

Transfer the resulting certificate `ks5_demohost.crt` to the nShield Agent machine at `/opt/nfast/keysafe5/conf/messagebus/tls/tls.crt`.

On the nShield Agent machine, if the hardserver is already running, use the KeySafe 5 install script to not restart it when installing the KeySafe 5 agent.

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

Otherwise, use the nShield install script which will start both the nShield Security World software and the KeySafe 5 agent.

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

6.3.9. Uninstall

```
helm --namespace nshieldkeysafe5 uninstall keysafe5-istio
helm --namespace nshieldkeysafe5 uninstall keysafe5-backend
helm --namespace nshieldkeysafe5 uninstall keysafe5-ui
helm --namespace mongons uninstall mongo-chart
```

6.4. Upgrade Steps

This chapter details how to update an existing KeySafe 5 install to the latest version.

When upgrading KeySafe 5 it is recommended to first update the Helm charts installed in the central platform and then update all KeySafe 5 Agent installs on host machines being managed by KeySafe 5.

6.4.1. Upgrading from KeySafe 5 1.4

To upgrade the release of a Helm Chart we do a `helm upgrade` command, see Helm Upgrade

List all installed releases using `helm list -A`.

```
$ helm list -A
NAME                NAMESPACE       REVISION    UPDATED                                 STATUS      CHART
APP VERSION
keysafe5-backend    nshieldkeysafe5 1           2024-04-25 15:59:40.995994525 +0100 BST deployed    nshield-
keysafe5-backend-1.4.0  1.4.0
keysafe5-istio      nshieldkeysafe5 1           2024-04-25 15:58:09.344300669 +0100 BST deployed    nshield-
keysafe5-istio-1.4.0    1.4.0
keysafe5-ui         nshieldkeysafe5 1           2024-04-25 15:57:42.260802671 +0100 BST deployed    nshield-
keysafe5-ui-1.4.0       1.4.0
mongo-chart         mongons         1           2024-04-25 15:55:05.825098514 +0100 BST deployed    mongodb-
12.1.31                 5.0.10
rabbit-chart        rabbitns        1           2024-04-25 15:58:19.881365343 +0100 BST deployed    rabbitmq-
11.16.2                3.11.18
```

Ensure all pods are healthy prior to performing an upgrade, unhealthy pods can prevent helm from fully completing an upgrade.

The process involves upgrading the charts in the following order:

1. mongo-chart
2. keysafe5-backend
3. keysafe5-ui
4. keysafe5-istio

6.4.1.1. Unpack the source

```
mkdir ~/keysafe5-1.5.0
tar -C ~/keysafe5-1.5.0 -xf nshield-keysafe5-1.5.0.tar.gz
cd ~/keysafe5-1.5.0/keysafe5-k8s
```

6.4.1.2. Docker Images

The Docker images need to be loaded onto a Docker registry that each node in your Kubernetes cluster can pull the images from.

Follow the instructions on this section of the install guide to do this: Docker Images.

6.4.1.3. Moving the CA

The CA needs to be moved from the 1.4 directory of KeySafe 5 to the 1.5.0 directory. Depending on your existing setup this is done in different ways. This guide includes the steps for moving internalCA and externalCA.

Both methods use the `~/keysafe5-1.5.0/keysafe5-k8s/updateinternalcerts.sh` script.

6.4.1.3.1. externalCA

For externalCA a new directory will need to be created in the 1.5.0 upgrade directory. This directory will need to contain the server, the client keys and certificates in PEM format.

```
mkdir ~/keysafe5-1.5.0/keysafe5-k8s/externalCA
```

The following files will need to be included in this directory:

```
  ca.crt                      The certificate of the CA that is to be trusted by
                              the system.
  agentcomms.key              The key to be used by the Agent Communications
                              Server
  agentcomms.crt              And its certificate
  ks5agentcomms.key           The key to be used by ks5
  ks5agentcomms.crt           And its certificate
  mongodb-0.key               The key to be used by mongo-chart-mongodb-0
  mongodb-0.crt               And its certificate
  mongodb-1.key               The key to be used by mongo-chart-mongodb-1
  mongodb-1.crt               And its certificate
  mongodb-a.key               The key to be used by mongo-chart-mongodb-arbiter
  mongodb-a.crt               And its certificate
  ks5mongodb.key              The key to be used by ks5-mongo-user
  ks5mongodb.crt              And its certificate
```

Once this directory has been created, updateinternalcerts.sh can be used to refresh certificates:

This specific command refresh certificates in the "certs" namespace, for more instructions refer to the help of updateinternalcerts.sh

```
./updateinternalcerts.sh -n certs externalCA
```

6.4.1.3.2. internalCA

If you are using internalCA then the CA is contained within a folder called "CA" or "internalCA" of the previous installation. Copy this existing folder into the current directory for the upgrade. This might be something like

```
cp -r ~/existing-ks5-install/internalCA .
```

Then generate the new certificates using updateinternalcerts.sh - the following example sets the expiration date for 1 year. This command may appear to fail, but if a folder called keysafe5-cert-update-agentcomms is created then this step was successful.

```
./updateinternalcerts.sh agentcomms 365
```

6.4.1.4. Create secrets

In order for the services to communicate with the message bus, secrets need to be created to be referenced during helm installation. Now that the CAs have been moved, these secrets can be made. Replace "CA" in the following scripts with the name of the CA you cre ated in the previous step.

```
kubectl create secret generic agentcomms-server-certificates \
  --namespace=nshieldkeysafe5 \
  --from-file=ca.crt=CA/cacert.pem \
  --from-file=tls.crt=keysafe5-cert-update-agentcomms/agentcomms.crt \
  --from-file=tls.key=keysafe5-cert-update-agentcomms/agentcomms.key

kubectl create secret generic agentcomms-client-certificates \
  --namespace=nshieldkeysafe5 \
  --from-file=ca.crt=CA/cacert.pem \
  --from-file=tls.crt=keysafe5-cert-update-agentcomms/keysafe5-backend-services.crt \
  --from-file=tls.key=keysafe5-cert-update-agentcomms/keysafe5-backend-services.key
```

6.4.1.5. MongoDB to 8.0.13

To update a non-Kubernetes existing MongoDB install to a MongoDB 8.0.13 install, see the official documentation at Upgrade to the Latest Revision of MongoDB.

To update a MongoDB 8.0.13 install deployed via Bitnami Helm Charts:

First ensure that MongoDB is running.

```
helm list -A
```

Fetch the existing helm chart's settings

```
helm -n mongons get values --output yaml mongo-chart > mongo-chart-values.yaml
```

The password and replicaset key have not changed, so we can upgrade.

```
helm upgrade --install mongo-chart \
  --namespace mongons \
  --set image.registry=$DOCKER_REGISTRY \
  --set image.tag=8.0.13-debian-12-r0-2025-09-15 \
  --set image.repository=keysafe5/bitnami/mongodb \
  --set tls.image.registry=$DOCKER_REGISTRY \
  --set tls.image.tag=1.29.1-debian-12-r0-2025-09-15 \
  --set tls.image.repository=keysafe5/bitnami/nginx \
  --set architecture=replicaset \
  --set auth.enabled=true \
  --set image.debug=false \
  --set systemLogVerbosity=5 \
  --set auth.rootPassword=secret \
  --set tls.enabled=true \
  --set tls.mTLS.enabled=true \
  --set tls.autoGenerated=false \
  --set 'tls.replicaset.existingSecrets={mongodb-certs-0, mongodb-certs-1}' \
  --set tls.arbiter.existingSecret=mongodb-certs-arbiter-0 \
  --set featureCompatibilityVersion=7.0 \
  --set global.security.allowInsecureImages=true \
  --values mongo-chart-values.yaml \
  --wait --timeout 3m \
  helm-charts/bitnami-mongodb-17.0.0.tgz
```

There will be some warnings regarding rolling tags and container substitution. This is expected as we are using the container images supplied in the release package. See Release Package - Docker Images for more information.

Obtain details of newly deployed helm charts

```
helm list -A
```

6.4.1.6. Define and update the new database users

First connect to the database

```
export MONGO1=mongo-chart-mongodb-0.mongo-chart-mongodb-headless.mongons.svc.cluster.local:27017
export MONGO2=mongo-chart-mongodb-1.mongo-chart-mongodb-headless.mongons.svc.cluster.local:27017
export MONGODB=${MONGO1},${MONGO2}
export MONGO_RUN="kubectl -n mongons exec mongo-chart-mongodb-0 0 -- "
export TLS_PRIVKEY="$(${MONGO_RUN} bash -c 'cat /certs/mongodb.pem')"
export TLS_CERT="$(${MONGO_RUN} bash -c 'cat /certs/mongodb-ca-cert')"

export MONGODB_ROOT_PASSWORD=$(kubectl get secret --namespace mongons \
mongo-chart-mongodb -o jsonpath="{.data.mongodb-root-password}" \
  | base64 --decode)

kubectl run --namespace mongons mongo-chart-mongodb-client \
  --rm --tty -i --restart='Never' --env="MONGODB_ROOT_PASSWORD=$MONGODB_ROOT_PASSWORD" \
  --env="TLS_PRIVKEY=$TLS_PRIVKEY" --env="TLS_CERT=$TLS_CERT" --env="MONGODB=$MONGODB" \
  --image $DOCKER_REGISTRY/keysafe5/bitnami/mongodb:8.0.13-debian-12-r0-2025-09-15 --command -- bash
```

then new definitions have to be added

```
$ echo "$TLS_CERT" > /tmp/tls.crt
$ echo "$TLS_PRIVKEY" > /tmp/tls.key
$ mongosh admin --tls --tlsCAFile /tmp/tls.crt --tlsCertificateKeyFile /tmp/tls.key \
  --host $MONGODB --authenticationDatabase admin -u root -p $MONGODB_ROOT_PASSWORD

> use admin
> db.createRole(
  {
    role: "agent-mgmt-db-user",
    privileges: [
        {
          "resource": {"db": "agent-mgmt-db", "collection": ""},
          "actions": ["createIndex", "dropCollection", "find", "insert", "remove", "update"]
        },
      ],
    roles: []
  }
)
> db.updateRole( "hsm-mgmt-db-user",
  {
    privileges : [
        {
          "resource": {"db": "hsm-mgmt-db", "collection": ""},
          "actions": ["createIndex", "dropIndex", "find", "insert", "remove", "update"]
        },
      ]
  }
)
> use $external
> x509_user = {
   "roles" : [
     {"role": "agent-mgmt-db-user", "db": "admin" },
     {"role": "codesafe-mgmt-db-user", "db": "admin" },
     {"role": "hsm-mgmt-db-user", "db": "admin" },
     {"role": "sw-mgmt-db-user", "db": "admin" },
   ]
 }
> db.updateUser("CN=ks5-mongo-user", x509_user)
> exit
$ exit
```

6.4.1.7. Upgrading the KeySafe 5 backend

The parameters used for the running Helm chart need to be retrieved into a file called `keysafe5-backed-values.yaml`.

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

The new services, monitoring_mgmt, licence_mgmt and agent_mgmt support the same common values as other services, such as probe thresholds. These can be added to the keysafe5-backend-values.yaml file if desired.

```
helm upgrade --install keysafe5-backend \
  --namespace=nshieldkeysafe5 \
  --values keysafe5-backend-values.yaml \
  --set hsm_mgmt.image=$DOCKER_REGISTRY/keysafe5/hsm-mgmt:1.5.0 \
  --set sw_mgmt.image=$DOCKER_REGISTRY/keysafe5/sw-mgmt:1.5.0 \
  --set codesafe_mgmt.image=$DOCKER_REGISTRY/keysafe5/codesafe-mgmt:1.5.0 \
  --set agent_mgmt.image=$DOCKER_REGISTRY/keysafe5/agent-mgmt:1.5.0 \
  --set messageBus.compatibilityMode=true \
  --set messageBus.URL=127.0.0.1:18084 \
  --set messageBus.auth.type=tls \
  --set messageBus.tls.enabled=true \
  --set messageBus.tls.existingSecret=agentcomms-client-certificates \
  --set messageBus.serverTLS.existingSecret=agentcomms-server-certificates \
  --values keysafe5-backend-values.yaml \
  --wait --timeout 3m \
  helm-charts/nshield-keysafe5-backend-1.5.0.tgz
```

6.4.1.8. Upgrading the KeySafe 5 WebUI

The same process as for the backend is also used for the WebUI:

```
helm -n nshieldkeysafe5 get values --all --output yaml keysafe5-ui > keysafe5-ui-values.yaml
```

You may make changes to the yaml files before upgrading though this is not required.

```
helm upgrade --install keysafe5-ui \
  --namespace=nshieldkeysafe5 \
  --set ui.image=$DOCKER_REGISTRY/keysafe5/mgmt-ui:1.5.0 \
  --set ui.pullPolicy=Always \
  --values keysafe5-ui-values.yaml \
  --wait --timeout 3m \
  helm-charts/nshield-keysafe5-ui-1.5.0.tgz
```

6.4.1.9. Upgrading the KeySafe 5 Istio

Check the version of Istio installed aligns with the software version of `istioctl`.

Enable the agent-mgmt port and certificates reference

```
helm -n nshieldkeysafe5 get values --all --output yaml keysafe5-istio > keysafe5-istio-values.yaml

istioctl x precheck

istioctl upgrade -y \
  --set values.gateways.istio-ingressgateway.ingressPorts[0].name=agent-comms \
  --set values.gateways.istio-ingressgateway.ingressPorts[0].port=18084 \
  --set values.gateways.istio-ingressgateway.ingressPorts[0].protocol=TCP

helm upgrade --install keysafe5-istio \
  --namespace=nshieldkeysafe5  \
  --values keysafe5-istio-values.yaml \
  --wait --timeout 3m  \
  helm-charts/nshield-keysafe5-istio-1.5.0.tgz
```

6.4.1.10. KeySafe 5 Agent Upgrade

To upgrade KeySafe 5 Agents, see Agent Upgrade.

6.4.1.11. Confirm Upgrade

To ensure that the upgrades have been successful, you can run the following commands and ensure they look similar to the recommended output.

First check:

```
helm list -A
```

Output should look like:

```
NAME                    NAMESPACE       REVISION        UPDATED                                 STATUS
CHART                                   APP VERSION
keysafe5-backend        nshieldkeysafe5 1               2025-09-10 14:18:14.413954531 +0100 BST deployed
nshield-keysafe5-backend-1.5.0    1.5.0
keysafe5-istio          nshieldkeysafe5 1               2025-09-10 10:53:07.370113525 +0100 BST deployed
nshield-keysafe5-istio-1.5.0      1.5.0
keysafe5-ui             nshieldkeysafe5 1               2025-09-10 10:44:59.935963805 +0100 BST deployed
nshield-keysafe5-ui-1.5.0         1.5.0
mongo-chart             mongons         1               2025-09-10 10:31:07.844261312 +0100 BST deployed
mongodb-17.0.0                          8.0.13
rabbit-chart            rabbitns        1               2025-09-10 09:54:54.077173236 +0100 BST deployed
rabbitmq-12.13.1                        3.12.13
```

As shown in the above example, rabbitmq may still be present. It does not affect the installa tion whether it is available or not. Once the upgrade is completed, the old RabbitMQ compo nents can be safely removed.

Second check:

```
kubectl get pods -A
```

Output should look like:

```
NAMESPACE         NAME                                          READY   STATUS    RESTARTS    AGE
kube-system       coredns-6799fbcd5-sflbm                       1/1     Running   0           5h39m
```

```
kube-system       local-path-provisioner-6c86858495-5v8rb      1/1     Running   0          5h39m
kube-system       svclb-rabbit-chart-rabbitmq-750352e4-jt2fl   5/5     Running   0          5h38m
nshieldkeysafe5   ratelimit-588cd5944c-85kck                   1/1     Running   0          5h32m
nshieldkeysafe5   redis-6d75fc6d74-sr852                       1/1     Running   0          5h32m
rabbitns          rabbit-chart-rabbitmq-1                      1/1     Running   0          5h32m
rabbitns          rabbit-chart-rabbitmq-0                      1/1     Running   0          5h31m
mongons           mongo-chart-mongodb-1                        1/1     Running   0          4h56m
mongons           mongo-chart-mongodb-arbiter-0                1/1     Running   0          4h56m
mongons           mongo-chart-mongodb-0                        1/1     Running   0          4h55m
nshieldkeysafe5   nshield-keysafe5-ui-6d9b97c57b-hjvgv         1/1     Running   0          4h42m
nshieldkeysafe5   nshield-keysafe5-ui-6d9b97c57b-wchpd         1/1     Running   0          4h42m
nshieldkeysafe5   nshield-keysafe5-ui-6d9b97c57b-xs9bl         1/1     Running   0          4h41m
istio-system      istiod-6f4cc8459f-wn6cp                      1/1     Running   0          4h35m
kube-system       svclb-istio-ingressgateway-106b0a9e-4lr7w    4/4     Running   0          4h35m
istio-system      istio-ingressgateway-c5fbff6d6-q4b9s         1/1     Running   0          4h35m
nshieldkeysafe5   nshield-keysafe5-2                           6/6     Running   0          69m
nshieldkeysafe5   nshield-keysafe5-1                           6/6     Running   0          69m
nshieldkeysafe5   nshield-keysafe5-0                           6/6     Running   0          68m
```

6.5. Helm Chart Details

To install the KeySafe 5 REST APIs you must install the helm-keysafe5-backend chart. A MongoDB instance will also be required. An existing deployment can be used but helm charts and images are provided in the release package. Please refer to the Manual Install Steps section for more information about using the supplied MongoDB helm charts and images.

To install the KeySafe 5 graphical user interface you must also install the helm-keysafe5-ui chart.

To expose these services externally to the Kubernetes cluster, see Configure external access to KeySafe 5.

If you are upgrading an existing KeySafe 5 install, see Helm Chart Upgrade.

6.5.1. Helm

Helm is a package manager for Kubernetes.

As with `kubectl` you can apply a namespace to the Kubernetes resources created by a Helm chart by specifying `--namespace my-namespace` when using `helm`.

To install the chart with the release name `my-release`:

```
helm install my-release helm-keysafe5-backend/
```

List all releases using `helm list`.

To upgrade or modify the existing `my-release` deployment, configure the chart parameters and then run:

```
helm upgrade --install my-release helm-keysafe5-backend/
```

To uninstall or delete the `my-release` deployment:

```
helm delete my-release
```

6.5.2. helm-keysafe5-backend

The keysafe5-backend Helm chart deploys Kubernetes Services for the KeySafe 5 REST APIs.

- agent-mgmt (nShield Agent Management Service API)
- codesafe-mgmt (nShield CodeSafe Management Service API)
- hsm-mgmt (nShield HSM Management Service API)
- sw-mgmt (nShield Security World Management Service API)

This Helm chart installs the KeySafe 5 services into a Kubernetes cluster but does not configure external access to the services. See Configure external access to KeySafe 5.

6.5.2.1. Kubernetes Secrets

The helm-keysafe5-backend Helm chart expects to be provided with pre-existing Kubernetes Secrets for the Database and messageBus connections.

For more information, refer to Kubernetes Secrets.

Purpose	Description	Secret Type
MongoDB SCRAM Auth	The username and password pairing used to authenticate with the MongoDB server	kubernetes.io/basic-auth
MongoDB TLS Certificates	TLS certificates used to connect to the MongoDB server. Can also be used as authentication if X509 auth is enabled	Opaque
MessageBus TLS Server Certificates	TLS certificates used as the message bus server.	Opaque
MessageBus TLS Client Certificates	TLS certificates used to connect to the message bus server.	Opaque

Because TLS secrets have the `Opaque` type, the filenames that are created are critical. Entrust KeySafe 5 expects the following:

- CA certificate to be named `ca.crt`.
- TLS certificate to be named `tls.crt`.
- TLS key file to be named `tls.key`.

Your certificates will need to adhere to X.509 v3, sometimes known as a multiple-domain certificates, or SAN certificates. The X.509 extension Subject Alternative Name (SAN) allows specifying multiple hostnames, and has replaced Common Name as the source of the hostname.

6.5.2.2. Configuration

To deploy the application, configure the chart with:

- The Docker images to use for `agent-mgmt`, `codesafe-mgmt`, `hsm-mgmt` and `sw-mgmt`.
- Database connection configuration.
- Message Bus connection configuration.
- A reference to the Persistent Volume Claim, in the same Kubernetes namespace, to use for large object storage.
- A reference to the KeyControl Compliance Manager's certificate authority

For further details on the configurable values of the Helm chart, see the `README.md` in the root directory of the Helm chart. For example, you may wish to:

- Increase or decrease the verbosity of the logging in the backend services.
- Change the period of time before a resource liveness health check is marked as failing. For example, if you have decreased the rate at which KeySafe 5 agents report to the central platform, you will want to increase the `health.livenessFailurePeriod` value.

To use the KeyControl Compliance Manager, you must add its certificate authority file path to the config map.

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

An example install:

```
helm install my-release \
  --create-namespace --namespace nshieldkeysafe5 \
  --set database.type=mongo \
  --set database.mongo.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" \
  --set database.mongo.replicaSet=rs0 \
  --set database.mongo.auth.type=pwd \
  --set database.mongo.auth.existingSecret=my-mongo-credentials-secret \
  --set database.mongo.tls.enabled=true \
  --set database.mongo.tls.existingSecret=my-mongo-tls-secret \
  --set messageBus.compatibilityMode=true \
  --set messageBus.URL=127.0.0.1:18084 \
  --set messageBus.auth.type=tls \
  --set messageBus.tls.enabled=true \
  --set messageBus.tls.existingSecret=agentcomms-client-certificates \
  --set messageBus.serverTLS.existingSecret=agentcomms-server-certificates \
  --set objectStore.pvc=data-nshield-keysafe5 \
  --set logging.level=info \
  --set logging.format=json \
  --set integrations.kcm.caConfigMap=compliance-ca \
  helm-charts/nshield-keysafe5-backend-{prod_vrsn}.tgz
```

6.5.3. helm-keysafe5-ui

The keysafe5-ui Helm chart deploys a Kubernetes Service for the KeySafe 5 Graphical User Interface.

The chart deploys the web front-end of KeySafe 5. For the WebUI to be usable it must point to a deployed KeySafe 5 back-end services endpoint (as installed by helm-keysafe5-backend).

This Helm chart installs the KeySafe 5 WebUI into a Kubernetes cluster, but does not configure external access to the service. (See Configure external access to KeySafe 5.)

6.5.3.1. Configuration

To deploy the application, configure the chart with:

- The Docker image to use for the `ui` container.
- OIDC Identity Provider config, if Authentication is enabled.
- The location of the KeySafe 5 back-end services API that this WebUI displays information for.

For further details on the configurable values of the Helm chart, see the `README.md` in the root directory of the Helm chart.

An example install:

```
# Untar the chart and copy your OIDC provider config file into the config directory
tar -xf helm-charts/nshield-keysafe5-ui-1.5.0.tgz -C helm-charts
cp my-oidc-provider-config.json helm-charts/nshield-keysafe5-ui/config/OIDCProviders.json

# Install the chart
helm install keysafe5-ui \
  --create-namespace --namespace nshieldkeysafe5 \
  --set svcEndpoint="https://XXX.XXX.XXX.XXX" \
  --set authMethod=oidc \
  --wait --timeout 10m \
  helm-charts/nshield-keysafe5-ui
```

Because the OIDC Provider configuration is volume mapped into the Kubernetes application, you must untar the packaged Helm chart so that you can copy in the `OIDCProviders.json` file to the correct location before installing the chart.

6.5.3.2. Configure WebUI authentication

If `authMethod` is set to `oidc` then you must provide an OIDC configuration file detailing the accepted Identity Provider configuration. Please refer to your Identity Provider's documentation for details, these items are usually returned from its `well-known` configuration endpoint.

Copy the OIDC configuration file to `config/OIDCProviders.json` in the root directory of the Helm chart before installing the Helm chart. This file is a JSON document. You can find an example file at `config/OIDCProviders.json.example`.

The configuration document is a JSON list of individual provider configurations.

An example of the `OIDCProviders.json` file containing a single Identity Provider:

```
[
    {
        "name":"Example Provider",
        "authority": "https://example-auth-provider.com/auth",
        "client_id":"8acc1449-7275-4524-b25f-4a60ddddfe8d",
        "redirect_uri": "https://example-keysafe5.com/callback",
        "response_type": "code",
        "scope": "openid",
        "post_logout_redirect_uri": "https://example-keysafe5.com/",
        "issuer": "https://example-auth-provider.com/auth",
        "authorization_endpoint": "https://example-auth-provider.com/auth/callback/authorize",
        "token_endpoint": "https://example-auth-provider.com/auth/token",
        "jwks_uri": "https://example-auth-provider.com/auth/keys",
        "end_session_endpoint": "https://example-auth-provider.com/auth/logout",
        "userinfo_endpoint": "https://example-auth-provider.com/auth/userinfo",
        "iconSVG" : "login.svg"
    }
]
```

To display a custom icon for the provider in the user interface, copy an SVG-format image to the `config` directory and reference the file name under the key `iconSVG` in the provider configuration.

6.5.4. Configure external access to KeySafe 5

To enable external access to the services installed by `helm-keysafe5-backend` and `helm-keysafe5-ui`, you need to configure a Kubernetes Ingress Gateway to route requests to the appropriate Kubernetes Services.

If you use Istio, you can use the `helm-keysafe5-istio` Helm chart to configure an existing Istio Ingress Gateway. Alternatively, you can configure your own Ingress Gateway. (See Configure a custom ingress provider.)

6.5.4.1. helm-keysafe5-istio

The keysafe5-istio Helm chart creates an Istio Gateway and VirtualService for the KeySafe 5 back-end services and user interface to be accessible externally from the Kubernetes clus ter.

The chart:

- Routes HTTP requests to the KeySafe 5 application running in the same Kubernetes cluster.
- Authenticates requests, if authentication is enabled (default) and an Identity Provider (IdP) is configured.
- Applies CORS policy to requests to limit Cross-Origin Resource Sharing.
- Applies security related HTTP Headers to responses including automatically applying the Content-Security-Policy header to help reduce Cross Site Scripting (XSS) risks.
- Limits TLS protocol used to TLS v1.2 and higher.
- Limits ciphers supported by the gateway.

6.5.4.2. Configure helm-keysafe5-istio

For further details of the configurable values of the Helm chart, see the `README.md` in the root directory of the Helm chart.

An example install:

```
helm install keysafe5-istio \
  --create-namespace --namespace=nshieldkeysafe5 \
  --set requireAuthn=true \
  --set httpsEnabled=true \
  --set portNumber=443 \
  --set issuer[0].authIssuer="https://foobar.auth0.com" \
  --set issuer[0].authJWKsURI="https://www.googleapis.com/oauth2/v1/certs" \
  --set issuer[0].authAudiences[0]="https://keysafe5.location" \
  --wait --timeout 1m \
  helm-charts/nshield-keysafe5-istio-1.5.0.tgz
```

6.5.4.3. helm-keysafe5-istio port number

If the port number changes to something other than 443 or 80, you need to open a port on the Istio Ingress Gateway. You can do this using your own istioctl install manifest. For more information, refer to https://istio.io/latest/docs/setup/install/istioctl/.

6.5.4.4. helm-keysafe5-istio authentication

Authentication can be provided by any OpenID Connect Provider, such as Entrust Identity As A Service.

If `requireAuthn` is `true`, then at least one `authIssuer` must be configured.

An example configuration:

```
issuers:
  - authIssuer: 'https://foobar.auth0.com'
    authJWKsURI: 'https://www.googleapis.com/oauth2/v1/certs'
    authJWKs: ''
    authAudiences:
    - 'https://keysafe5.location'
```

For each `authIssuer`, `authAudiences` and one of `authJWKsURI` or `authJWKs` must be specified.

Key	Description
`authIssuer`	Identifies the issuer that issued the JWT. A JWT with different iss claim will be rejected.
`authAudiences`	Identifies the list of JWT audiences that are allowed access. A JWT containing any of these audiences will be accepted.
`authJWKsURI`	URL of the provider's public key set to validate signature of the JWT. For more information, refer to https://openid.net/specs/openid-connect-discovery-1_0.html#ProviderMetadata.
`authJWKs`	JSON Web Key Set of public keys to validate signature of the JWT. For more information, refer to https://auth0.com/docs/jwks.

For additional information, refer to https://istio.io/latest/docs/reference/config/security/request_authentication/#JWTRule.

Once authenticated, a user has full access to view and manage the HSM and Security World resources accessible from the platform. No authorization policy is applied to requests.

6.5.4.5. Enable HTTPS for helm-keysafe5-istio

Configuring a TLS certificate for the Istio Gateway requires creating a Kubernetes secret.

To create a Kubernetes secret from an existing TLS private key and certificate:

You must create the Kubernetes Secret in the same namespace as the Istio Ingress Gateway.

```
kubectl --namespace istio-system create secret tls keysafe5-server-credential \
  --cert=path/to/cert/file \
  --key=path/to/key/file
```

6.5.5. Configure a custom ingress provider

If you do not want to use Istio, you can configure your own Kubernetes Ingress.

If you configure your own Ingress to the application, then it is your responsibility to configure routing to the services and any authentication or authorization to access the services.

Each part of the KeySafe 5 application is exposed as a Kubernetes Service.

`helm-keysafe5-backend` exposes the following Kubernetes Services for serving requests to the RESTful APIs:

ClusterIP Port	API endpoints
18080	/mgmt/v1/hsms /mgmt/v1/hosts /mgmt/v1/pools /mgmt/v1/feature-certificates /mgmt/v1/upgrade-images
18081	/mgmt/v1/worlds
18082	/codesafe/v1
18088	/system/v1/agents

Also, the following ports are exposed:

ClusterIP Port	Service
18084	KeySafe 5 Agent Communications

`helm-keysafe5-ui` exposes a Kubernetes Service called `keysafe5-ui-svc` on ClusterIP port `8080` for accessing the graphical user interface of KeySafe 5.

6.6. Certificate Details

In KeySafe 5 Kubernetes deployment, certificates are used to secure communications. The following sections provide details about the different types of certificates used in KeySafe 5.

6.6.1. WebUI/API Interface Certificates

Certificates are used to secure the API and WebUI interface. These certificates are found in Kubernetes Secret resource defined by the value `tls.existingSecret` in the `helm-keysafe5-istio` Helm Chart.

File Name	Description
tls.crt	The TLS certificate for the WebUI/API interface.
tls.key	The private key for the WebUI/API interface.

6.6.2. Agent Communication Certificates

TLS certificates are used to secure communication between KeySafe 5 central platform and KeySafe 5 Agents.

6.6.2.1. Server Certificates

The server certificates are used by KeySafe 5 to secure the Agent communications interface.

These certificates are found in Kubernetes Secret resource defined by the value `messageBus.serverTLS.existingSecret` in the `helm-keysafe5-backend` Helm Chart.

File Name	Description
tls.crt	The server TLS certificate for the KeySafe 5 Agent communication interface.

File Name	Description
tls.key	The private key for the KeySafe 5 Agent communication interface.
ca.crt	The CA certificate used to sign the TLS certificate and all KeySafe 5 Agent certificates. KeySafe 5 Agents not signed by this CA will not be able to connect to the KeySafe 5 Service.

6.6.2.2. Client Certificates

The client certificates are used by KeySafe 5 internally to authenticate the connection to the Agent Communication interface.

These certificates are found in Kubernetes Secret resource defined by the value `messageBus.tls.existingSecret` in the `helm-keysafe5-backend` Helm Chart.

> The TLS certificate that KeySafe 5 uses for connection to the Agent Communications interface must contain `keysafe5-backend-services` in the certificate's Distinguished Name so that the Agent Communications interface can properly limit permissions for this certificate. If the certificate's DistinguishedName does not contain `keysafe5-backend-services` then KeySafe 5 will be unable to connect to the Agent Communication interface.

File Name	Description
tls.crt	The TLS certificate for KeySafe 5 to authenticate to the Agent Communi cation interface. This certificate is signed by the CA used to sign KeySafe 5 Agent certificates.
tls.key	The private key for the KeySafe 5 Service to authenticate to the Agent Communication interface.
ca.crt	The CA certificate used to sign the Agent Communication Server certificates.

6.7. Database

All persistent data for KeySafe 5 is stored in the database.

For the Kubernetes deployment of KeySafe 5, MongoDB is used as the database.

6.7.1. MongoDB database

KeySafe 5 stores data in multiple different databases within MongoDB.

The names of the databases used within MongoDB are defined by the `helm-keysafe5-backend` Helm chart.

Key	Description	Default value
agent_mgmt.dbName	Name of the database to use for storing persistent Agent data	agent-mgmt-db
codesafe_mgmt.dbName	Name of the database to use for storing persistent CodeSafe data	codesafe-mgmt-db
hsm_mgmt.dbName	Name of the database to use for storing persistent HSM data	hsm-mgmt-db
sw_mgmt.dbName	Name of the database to use for storing persistent Security World data	sw-mgmt-db

6.7.1.1. Collections

6.7.1.1.1. Agent Management database

KeySafe 5 stores nShield agent data in the following collections:

- agents

6.7.1.1.2. HSM Management database

KeySafe 5 stores nShield HSM related data in the following collections:

- config
- features
- hardservers
- hosts
- hsms
- hsmoperations
- images
- pools
- tenancies

6.7.1.1.3. Security World Management database

KeySafe 5 stores nShield Security World data in the following collections:

- worlds
- versions

For each Security World known to KeySafe 5, the following collections are automatically cre ated, where each collection name is prefixed by the ID of the Security World database record that the collection corresponds to:

- <id>_actions
- <id>_authorizations
- <id>_authorized_pools
- <id>_cards
- <id>_cardsets
- <id>_domains
- <id>_groups
- <id>_keys
- <id>_module_certs
- <id>_operations
- <id>_p11objects
- <id>_softcards
- <id>_secrets
- <id>_kcmconnection

6.7.1.1.4. CodeSafe Management database

KeySafe 5 stores nShield CodeSafe related data in the following collections:

- certificates
- certificatestatus
- images
- machines
- operations
- steps

6.7.1.2. User roles

MongoDB has the notion of roles, where each role has a defined set of allowed actions. A user of a MongoDB database can be given a role which then determines what the user can and cannot do to the data.

For details about MongoDB roles, see the MongoDB documentation.

From a security point of view we want to give KeySafe 5 as a user of the MongoDB database the least privileges which suffice for the functionality it requires from the MongoDB database.

The documentation below details the minimum privileges required for a KeySafe 5 MongoDB user for each database created by KeySafe 5.

6.7.1.2.1. Agent Management database

The following actions are required by KeySafe 5 for the operation of MongoDB for the Agent Management collections:

- createIndex
- dropCollection
- find
- insert
- remove
- update

The MongoDB administrator will configure the Agent Management database with the follow ing actions and privileges for KeySafe 5 `agent-mgmt-db-user` role:

```
use admin
db.createRole(
  {
    role: "agent-mgmt-db-user",
    privileges: [
        {
          "resource": {"db": "agent-mgmt-db", "collection": ""},
          "actions": ["createIndex", "dropCollection", "find", "insert", "remove", "update"]
        },
      ],
    roles: []
  }
)
```

6.7.1.2.2. HSM Management database

The following actions are required by KeySafe 5 for the operation of MongoDB for the HSM Management collections:

- createIndex
- find
- insert
- remove
- update

The MongoDB administrator will configure the HSM Management database with the following actions and privileges for KeySafe 5 `hsm-mgmt-db-user` role:

```
use admin
db.createRole(
  {
    role: "hsm-mgmt-db-user",
    privileges: [
        {
          "resource": {"db": "hsm-mgmt-db", "collection": ""},
          "actions": ["createIndex", "find", "insert", "remove", "update"]
        },
      ],
    roles: []
  }
)
```

6.7.1.2.3. Security World Management database

As KeySafe 5 creates new collections in the Security World Management Database as new Security Worlds are introduced to the system, RBAC (Role-based access control) must be applied at the database level rather than individual collections.

The following actions are required by KeySafe 5 for the operation of MongoDB for the Secu rity World Management collections:

- createIndex
- dropCollection
- find
- insert
- remove
- update

The MongoDB administrator will configure the Security World Management database with the following actions and privileges for KeySafe 5 `sw-mgmt-db-user` role:

```
use admin
db.createRole(
  {
    role: "sw-mgmt-db-user",
    privileges: [
        {
          "resource": {"db": "sw-mgmt-db", "collection": ""},
          "actions": ["createIndex", "dropCollection", "find", "insert", "remove", "update"]
        },
      ],
    roles: []
  }
)
```

6.7.1.2.4. CodeSafe Management database

The following actions are required by KeySafe 5 for the operation of MongoDB for the Code Safe Management collections:

- createIndex
- find
- insert
- remove
- update

The MongoDB administrator will configure the CodeSafe Management database with the following actions and privileges for KeySafe 5 `codesafe-mgmt-db-user` role:

```
use admin
db.createRole(
  {
    role: "codesafe-mgmt-db-user",
    privileges: [
        {
          "resource": {"db": "codesafe-mgmt-db", "collection": ""},
          "actions": ["createIndex", "find", "insert", "remove", "update"]
        },
      ],
    roles: []
  }
)
```

6.7.1.2.5. Creating a MongoDB user with the user-defined roles

The MongoDB administrator may create a user for the KeySafe 5 application to access the KeySafe 5 databases by using the `db.createUser` command in the MongoDB shell.

```
ks5_user = {
   "user" : "ks5username",
   "roles" : [
     {"role": "agent-mgmt-db-user", "db": "admin" },
     {"role": "codesafe-mgmt-db-user", "db": "admin" },
     {"role": "hsm-mgmt-db-user", "db": "admin" },
     {"role": "sw-mgmt-db-user", "db": "admin" },
   ]
 }
```

```
> db.createUser(ks5_user)
```

Note that when using TLS authentication for MongoDB, the username needs to match the subject of the client certificate.

6.7.1.3. Authentication methods

KeySafe 5 supports the following authentication mechanisms for access to the MongoDB server:

- No authentication
- SCRAM
- X.509 certificate authentication

The type of authentication is specified by `database.mongo.auth.type` value in the `helm-keysafe5-backend` Helm chart.

6.7.1.3.1. No authentication

This option is used during development. It should not be used during production.

6.7.1.3.2. SCRAM

Using Salted Challenge Response Authentication Mechanism (SCRAM), MongoDB verifies the supplied credentials against the MongoDB's username, password and authentication database.

In the `helm-keysafe5-backend` Helm chart:

- `database.mongo.auth.type` must be set to `pwd`.
- `database.mongo.auth.existingSecret` must be set to the name of an existing Kubernetes Secret that contains the username and password to use (the Secret must contain a value for `username` and `password` keys).
- `database.mongo.auth.authDatabase` must be set to the name of MongoDB's authentica tion database.

6.7.1.3.3. X.509 certificate authentication

KeySafe 5 can use X.509 certificates instead of usernames and passwords to authenticate to the MongoDB database.

In the `helm-keysafe5-backend` Helm chart:

- `database.mongo.auth.type` must be set to `tls`.
- `database.mongo.tls.enabled` must be set to `true`.
- `database.mongo.tls.existingSecret` must be set to the name of an existing Kubernetes Secret that contains the TLS certificates to use (the Secret must contain the keys `tls.crt`, `tls.key` and `ca.crt`).

6.7.1.4. Backup

To be able to restore the KeySafe 5 application, Entrust recommends that you regularly backup the MongoDB database as suggested in the MongoDB documentation.

6.7.1.5. Maintenance

The KeySafe 5 application (`helm-keysafe5-backend` Helm chart) does not support having database collections removed while the application is running.

If deleting collections, or replacing the MongoDB server that KeySafe 5 uses, then please stop the `helm-keysafe5-backend` Helm chart before performing database maintenance and restart the application once the database maintenance is complete.

6.8. Hardening The Deployment

To harden the demo deployment there are a number of steps to follow. The documentation below requires modifying the configuration of the Helm charts installed by following the Manual Install steps or running the `deploy.sh` script. To obtain the installed configuration for each installed Helm chart, run the following commands:

```
helm -n nshieldkeysafe5 get values --all --output yaml keysafe5-backend > keysafe5-backend-values.yaml
helm -n nshieldkeysafe5 get values --all --output yaml keysafe5-ui > keysafe5-ui-values.yaml
helm -n nshieldkeysafe5 get values --all --output yaml keysafe5-istio > keysafe5-istio-values.yaml
```

Documentation for each configurable value in the KeySafe 5 Helm charts can be found by untarring the chart.tgz and viewing the contents of either README.md or the default values.yaml file.

6.8.1. Certificates

The Manual Install Steps steps and the `deploy.sh` script will generate a local Certificate Authority, and install a number of short-lived demo certificates. These certificates must be replaced to continue using the system after their expiry.

Your new certificates will need to adhere to X.509 v3, sometimes known as a multiple-domain certificates, or SAN certificates. The X.509 extension Subject Alternative Name (SAN) allows specifying multiple hostnames, and has replaced Common Name as the source of the hostname.

6.8.1.1. External KeySafe 5 Server TLS Certificate

To update the TLS certificate used for the KeySafe 5 Server (for HTTPS connections to the REST API or User Interface) you must create a Kubernetes Secret containing the new certificate/private key and redeploy the `keysafe5-istio` Helm chart.

For more information on enabling HTTPS for helm-keysafe5-istio, see the Helm Chart Installation section of the Installation Guide.

The Manual Install steps and `deploy.sh` script create a Kubernetes Secret called `keysafe5-server-credential`. You can either delete the existing Secret as shown below, or use a different name for the Secret containing your new TLS certificate.

```
kubectl --namespace istio-system delete secret keysafe5-server-credential

kubectl --namespace istio-system create secret tls keysafe5-server-credential \
  --cert=path/to/cert/file \
  --key=path/to/key/file
```

Before running `helm upgrade`, set the following values in your `keysafe5-istio-values.yaml`:

- httpsEnabled=true
- tls.existingSecret=keysafe5-server-credential (or the name you used when creating the Kubernetes Secret containing your certificate/private key)

```
helm upgrade --install keysafe5-istio \
  --namespace=nshieldkeysafe5 \
  --values keysafe5-istio-values.yaml \
  --wait --timeout 1m \
helm-charts/nshield-keysafe5-istio-1.5.0.tgz
```

6.8.1.2. Internal Certificates

The Manual Installation steps and `deploy.sh` script create a Certificate Authority, and install KeySafe 5 using TLS authorised by the CA for the communications between the central platform and MongoDB.

You can refresh these internal certificates by running the `updateinternalcerts.sh` script, specifying the number of days for which the new certificates will be valid.

Alternatively if you have an external CA, you may generate keys and certificates and pass the directory containing them to `updateinternalcerts.sh` script.

The help for `updateinternalcerts.sh` is shown with the `-h` option:

```
Usage: ./updateinternalcerts.sh [OPTION]... [SERVER] DAYS
       ./updateinternalcerts.sh [OPTION]... [SERVER] DIRECTORY

Update the internal server and client TLS credentials for a deployment created
by deploy.sh.

This can update certificates and keys for both the Agent Communications
Server and MongoDB as set up by deploy.sh, along with the certificates and
keys for the KeySafe 5 BackEnd services.  The Agent Communications Server has
a single key and certificate.  MongoDB has three keys and certificates, two
for the replica-set, and one for the arbiter.  The KeySafe 5 backend services
also have two keys and client certificates for connecting to both the Agent
Communications Server, and MongoDB.

When using the insecure internal CA, specify when the new keys and certs will
expire and this script will do the rest.

When using a secure external CA, all the keys and certificates will have to
be provided and they will be packaged up for the deployment.

When you update the certificates for the Agent Communications Server, the
script will generate an updated agent-config.tar.gz and agent-config.zip
files containing a config file and ca.crt.

Certificate Addresses

The certificates contain names and IP addresses that verify the connection,
signed by a mutally trusted authority.

The backend servers only use Kubernetes internal addressing to access
MongoDB, and 127.0.0.1 for the Agent Communications Server.

MongoDB is only used by the backend services so does not require an external
address.

The Agent Communications Server is also accessed by the KeySafe 5 Agents
through an external address.

Shared Optional parameters

-n, --namespace=NAMESPACE   The deploy script normally uses nshieldkeysafe5
                            as the namespace for the KeySafe 5 servers and
                            mongons for MongoDB.  However you can override
                            these to use a single namespace for all the
                            servers.  If that is the case, then use this
                            option.
SERVER                      One of "mongodb" or "agentcomms".
                            If the server is not specified then both will
                            be updated.
Internal CA
-----------

DAYS                        Number of days before the generated certificates
                            expire

This script will use the IP address provided by the loadbalancer, but you may
override this with a different IP address or DNS name.
```

```
-a, --agentcomms=ADDRESS    The external address for the Agent Communications
                            Server
-m, --mongodb=ADDRESS       The external address for MongoDB

External CA
-----------

DIRECTORY                   The directory containing the server and client
                            keys and certificates in PEM format.
Files in the directory
----------------------
ca.crt                      The certificate of the CA that is to be trusted by
                            the system.
agentcomms.key              The key to be used by the Agent Communications
                            Server
agentcomms.crt              And its certificate
ks5agentcomms.key           The key to be used by ks5
ks5agentcomms.crt           And its certificate
mongodb-0.key               The key to be used by mongo-chart-mongodb-0
mongodb-0.crt               And its certificate
mongodb-1.key               The key to be used by mongo-chart-mongodb-1
mongodb-1.crt               And its certificate
mongodb-a.key               The key to be used by mongo-chart-mongodb-arbiter
mongodb-a.crt               And its certificate
ks5mongodb.key              The key to be used by ks5-mongo-user
ks5mongodb.crt              And its certificate

If agentcomms is specified, the mongodb files may be skipped, and vice versa.

Internal CA examples

Refresh certificates for both servers for the next 30 days, and set a DNS
entry for the Agent Communications Server
  ./updateinternalcerts.sh -r keysafe5.example.com 30

Refresh just MongoDB for the next 4 weeks
  ./updateinternalcerts.sh mongodb 28

External CA examples

Refresh certificates for both servers (and KeySafe 5) in a specific namespace
  ./updateinternalcerts.sh -n kansas all-certs-dir

  Refresh certificates for just MongoDB (and KeySafe 5) in standard namespaces
  ./updateinternalcerts.sh mongodb mongo-certs-dir
```

This script will update MongoDB TLS certificates, though you may choose to update only one set of TLS certificates by specifying only that server.

The `updateinternalcerts.sh` script must be run from a directory containing the KeySafe 5 Helm charts (for example, from the root directory of the untarred KeySafe 5 package). If the CA, created when the original deploy script was run, is not available then a new one will be created and used.

If both certificates have expired when running `updateinternalcerts.sh`, updating the certificates of only one service may return an error. In this case re-running `updateinternalcerts.sh` without specifying any server

> will update both server certificates at once, and will usually solve the problem.

If an error occurs during certificate update you can restore the previous setup by rolling back Helm chart installations to a previous release, see Helm Chart Upgrade.

6.8.1.2.1. MongoDB TLS Certificates

The Manual Install steps and `deploy.sh` script installs the Bitnami packaged MongoDB Helm chart with TLS enabled for the connections between KeySafe 5 and the MongoDB server and also with TLS used for authentication. These certificates will initially be valid for 30 days from the time the process was run.

You can refresh the MongoDB certificates by running the `updateinternalcerts.sh` script, specifying the number of days the new certificates will be valid for.

```
updateinternalcerts.sh mongodb 365
```

This script will:

- Generate the new TLS certificates
- Update the MongoDB helm chart to use the new certificates
- Update the `keysafe5-backend` helm chart to use the new certificates

You may specify an external address with the `-m` parameter but this is usually not required.

For an external CA, the following command will apply the keys and certificates from the directory `mycerts`

```
updateinternalcerts.sh mongodb mycerts
```

If using an external CA, the following suggestions should improve the chances of success:

- Create keys for all the MongoDB servers (`mongo-chart-mongodb-arbiter-0`, `mongo-chart-mongodb-0`, and `mongo-chart-mongodb-1`)
- Use 4k RSA keys
- Set the Key Usage to `Digital Signature, Non Repudiation, Key Encipherment`
- For the server certificates, do not set the Extended Key Usage (the keys are used as both server and client keys)
- For the server certificates' Subject Alternative Name, add DNS records for `mongodb`, `mongo-chart-mongodb.mongons.svc.cluster.local`, `<server>.mongo-chart-mongodb-`

headless.mongons.svc.cluster.local, and mongo-chart.mongons.svc.cluster.local

- For the client certificate, set the Extended Key Usage as "TLS Web Client Authentication"
- For the client certificate's Subject Alternative Name, set the first entry to the DNS value of ks5-mongo-user, and add DNS entries for keysafe5-backend-ks5-mongo-user-headless, keysafe5-backend-ks5-mongo-user.nshieldkeysafe5.svc.cluster.local, and keysafe5-backend.nshieldkeysafe5.svc.cluster.local.

6.8.2. Authentication

If you chose to install the demo deployment without authentication you should enable authentication for accessing the KeySafe 5 REST API and User Interface.

For how to configure authentication for the KeySafe 5 REST APIs see Helm Chart Installation: helm-keysafe5-istio authentication in the Installation Guide.

To update the keysafe5-istio Helm chart installed by the demo deployment, set the following values in keysafe5-istio-values.yaml.

- requireAuthn=true
- issuer[0].authIssuer="https://foobar.auth0.com"
- issuer[0].authJWKsURI="https://www.googleapis.com/oauth2/v1/certs"
- issuer[0].authAudiences[0]="https://keysafe5.location"

Then run helm upgrade.

```
helm upgrade --install keysafe5-istio \
  --namespace=nshieldkeysafe5 \
  --values keysafe5-istio-values.yaml \
  --wait --timeout 1m \
helm-charts/nshield-keysafe5-istio-1.5.0.tgz
```

To update the keysafe5-ui Helm chart installed by the demo deployment, set the following values in keysafe5-ui-values.yaml:

- authMethod=oidc

Untar the chart and copy your OIDC provider config file (OIDCProviders.json) into the config directory:

For more details on how to populate OIDCProviders.json and how to configure authentication for the KeySafe 5 User Interface see Helm Chart Installation: Configure WebUI authentication in the Installation Guide.

```
tar -xf helm-charts/nshield-keysafe5-ui-1.5.0.tgz -C helm-charts
cp my-oidc-provider-config.json helm-charts/nshield-keysafe5-ui/config/OIDCProviders.json
```

Then run `helm upgrade`:

```
helm upgrade --install keysafe5-ui \
  --namespace=nshieldkeysafe5 \
  --values keysafe5-ui-values.yaml \
  --wait --timeout 3m \
helm-charts/nshield-keysafe5-ui
```

6.8.3. K3s

If not using your own Kubernetes cluster, the `deploy.sh` script will create one using K3s. To harden this K3s install follow the official documentation at K3s Hardening Guide.

> The `deploy.sh` script installs K3s with `traefik` and `metrics-server` explicitly disabled.

6.9. Troubleshooting

6.9.1. Obtaining Central platform service Logs

The KeySafe 5 application is configured to log to `stdout`. This means you can view logs by running standard `kubectl` commands.

To get the KeySafe 5 backend services logs run `kubectl get pods`

> By default, the KeySafe 5 backend Helm chart will create multiple replicas of each service. The below example commands only retrieves the logs from the first replica of each service.

```
kubectl -n nshieldkeysafe5 logs nshield-keysafe5-0 agent-mgmt
kubectl -n nshieldkeysafe5 logs nshield-keysafe5-0 codesafe-mgmt
kubectl -n nshieldkeysafe5 logs nshield-keysafe5-0 hsm-mgmt
kubectl -n nshieldkeysafe5 logs nshield-keysafe5-0 sw-mgmt
```

To get the KeySafe 5 WebUI logs.

```
UI_POD=$(kubectl -n nshieldkeysafe5 get pods -l app=keysafe5-ui-app -o jsonpath='{.items[0].metadata.name}')
kubectl logs -n nshieldkeysafe5 $UI_POD
```

Because all logs are directed to `stdout`, you can integrate the application logs with third-party log monitoring tools such as Prometheus or Splunk.

6.9.2. Kubernetes resource debug

If a Kubernetes resource is not working as expected, use `kubectl describe` to display any errors with that resource.

```
$ kubectl describe -n nshieldkeysafe5 pod nshield-keysafe5-0
[. . .]
Warning  FailedMount  6s (x8 over 70s)  kubelet            MountVolume.SetUp failed for volume "keysafe5-
messagebus-tls-volume" : secret "agentcomms-client-certificates" not found
```

You can also use `kubectl get events` to detect errors.

```
kubectl get events --all-namespaces
```

For more information on debugging Kubernetes applications, see the Kubernetes documentation here.

6.10. Uninstall Steps

6.10.1. Central platform

To fully remove the KeySafe 5 application from your Kubernetes cluster, use `helm uninstall`. This uninstalls all KeySafe 5 Helm charts.

You can use `helm list` to see which charts are installed. If you do not know the namespace, use `--all-namespaces` to show charts from all namespaces.

```
$ helm list -n nshieldkeysafe5 --short
keysafe5-backend
keysafe5-istio
keysafe5-ui
```

And to delete them:

```
helm uninstall keysafe5-ui --namespace=nshieldkeysafe5
helm uninstall keysafe5-istio --namespace=nshieldkeysafe5
helm uninstall keysafe5-backend --namespace=nshieldkeysafe5
```

KeySafe 5 application data remains in your MongoDB database after uninstalling the applica tion. To clear this data from the database, remove the databases that were defined by `agent_mgmt.dbName`, `codesafe_mgmt.dbName`, `hsm_mgmt.dbName`, and `sw_mgmt.dbName` in the `helm-`

`keysafe5-backend` chart.

6.10.1.1. Secrets

You can use `kubectl get secrets` to see the secrets.

```
kubectl get secrets --namespace=nshieldkeysafe5
```

And delete them:

```
kubectl --namespace=nshieldkeysafe5 delete secrets agentcomms-server-certificates
kubectl --namespace=nshieldkeysafe5 delete secret agentcomms-client-certificates
kubectl --namespace=nshieldkeysafe5 delete secret mongodb-demo-client-certificates
```

6.10.1.2. Volumes

You can use `kubectl get pvc` to see the persistent volumes.

```
kubectl get pvc --namespace=nshieldkeysafe5
```

And delete them

```
kubectl --namespace=nshieldkeysafe5 delete pvc data-nshield-keysafe5
```

6.10.2. MongoDB

If you have used the provided Bitnami MongoDB Helm chart, you can delete the MongoDB instance and all its data by uninstalling the `mongo-chart` Helm chart as shown below.

```
helm uninstall mongo-chart --namespace=mongons

# Get a list of MongoDB Secrets
kubectl get secrets --namespace=mongons

# Remove the MongoDB Secrets
kubectl --namespace=mongons delete secret <secret-name>

# Get a list of MongoDB Persistent Volume Claims
kubectl get pvc --namespace=mongons

# Remove the MongoDB Persistent Volume Claims
kubectl --namespace=mongons delete pvc <pvc-name>
```

7. KeySafe 5 Agent

The KeySafe 5 agent runs alongside the existing hardserver and enables the central management platform to manage all HSMs and Security Worlds visible to the hardserver. The agent communicates the current state of the HSMs / Security World to the central platform and can action management operations for these resources.

The connection between the agent and the central monitoring platform is configured in the KeySafe 5 agent configuration file.

The KeySafe 5 agent ensures that all key management data, with the exception of keys, is synchronized between the nShield client machine and a central (MongoDB or SQLite) database. Keys are only synced to the KeySafe 5 server.

This means that when resources, such as Card Sets or Softcards, appear in the `kmdata/local` directory on a client machine, they are automatically stored in the central database. It also means that when a Card Set or Softcard is created via the new management tools, the resource also appears in `kmdata/local` on any host machine that is in the right Security World.

The Card Set or Softcard can then be used with the traditional nShield tools on each nShield client machine.

> If a resource is deleted via the KeySafe 5 application then it will be removed from `kmdata/local` for all client machines, and Connects, running a KeySafe 5 agent. If the resource is deleted locally on a nShield client machine then that deletion is not synchronized to other client machines in the same Security World.

The KeySafe 5 agent will also report on the status of CodeSafe 5 machines/certificates visible to the agent, and allow these resources to be managed via KeySafe 5. The time taken for the agent to publish a CodeSafe 5 update message will increase by several seconds per CodeSafe 5 resource (machine or certificate) in the system. This means that in systems with many CodeSafe 5 machines/certificates present, KeySafe 5 will be slower to reflect local changes in the state of these resources.

7.1. Installation Steps

The KeySafe 5 agent is installed alongside an existing nShield Security World Software installation.

> The KeySafe 5 agent is a privileged client of the hardserver. For more

> information on privileged clients, see the nShield Security World Software documentation.

Ensure the system clock of the KeySafe 5 agent is synchronized with the central platform.

If you are upgrading an existing KeySafe 5 Agent install, see Agent Upgrade.

7.1.1. Install on Linux

1. Untar the KeySafe 5 agent install package to the root directory of the machine. The agent install package can be found in `keysafe5-agent` directory of the KeySafe 5 release package.

   This unpacks the KeySafe 5 agent binaries and associated scripts into the `/opt/nfast/` directory.

   ```
   sudo tar -C / -xf /path/to/keysafe5-1.5.0-Linux-keysafe5-agent.tar.gz
   ```

2. Ensure the `messagebus/tls` directory is in place.

   ```
   sudo mkdir -p /opt/nfast/keysafe5/conf/messagebus/tls
   ```

3. Configure this KeySafe 5 agent instance as described in Agent Configuration and Message Bus Authentication.
4. Run the install script:

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

The installer creates the following items, as required:

- A new configuration from the example configuration if one does not already exist.
- Either a SysV-style init script or systemd script for automatically starting and stopping the service.
- The `keysafe5d` user.

  This user is dedicated to running the `keysafe5-agent` service, and is a member of the `nfast` and `nfastadmin` groups.

It also sets the correct permissions on the `/opt/nfast/keysafe5/conf/messagebus/tls` directory.

The KeySafe 5 agent is not affected by the standard nShield `/opt/nfast/sbin/init.d-nci-`

pher script. To stop, start, or restart the KeySafe 5 agent you may either:

- Use /opt/nfast/scripts/init.d/keysafe5-agent, or
- Use your standard init system scripts, addressing the nc_keysafe5-agent service.

7.1.2. Install on Windows

The KeySafe 5 Agent requires the hardserver TCP ports be enabled. To do this, either:

- Run config-serverstartup.exe --port 9000 --privport 9001, or
- Edit the file (located at %NFAST_KMDATA%\config\config) and set nonpriv_port=9000 and priv_port=9001.

After enabling the hardserver TCP ports, you must restart the hardserver service.

If those ports are not available and different ports are set, then the environment variables NFAST_SERVER_PORT and NFAST_SERVER_PRIVPORT must also be set appropriately as described in the nShield documentation. They may be set globally in System Environment Variables, or only for this service by adding a Multi-String Value named Environment under Computer\HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\nShield KeySafe 5 Agent, and to *Value data* adding the lines NFAST_SERVER_PORT=port-number and NFAST_SERVER_PRIVPORT=port-number. You may need to restart the computer after adding the System Environment Variables.

Launch the keysafe5-agent.msi installer. The installer is in the keysafe5-agent directory of the KeySafe 5 release package.

After installing the files, the installer will do some configuring, not overwriting any existing configuration:

- Copy the example configuration.
- Generate a private key and its Certificate Signing Request.
- The KeySafe 5 Agent service will be created and started.

If the configuration did not complete then the KeySafe 5 Agent service will not start, and the steps described in Agent Configuration and Message Bus Authentication will need to be followed before restarting the service using Windows Service Manager.

7.2. Upgrade Steps

The following information may be useful when upgrading:

The TLS certificate does not need to be generated again, as the backed up directory contains it. However, it can be generated again if preferred.

If the agent config from 1.4 is configured with the RabbitMQ(AMQP) message bus it cannot be used in 1.5 as RabbitMQ(AMQP) is no longer supported.

Any existing agent configuration files that use the RabbitMQ(AMQP) message bus must be re-configured to use the NATS message bus. Information on configuration can be found here.

To update the KeySafe 5 Agent installed on a machine:

- Take a backup of the Agent config directory located at `%NFAST_DATA_HOME%/keysafe5/conf`.
- Uninstall the existing KeySafe 5 Agent as detailed in the *KeySafe 5 Installation Guide* for the currently installed version of the product.
- Install the new KeySafe 5 Agent as detailed in chapter KeySafe 5 Agent Installation.

7.3. Configuration Items

The KeySafe 5 agent configuration file is located at `%NFAST_DATA_HOME%/keysafe5/conf/config.yaml`.

The install contains an example configuration file at `%NFAST_DATA_HOME%/keysafe5/conf/config.yaml.example`. Make a copy of it at the same location and rename the copy to `%NFAST_DATA_HOME%/keysafe5/conf/config.yaml`.

Unless configured otherwise, `%NFAST_DATA_HOME%` is located at `/opt/nfast` on Linux and `%ProgramData%\nCipher` on Windows.

Configuration Key	Description	Example Value
`override_hostname`	Set the hostname for this agent. This appears as the Host resource name in KeySafe 5. The hostname is set by the operating system if not overridden here. This is not related to the hostname used by TLS authentication.	`hostname`
`logging.level`	Minimum severity level of log statements to output. Valid values: `trace`, `debug`, `info`, `warning`, `error`. The default is to output at `info` level and above.	`info`

Configuration Key	Description	Example Value
`logging.format`	Format of the log statements. Valid values: `json`, `logfmt`. The default is to output in `json` format.	`json`
`logging.file.enabled`	To enable log output to file, set to `true`. The default is to output to file (`true`).	`true`
`logging.file.path`	The absolute path of the file that logs should be written to. The default is `/opt/nfast/log/keysafe5-agent.log` on Linux and `%ProgramData%\nCipher\Log Files\KeySafe5-agent.log` on Windows.	`/opt/nfast/log/keysafe5-agent.log`
`logging.journal.enabled`	To enable logging to journal (Linux only), set to `true`. The default is `false`.	`true`
`message_bus.url`	The URL points to the message bus service with its IP address or DNS name, including the port number. IPv6 addresses must be in the form [host]:port. This parameter is required, there is no default.	`127.0.0.1:18084`
`message_bus.auth_type`	Authentication method for the message bus connection. Valid values: `none`, `tls`. The default is to use TLS authentication.	`tls`
`message_bus.disable_tls`	To disable Mutual TLS for the message bus connection, set to `true`. The default is to use Mutual TLS (`false`). Entrust recommends that this is always set to `false`.	`false`
`message_bus.min_protocol_version`	The minimum TLS protocol version that is used by the message bus connection of the keysafe5 agent. The default is `TLSV1_2`.	`TLSV1_2`
`message_bus.cipherSuites`	The available ciphersuites for the message bus connection of the keysafe5 agent. The defaults are `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`	`ECDHE-ECDSA-AES128-GCM-SHA256`
`kmdata_network_mount`	If any directory within the kmdata directory on this machine is mounted as a network share (e.g. NFS or SMB) this configuration should be set to true. For example, if this configuration is set to false, the agent will not be able to detect changes in the kmdata/local directory if the directory is an NFS mount. This setting updates the agent to use a method of detecting file modifications/additions/removals that works for network mounted directories.	`false`

Configuration Key	Description	Example Value
`kmdata_poll_interval`	The rate at which the agent polls the `kmdata` directory to look for changes. The format is as used for `update_interval`. This value is only used if `kmdata_network_mount` is `true`. The default is once a second.	`1s`
`update_interval`	The period of time between publishing data updates. The interval string is a sequence of decimal numbers, each with optional fraction and a unit suffix, such as "300ms", "1.5h" or "2h45m". Valid time units are "ns", "us" (or "µs"), "ms", "s", "m", "h". The default is once a minute.	`1m`
`max_update_message_response_time`	The maximum amount of time to allow the central plat form to pull update messages sent by this agent. Update messages published by this agent will expire after the lower of this time, or the configured update_interval. This setting impacts the freshness of the data processed by the central platform. The default is 1 minute.	`1m`
`health_interval`	The period of time between checking the underlying service health and attempting recovery if necessary. The format is as used for `update_interval`. The default is once a minute.	`1m`
`recovery_interval`	The recovery_interval specifies how often recovery of the connections to external services should be performed. The default is 5 seconds.	`5s`
`codesafe_update_interval`	The period of time between publishing CodeSafe 5 data updates. The format is as used for `update_interval`. The default is once every 5 minutes.	`3m`
`codesafe_cache_period`	The `codesafe_cache_period` specifies how often the CodeSafe 5 certificate cache will expire. The caching is performed to negate performance impacts of running unnecessary CodeSafe 5 certificate commands. Cache expiry may be performed earlier if approaching a time where a certificates validity status may change, that is, when it is approaching `NotBefore` or `NotAfter`. The cache is invalidated if there is a change in CodeSafe 5 certificates. The format is as used for `update_interval`. The default is 60 minutes.	`60m`

Configure the KeySafe 5 agent's message bus connection to use the same instance used by the central management platform that you want to connect to.

7.3.1. Message Bus authentication

You can configure the authentication method for the message bus connection as one of the following options:

- `none` No authentication.
- `tls` X.509 certificate authentication.

Entrust recommends `tls` as the message bus authentication method. Entrust recommends restricting access to files containing sensitive authentication details.

On Windows, the KeySafe 5 agent installer will create the required files with appropriate per missions.

7.3.1.1. TLS

The directory `%NFAST_DATA_HOME%/keysafe5/conf/messagebus/tls` is used to store the TLS key and certificates for the agent's connection to the message bus in the following files:

ca.crt	The CA certificate.
tls.key	The agent's private key.
tls.crt	A valid certificate of the key signed by the Certificate Authority.

Your certificates will need to adhere to X.509 v3, sometimes known as a multiple-domain certificates, or SAN certificates. The X.509 extension Subject Alternative Name (SAN) allows specifying multiple hostnames and IP addresses, and has replaced Common Name as the source of the hostname.

The username extracted from the TLS client certificate (`tls.crt`) is the certificate's Distinguished Name.

The extracted certificate username must contain the KeySafe 5 agent's hostname (set either by `override_hostname` in the agent configuration file, or defaults to the machine's hostname). If the username does not contain the KeySafe 5 agent's hostname then the agent will not start.

7.3.1.1.1. Generating a KeySafe 5 agent private key and TLS certificate

To generate a private key and certificate signing request (CSR) for a specific KeySafe 5 agent, use `%NFAST_HOME%/keysafe5/bin/ks5agenttls`.

1. Generate the agent’s private key

   On Linux ensure the directory `/opt/nfast/keysafe5/conf/messagebus/tls` exists, then:

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

   On Windows the command to write to `%NFAST_DATA_HOME%\keysafe5\conf\messagebus\tls\tls.key` is:

   ```
   %NFAST_HOME%\bin\ks5agenttls.exe -keypath=C:\ProgramData\nCipher\keysafe5\conf\messagebus\tls\tls.key
   -keygen
   ```

   This will generate an ECDSA P-521 private key and save it to the file pointed to by the `keypath` option. If `keypath` is not specified the file `tls.key` is saved to the current directory.

2. Generate the CSR

   On Linux:

   ```
   /opt/nfast/keysafe5/bin/ks5agenttls -keypath=/opt/nfast/keysafe5/conf/messagebus/tls/tls.key -csrgen
   CSR has been generated and saved to ks5_demohost.csr
   ```

   On Windows the command is:

   ```
   %NFAST_HOME%\bin\ks5agenttls.exe -keypath=C:\ProgramData\nCipher\keysafe5\conf\messagebus\tls\tls.key
   -csrgen
   ```

   This will generate a certificate signing request and save it to `ks5_<agent_hostname>.csr` (where `<agent_hostname>` is the value of `override_hostname` in the KeySafe 5 agent configuration file, if set, or the host machines host name). Alternatively, the CSR can be printed to the console, rather than saved to a file, by specifying `-csr-stdout`.

   The generated CSR requests a certificate that contains the agent hostname as the CommonName and as a DNS SubjectAlternativeName (SAN).

3. Create a TLS Certificate for this KeySafe 5 agent

   The CSR should be provided to a KeySafe 5 administrator who, in a secure location/environment, creates a message bus service client TLS certificate using the CA trusted by the message bus server.

If using the Service deployment of KeySafe 5, run `%NFAST_HOME%/bin/keysafe5-server-admin` on the machine running the KeySafe 5 Service deployment:

```
$ /opt/nfast/bin/keysafe5-server-admin sign ks5_demohost.csr 365
Successfully signed CSR ks5_demohost.csr for 365 days
Saved certificates tls.crt and ca.crt in the current directory
```

If using the demo Kubernetes deployment of KeySafe 5, run the `agentcert.sh` script that is shipped alongside the deployment scripts from the preserved directory in which the script was run.

```
./agentcert.sh ks5_demohost.csr 365
Successfully signed CSR ks5_demohost.csr for 365 days
Saved certificates ks5_demohost.crt and ca.crt in the current directory
```

If you get a message about a lack of permissions opening `/etc/rancher/k3s/k3s.yaml` you can set up `kubectl` access by running:

```
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
```

You may append the `export KUBECONFIG=${HOME}/.kube/config` to your shell's configuration file.

4. Configure the KeySafe 5 agent

   The resulting TLS certificate and accompanying CA certificate, along with the agent's private key should be stored within `%NFAST_DATA_HOME%/keysafe5/conf/messagebus/tls` in the following files:

   - `ca.crt` - The CA certificate.
   - `tls.key` - The agent's private key.
   - `tls.crt` - A valid certificate of the key signed by the Certificate Authority.

7.3.2. KeySafe 5 agent on nShield Connect 5c/XC

A KeySafe 5 agent is installed on the nShield Connect for nShield Connect images released with Security World v13.4 and later software. This agent allows an nShield Connect to be monitored and managed without, or in addition to, the Connect being enrolled to a nShield host machine (a machine with nShield Security World software installed) which also has a KeySafe 5 agent installed.

By default, the KeySafe 5 agent on the nShield Connect is disabled. It must be configured to communicate with the central KeySafe 5 platform, and enabled. The agent can only be configured via the nShield Connect serial console.

By default, a KeySafe 5 1.5.0 central platform deployment will only be able to communicate with version 1.5 or later KeySafe 5 Agents. If your Connect has a v1.4 or earlier KeySafe 5 Agent then you must enable Agent Compatibility Mode when configuring the KeySafe 5 central platform installation.

7.3.2.1. ks5agent command (only on serial console network-attached HSMs)

The KeySafe 5 agent is configured and managed on the Connect using the `ks5agent` Serial Console command.

```
(cli)help ks5agent

        Manage the KeySafe 5 agent

        USAGE
          ks5agent
          ks5agent enable
          ks5agent disable
          ks5agent version
          ks5agent logs [tail [linecount]]
          ks5agent cfg [message_bus.url=x.x.x.x:18084]
          ks5agent resetcfg
          ks5agent mbcsr
          ks5agent mbtls [ca.crt|tls.crt] [data]


        OPTIONS
          enable      Start the KeySafe 5 agent (setting will persist on reboot)
          disable     Stop the KeySafe 5 agent
          version     Show version information for the KeySafe 5 agent
          logs        Display the KeySafe 5 agent log file
          cfg         Configure the KeySafe 5 agent
          resetcfg    Restore the KeySafe 5 agent configuration file back to the default values for this Connect
          mbcsr     Generate a Certificate Signing request for creation of KeySafe 5 agent TLS certificate
          mbtls     Show/set the TLS certificates for the KeySafe 5 Agent message bus connection

        If no action is specified, the current status of the KeySafe 5 agent will be
        displayed.
```

7.3.2.1.1. ks5agent cfg

The agent configuration can be displayed and set using the `ks5agent cfg` command.

```
(cli)ks5agent cfg
override_hostname: nshield_module_AAAA-AAAA-AAAA
logging:
  level: info
  format: json
  file:
    enabled: false
    path: /opt/nfast/log/keysafe5-agent.log
message_bus:
  url: 127.0.0.1:18084
  auth_type: tls
  tls_username_location: SAN-DNS-Field0
  disable_tls: false
  minProtocolVersion: TLSV1_2
  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
kmdata_network_mount: false
kmdata_poll_interval: 1s
update_interval: 1m
max_update_message_response_time: 1m
health_interval: 1m
recovery_interval: 5s
codesafe_update_interval: 3m
codesafe_cache_period: 60m

(cli)ks5agent cfg message_bus.url=<IPADDRESS>:18084 update_interval=2m
override_hostname: nshield_module_AAAA-AAAA-AAAA
logging:
  level: info
  format: json
  file:
    enabled: false
    path: /opt/nfast/log/keysafe5-agent.log
message_bus:
  url: <IPADDRESS>:18084
  auth_type: tls
  tls_username_location: SAN-DNS-Field0
  disable_tls: false
  minProtocolVersion: TLSV1_2
  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
kmdata_network_mount: false
kmdata_poll_interval: 1s
update_interval: 2m
max_update_message_response_time: 1m
health_interval: 1m
recovery_interval: 5s
codesafe_update_interval: 3m
codesafe_cache_period: 60m
```

The agent configuration values for `override_hostname`, `logging.file` and `kmdata_network_mount` are fixed and can not be set on nShield Connect. The value for `override_hostname` will be set to `nshield_module_{esn}`.

Multiple configuration items may be set with a single command.

```
ks5agent cfg message_bus.url=0.0.0.0:18084 update_interval=5m
```

If no configuration update is provided, the contents of the KeySafe 5 agent config file are displayed.

To update a configuration item, use the format `key=value` using a `.` character for nested configuration items. Examples:

```
ks5agent cfg update_interval=5m
ks5agent cfg logging.level=debug
ks5agent cfg message_bus.url=0.0.0.0:18084
```

If the agent is currently running, it will be restarted to pick up the change in configuration.

By default, you can only set values for keys that already exist in the configuration file. To force setting a key that does not currently exist in the configuration file, specify `--force`.

```
ks5agent cfg newkey=value --force
```

Running the `ks5agent resetcfg` command will reset the agent configuration to the default configuration for this agent on the nShield Connect.

7.3.2.1.2. Message bus authentication for ks5agent

The message bus authentication method is configured using the `ks5agent cfg` command and setting the `message_bus.auth_type` configuration item.

```
ks5agent cfg message_bus.auth_type=tls
ks5agent cfg message_bus.auth_type=none
```

TLS

TLS certificate authentication is configured with the following workflow:

1. Generate a CSR for this agent using the `ks5agent mbcsr` Serial Console command on the nShield Connect.
2. Generate a TLS certificate using this CSR.
3. Store the TLS certificate for this agent and the CA certificate using the `ks5agent mbtls` Serial Console command on the nShield Connect. These certificates must be entered in base64 encoded format. To create suitable input on a Unix system you can run `base64 --wrap=0 tls.crt`.

```
(cli)ks5agent mbcsr
<output will contain the CSR>

# Obtain a TLS certificate for the above CSR

(cli)ks5agent mbtls ca.crt <base64encoded_data>
Saved ca.crt

(cli)ks5agent mbtls tls.crt <base64encoded_data>
Saved tls.crt
```

7.3.2.1.3. Status

To show the current status of the KeySafe 5 agent on the nShield Connect, run the `ks5agent` Serial Console command with no arguments.

```
(cli)ks5agent
KeySafe 5 agent is disabled
```

The agent can be enabled and disabled using the `ks5agent enable` and `ks5agent disable` commands. This setting will persist over reboots.

To identify the version of agent installed on the Connect, use the `ks5agent version` command.

```
(cli)ks5agent version
1.5.0-de64c594
```

7.3.2.2. Logging

The agent logs of the KeySafe 5 agent running on the nShield Connect may be obtained by using the `ks5agent logs` Serial Console command.

By default, this will print the entire contents of the agent log file to the console. To display just the last 10 lines of the log file, use `ks5agent logs tail`. To display the last `n` lines of the log file, use `ks5agent logs tail <n>` where `<n>` is the number of lines to display.

```
(cli)ks5agent logs
(cli)ks5agent logs tail
(cli)ks5agent logs tail 20
```

If the nShield Connect is configured to append logs to the RFS, or configured to send logs to a Remote Syslog server, then the KeySafe 5 agent logs will be sent with these logs. For more information about configuring logging on the Connect, see the *nShield Connect User Guide*.

7.3.3. KeySafe 5 agent on nShield Connect 5c 10G

A KeySafe 5 agent is installed on the nShield Connect 5c 10G for nShield Connect images released with Security World v14.0 and later software.

The agent must be configured to communicate with the central KeySafe 5 platform to allow configuration and management of the nShield Connect 5c 10G.

Initial agent configuration must be done via the nShield Connect serial console. Once the agent is connected to a KeySafe 5 central platform then the agent configuration may be updated via KeySafe 5.

This KeySafe 5 agent is for platform management of the nShield Connect 5c 10G and will create a HSM resource of HSM type Platform in KeySafe 5. It will allow you to perform man agement operations on the nShield Connect 5c 10G from within KeySafe 5 (including net-work configuration, syslog configuration and HSM tenancy configuration/creation). See the *KeySafe 5 User Guide*. for more details.

7.3.3.1. ks5agent command

The KeySafe 5 agent is configured and managed on the Connect using the `ks5agent` Serial Console command.

```
(cli)help ks5agent
usage: ks5agent [-h] {restart,version,cfg,resetcfg,log,mbcsr,mbtls} ...

Manage the Platform KeySafe 5 agent.

options:
  -h, --help            show this help message and exit

ks5agent subcommands:
  {restart,version,cfg,resetcfg,log,mbcsr,mbtls}
    restart             Restart the KeySafe 5 agent.
    version             Show version information for the KeySafe 5 agent.
    cfg                 Configure the KeySafe 5 agent, or display the current
                        configuration.
    resetcfg            Restore the KeySafe 5 agent configuration file to the
                        default values.
    log                 Display the most recent log messages from the KeySafe
                        5 agent.
    mbcsr               Generate a Certificate Signing Request for the KeySafe
                        5 agent TLS certificate.
    mbtls               Show/set the TLS certificates for the KeySafe 5 Agent
                        message bus connection.
```

7.3.3.1.1. ks5agent cfg

The agent configuration can be displayed and set using the `ks5agent cfg` command. Updat ing the configuration will prompt to restart the KeySafe 5 agent.

```
(cli)ks5agent cfg
logging.level=info
logging.format=json
logging.file.enabled=false
logging.file.path=/opt/nfast/log/keysafe5-agent.log
logging.journal.enabled=true
message_bus.url=127.0.0.1:18084
message_bus.auth_type=tls
message_bus.disable_tls=false
message_bus.minProtocolVersion=TLSV1_2
message_bus.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
kmdata_network_mount=false
kmdata_poll_interval=1s
update_interval=1m
max_update_message_response_time=1m
health_interval=1m
recovery_interval=5s
codesafe_update_interval=3m
codesafe_cache_period=60m
override_hostname=hsm_AAAA-AAAA-AAAA

(cli)ks5agent cfg message_bus.url=<IPADDRESS>:18084
logging.level=info
logging.format=json
logging.file.enabled=false
logging.file.path=/opt/nfast/log/keysafe5-agent.log
logging.journal.enabled=true
message_bus.url=<IPADDRESS>:18084
message_bus.auth_type=tls
message_bus.disable_tls=false
message_bus.minProtocolVersion=TLSV1_2
message_bus.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
kmdata_network_mount=false
kmdata_poll_interval=1s
update_interval=1m
max_update_message_response_time=1m
health_interval=1m
recovery_interval=5s
codesafe_update_interval=3m
codesafe_cache_period=60m
override_hostname=hsm_AAAA-AAAA-AAAA

Restart KeySafe 5 agent to apply new configuration (y|n): y
Restarting KeySafe 5 agent.
Success.
```

The following agent configuration values are fixed and can not be set: - override_hostname - logging.file.enabled - logging.file.path - logging.journal.enabled - kmdata_network_mount - kmdata_poll_interval

The value for `override_hostname` will be set to `hsm_{esn}`.

Multiple configuration items may be set with a single command.

```
ks5agent cfg message_bus.url=0.0.0.0:18084 update_interval=5m
```

If no configuration update is provided, the contents of the KeySafe 5 agent config file are displayed.

To update a configuration item, use the format `key=value` using a `.` character for nested configuration items. Examples:

```
ks5agent cfg update_interval=5m
ks5agent cfg logging.level=debug
ks5agent cfg message_bus.url=0.0.0.0:18084
```

Running the `ks5agent resetcfg` command will reset the agent configuration to the default configuration for this agent on the nShield Connect.

7.3.3.1.2. Message bus authentication for ks5agent

The message bus authentication method is configured using the `ks5agent cfg` command and setting the `message_bus.auth_type` configuration item.

```
ks5agent cfg message_bus.auth_type=tls
ks5agent cfg message_bus.auth_type=none
```

TLS

TLS certificate authentication is configured with the following workflow:

1. Generate a CSR for this agent using the `ks5agent mbcsr` Serial Console command on the nShield Connect.
2. Generate a TLS certificate using this CSR.
3. Store the TLS certificate for this agent and the CA certificate using the `ks5agent mbtls` Serial Console command on the nShield Connect. These certificates must be entered in base64 encoded format. To create suitable input on a Unix system you can run `base64 --wrap=0 tls.crt`.

```
(cli)ks5agent mbcsr
<output will contain the CSR>

# Obtain a TLS certificate for the above CSR

(cli)ks5agent mbtls ca.crt <base64encoded_data>
Saved Base64 encoded data to ca.crt.

(cli)ks5agent mbtls tls.crt <base64encoded_data>
Saved Base64 encoded data to tls.crt.
```

To identify the version of agent installed on the Connect, use the `ks5agent version` command.

```
(cli)ks5agent version
1.5.0-e4687903
```

7.3.3.2. Logging

The agent logs of the KeySafe 5 agent running on the nShield Connect may be obtained by using the `ks5agent log` Serial Console command.

By default, this will print the entire contents of the agent log file to the console. To display the last `n` lines of the log file, use `ks5agent log -n <n>` where `<n>` is the number of lines to display.

```
(cli)ks5agent logs
(cli)ks5agent logs -n 20
```

If the nShield Connect is configured to send logs to a Remote Syslog server, then the KeySafe 5 agent logs will be sent with these logs.

The agent logs are also available via the KeySafe 5 API or WebUI.

7.4. Certificate Details

The KeySafe 5 Agent uses certificates to secure communications to the Agent Communica tions Interface in the KeySafe 5 central platform.

7.4.1. Agent Communication Certificates

TLS certificates are used to authenticate the agent's connection to the Agent Communication interface and secure communication between the KeySafe 5 Agent and the KeySafe 5 central platform.

For an agent on a nShield client machine, these certificates are found in the `%NFAST_DATA_HOME%/keysafe5/conf/messagebus/tls` directory.

File Name	Description
tls.crt	The TLS certificate for KeySafe 5 agent to authenticate to the Agent Communication interface.
tls.key	The private key for the KeySafe 5 agent to authenticate to the Agent Communication interface.

File Name	Description
ca.crt	The CA certificate used to sign the Agent Communication Server certificates.

7.4.1.1. TLS Certificate DistinguishedName Requirements

The KeySafe 5 Agent Communications interface uses a TLS certificate's DistinguishedName to identify the certificate and restrict permissions for specific certificates.

7.4.1.1.1. Agent on KeySafe 5 client machine

For an agent on a nShield client machine, the DistinguishedName in the TLS certificate used for connection to the Agent Communications interface must contain the hostname of the machine that the agent is running on (or the value of `overrride_hostname` in Agent configuration, if this value is set). If the certificate's DistinguishedName does not contain the machine's hostname (or the value of `override_hostname`, if set) then the Agent will not start.

For an agent on a nShield client machine, the TLS certificate DistinguishedName may not contain the value `keysafe5-backend-services`. If it does, then the Agent will not start.

7.4.1.1.2. Agent on Connect HSM

For an agent on a nShield Connect HSM, the Agent configuration value for `override_hostname` will automatically be set to `nshield_module_{esn}` or `hsm_{esn}`, and this configuration is unable to be modified. The DistinguishedName in the TLS certificate used for connection to the Agent Communications interface must contain this value with the correct ESN for the Connect HSM.

The Certificate Signing Request (CSR) downloaded from the Connect Serial Console will contain the correct name in the request.

7.5. Backup Details

Entrust recommends that you back up the following files and directories as part of your routine nShield backup schedules.

Unless configured otherwise, `%NFAST_DATA_HOME%` is located at `/opt/nfast` on Linux and `%ProgramData%\nCipher` on Windows.

File / Directory	Contents
%NFAST_-DATA_HOME%/keysafe5/conf/con-fig.yaml	KeySafe 5 Agent configuration file.
%NFAST_-DATA_HOME%/keysafe5/conf/messagebus/tls	KeySafe 5 Agent TLS certificates for secure communication with Agent Communications Interface. See Agent Certificate Details.

7.6. Troubleshooting

If the agent fails to start, ensure that the configuration file is present at %NFAST_-DATA_HOME%/keysafe5/conf/config.yaml.

If the configuration file is present but the agent still fails to start, see below for instructions on accessing the log.

If you are using TLS, ensure that the private key and certificate files are present in %NFAST_-DATA_HOME%/keysafe5/conf/messagebus/tls.

7.6.1. Logging

7.6.1.1. Linux

The KeySafe 5 agent log file is located at /opt/nfast/log/keysafe5-agent.log, unless con-figured otherwise.

7.6.1.2. Windows

The KeySafe 5 agent log file is located at %NFAST_LOGDIR%\KeySafe5-agent.log, unless con-figured otherwise.

The KeySafe 5 Windows Service actions are emitted to the Windows event log under the nShieldKeySafe5 source identifier.

You can use the nshieldeventlog utility to extract these log entries and output them to the console or a text file.

```
nshieldeventlog.exe --source=nShieldKeySafe5
```

As required, specify the following parameters.

- `-c` | `--count`: The number of records read from the event log.

  The default is `10000`

- `-f` | `--file`: The output filename.

See the nShield Security World Software documentation for more information on the `nshieldeventlog` utility.

7.7. Uninstall Steps

7.7.1. KeySafe 5 agent

Before uninstalling the nShield KeySafe 5 agent, Entrust recommends that you back up any configuration files and certificates from the installation.

7.7.1.1. Linux

To remove the KeySafe 5 agent from a Linux host run the KeySafe 5 uninstaller:

```
sudo /opt/nfast/keysafe5/sbin/install -u
```

Then proceed to remove the following files and directories:

- `/opt/nfast/keysafe5/bin/ks5agenttls`
- `/opt/nfast/keysafe5/conf/config.yaml.example`
- `/opt/nfast/keysafe5/sbin/install`
- `/opt/nfast/lib/versions/keysafe5-agent-atv.txt`
- `/opt/nfast/sbin/keysafe5-agent`
- `/opt/nfast/scripts/install.d/12keysafe5-agent`
- `/opt/nfast/log/keysafe5-agent.log`

The current configuration, stored in `/opt/nfast/keysafe5/conf`, may also be removed.

The agent log file will be located in a different location if you have changed the default value of `logging.file.path` in the agent configuration file.

If required, you can also remove the `keysafe5d` user that was created as part of the installation.

7.7.1.2. Windows

To remove the KeySafe 5 agent from a Windows host:

1. Stop the KeySafe 5 agent service using **Windows Service Manager**.
2. Open the **Control Panel** and select **Programs and Features**.
3. Select the **nShield KeySafe 5 Agent** package.
4. Select **Uninstall** and follow the on-screen instructions.

To remove any configuration files, delete the `%NFAST_DATA_HOME%\keysafe5` directory and remove the log file located at `C:\ProgramData\nCipher\Log Files\KeySafe5-agent.log`

The agent log file will be located in a different location if you have changed the default value of `logging.file.path` in the agent configuration file.