nShield Security World (Multi-Tenancy)

nShield v14.1.1 Tenant User Guide

11 June 2026

Table of Contents

1. Introduction

The nShield Tenant User Guide provides useful information about your nShield VCMs. You should consult it before and while using a new VCM.

1.1. Read this guide if ...

Read this guide if you need to configure or manage an Entrust VCM that has been provisioned for you by your service provider.

1.1.1. Terminology

Term	Description
Service Provider	A person or organisation that manages VCMs for use by tenants. The service provider has physical access to the HSM.
Tenant	A person or organisation that makes use of a VCM to provide cryptographic services.
VCM	A cryptographic module implemented as a share of a physical HSM that provides all the cryptographic services of an HSM and is securely separated from any other VCMs implemented on the same physical HSM.

1.2. Security World Software

PCIe and USB HSMS

The hardserver software controls communication between applications and Entrust nShield product line HSMs, which may be installed locally or remotely. It runs as a daemon (**Linux**) or a service (**Windows**) on the host computer.

The Security World for nShield is a collection of programs and utilities, including the hardserver, supplied by Entrust to install and maintain your nShield security system. The Security World Software includes the following:

- The appropriate installer for the client platform
- The client hardserver
- A set of utilities for configuring the nShield HSM

1.2.1. Software architecture

The software, firmware, and utilities have version numbers and there is also a version number for the World which refers to the World data that is stored in encrypted form on the client computer, typically in the `opt/nfast/kmdata` (**Linux**) or `C:\ProgramData\nCipher\Key Management Data` (**Windows**) directory or on the RFS. This data includes information concerning the World itself and also concerning each key that was created within that World. The World version created is determined by the version numbers of the software and firmware used when it was first created, see nShield Security World v14.1.1 Management Guide.

The latest World version is version 3. You can query the version of the World loaded on your system by using the command `kmfile-dump`.

1.2.2. Default directories

The default locations for Security World Software and program data directories on English-language systems are summarized in the following table:

Directory name	Default path (Linux)	Environment variable (Windows)	Default path (Windows)
nShield Installation	`/opt/nfast/`	`NFAST_HOME`	`C:\Program Files\nCipher\nfast`
Key Management Data	`/opt/nfast/kmdata/`	`NFAST_KMDATA`	`C:\ProgramData\nCipher\Key Management Data`
Dynamic Feature Certificates	`/opt/nfast/femcerts/`	`NFAST_CERTDIR`	`C:\ProgramData\nCipher\Feature Certificates`
Static Feature Certificates	`/opt/nfast/kmdata/hsm-ESN/features`		`%NFAST_KMDATA%\hsm-ESN\features` (**network-attached HSMs**) `%NFAST_KMDATA\features` (**PCIe and USB HSMs**)
Log Files	`/opt/nfast/log`	`NFAST_LOGDIR`	`C:\ProgramData\nCipher\Log Files`
User Log Files	`/home/<user>/nshieldlogs`	`NFAST_USER_LOGDIR`	`C:\Users\<user>\nshieldlogs`
Remote Static Feature Certificates	`/opt/nfast/kmdata/hsm-ESN/features`		`%NFAST_KMDATA%\hsm-ESN\features` (**network-attached HSMs**) `%NFAST_KMDATA\features` (**PCIe and USB HSMs**)

Directory name	Default path (Linux)	Environment variable (Windows)	Default path (Windows)
Remote Dynamic Feature Certificates	`/opt/nfast/kmdata/hsm-ESN/features`		`%NFAST_KMDATA%\hsm-ESN\features` (**network-attached HSMs**) `%NFAST_KMDATA\features` (**PCIe and USB HSMs**)

By default, the Windows `C:\ProgramData\` directory is a hidden directory. To see this directory and its contents, you must enable the display of hidden files and directories in the View settings of the Folder Options.

Dynamic feature certificates must be stored in the directory stated above. The directory shown for static feature certificates is an example location. You can store those certificates in any directory and provide the appropriate path when using the Feature Enable Tool. However, you must not store static feature certificates in the dynamic features certificates directory.

On Windows, the absolute paths to the Security World Software installation directory and program data directories are stored in the indicated nShield environment variables at the time of installation. If you are unsure of the location of any of these directories, check the path set in the environment variable.

The instructions in this guide refer to the locations of the software installation and program data directories as follows:

- By name (for example, Key Management Data).
- **Linux**: By absolute path (for example, `/opt/nfast/kmdata`).
- **Windows**: By nShield environment variable names enclosed with percent signs (for example, `%NFAST_KMDATA%`).

If the software has been installed into a non-default location:

- **Linux**: Create a symbolic link from `/opt/nfast/` to the directory where the software is actually installed.
- **Windows**: Ensure that the associated nShield environment variables are re-set with the correct paths for your installation. For more information about creating symbolic links, see your operating system's documentation.

1.2.3. Utility help options

Unless noted, all the executable utilities provided in the `bin` subdirectory of your nShield installation have the following standard help options:

- `-h|--help` displays help for the utility
- `-v|--version` displays the version number of the utility
- `-u|--usage` displays a brief usage summary for the utility.

1.3. Setting the PATH for nShield utilities

It is recommended that the PATH environment variable be changed to include `opt/nfast/bin` (**Linux**) or `<%NFAST_HOME%\bin>`, which is usually `C:\Program Files\nCipher\nfast\bin` (**Windows**).

This is the directory in the nShield installation that contains the nShield command-line utilities and some DLLs.

This allows all nShield command-line utilities to run without typing the full path. For example, you can run `enquiry` instead of `<opt/nfast/bin/enquiry>` on Linux or `<%NFAST_HOME%\bin\enquiry>` on Windows.

`opt/nfast/bin` (**Linux**) or `<%NFAST_HOME%\bin>` (**Windows**) must be set in the PATH in order to use the OpenSSL module in the Python that is bundled with nShield.

The Python bundled with nShield is located in `opt/nfast/python3/bin` (**Linux**) or `%NFAST_HOME\python3\bin`, which is usually `C:\Program Files\nCipher\nfast\python3\bin` (**Windows**). If using the nShield Python, you may additionally want to add this directory to the PATH environment variable so that you can run the nShield python as just the python command. You may not want to do this if you are also using other Python installations on the same machine.

1.4. Further information

This guide forms one part of the information and support provided by Entrust.

If you have installed the Java Developer component, the Java Generic Stub classes, nCipherKM JCA/JCE provider classes, and Java Key Management classes are supplied with HTML documentation in standard `Javadoc` format, which is installed in the appropriate `nfast/java` directory when you install these classes.

1.5. Security advisories

If Entrust becomes aware of a security issue affecting nShield HSMs, Entrust will publish a security advisory to customers. The security advisory will describe the issue and provide rec ommended actions. In some circumstances the advisory may recommend you upgrade the nShield firmware and or image file. In this situation you will need to re-present a quorum of administrator smart cards to the HSM to reload a Security World. As such, deployment and maintenance of your HSMs should consider the procedures and actions required to upgrade devices in the field.

The Remote Administration feature supports remote firmware upgrade of nShield HSMs, and remote ACS card presentation.

We recommend that you monitor the Announcements & Security Notices section on Entrust nShield, https://trustedcare.entrust.com/, where any announcement of nShield Security Advisories will be made.

2. VCM Management

2.1. VCM management

2.1.1. VCM creation

VCMs are provisioned for you by your service provider.

To request a VCM you must first generate a request message, including any properties you wish to specify, see VCM properties, using the command hsmadmin vcm create and send the request to your service provider. If you wish to see the information in the request message you can do this using the command hsmadmin vcm inspect-request.

The request will include two public keys. One is the key that will be used to connect to the `sshadmin` service in your VCM and the other is the key that will be used to sign request messages sent to the service provider, known as the 'tenant request key', see Tenant request key.

These keys are automatically created if they do not already exist. If keys already exist they are reused in any subsequent requests.

The `sshadmin` private key will be stored in your host file system and you should keep this safe and backed-up according to your security procedures. See SSH Client Key Protection (nShield 5s HSMs).

If you already have other VCMs provisioned, the service provider will need to ensure that your new VCM is created on a different physical HSM. Therefore the service provider may request information about any existing VCMs that you may have provisioned. If you attempt to enroll two VCMs hosted on the same physical HSM, the second enrollment will fail. You must either unenroll the first VCM or contact the service provider to request that they create the second VCM on a different HSM.

Once the service provider has created your VCM they will send you a configuration file containing the information needed to setup communication to the VCM using the command hsmadmin vcm enroll.

The VCM created by the service provider may have been configured with different properties than you requested and if this is the case you should seek further clarification from the service provider.

You must issue the `hsmadmin vcm enroll` command from the same host machine as you issued the `hsmadmin vcm create` command.

2.1.2. VCM properties

A VCM can be configured with different properties. These properties can be specified either when the VCM is created or can be modified on an existing VCM as described in Modifying VCM properties. A VCM must be stopped in order to modify the properties.

2.1.2.1. autostart

This property determines whether the VCM will automatically start after a reboot of the HSM on which it is hosted. This property is mutually exclusive with the `restrict-startup` property.

2.1.2.2. restrict-startup

This property determines whether the service provider is able to start the VCM without authorization from the tenant. If this property is selected the tenant must generate a start request message using the command hsmadmin vcm start and send it to the service provider to authorize the starting of the VCM. The command can optionally specify a validity period for the request. This property is mutually exclusive with the `autostart` property.

2.1.2.3. serial-cardreader

This property determines whether the VCM has access to the serial cardreader port that is part of the physical HSM on which the VCM is hosted. Since the HSM is owned by the service provider and located within the service provider's premises this property would normally only be used where the tenant and service provider belong to the same organization.

Only one VCM can have access to the serial cardreader port on a given HSM at any one time. Therefore this property is normally granted on a temporary basis, except in situations where an HSM is expected to host only one tenant.

2.1.2.4. features

This property determines the optional features of the VCM as listed at Available optional features. For more information on how to manage VCM features see VCM features.

2.1.2.5. name

This property sets the name of the VCM as seen by the service provider in some command outputs. This name is not visible to the tenant and so this property is normally only useful when the tenant and service provider belong to the same organization.

2.1.3. Modifying VCM properties

To modify the properties of an existing VCM you must generate a request message using the command hsmadmin vcm setproperties and send the request to your service provider. If you wish to see the information in the request message you can do this using the command hsmadmin vcm inspect-request.

The command can optionally specify a validity period for the request. You can do this by using the `--request-validity` option. This option sets the validity period of the authorization. If the service provider does not modify the VCM properties within the validity period, the command will fail and you will need to generate a new request.

When using this option it is important to ensure that the clock on your machine is synchronized with the clock in the service provider's machine. You will need to contact the service provider to ensure this is the case.

2.1.3.1. Modifying autostart property

The autostart property is modified by using the command hsmadmin vcm setproperties with the `autostart` or 'no-autostart' option.

2.1.3.2. Modifying restrict-startup property

The restrict-startup property is modified by using the command hsmadmin vcm setproperties with the `restrict-startup` or 'no-restrict-startup' option.

2.1.3.3. Modifying serial-cardreader property

The serial-cardreader property is modified by using the command hsmadmin vcm setproperties with the `serial-cardreader` or 'no-serial-cardreader' option.

2.1.3.4. Modifying features property

The features property is modified by using the command hsmadmin vcm setproperties with

the `features` option and specifying the features requested. See VCM features for information on how to specify features.

2.1.3.5. Modifying name property

The name property is modified by using the command hsmadmin vcm setproperties with the `name` option and specifying a new name for the VCM.

2.1.4. Authorizing VCM start

If your VCM has been configured to require authorization to start it, see restrict-startup then you must create a message that authorizes the start and send it to the service provider whenever your VCM needs to be started.

You can create the message with the command hsmadmin vcm start. The command can optionally specify a validity period for the request. You can do this by using the `--request-validity` option. This option sets the validity period of the authorization. If the service provider does not start the VCM within the validity period the start command will fail and you will need to generate a new request.

When using this feature, it is important to ensure that the clock on your machine is synchronized with the clock on the service provider's machine. You will need to contact the service provider to ensure this is the case.

2.1.5. VCM keys

The client keys that secure communication with the services inside the VCM are the same as the client keys that secure communication with the platform services if you have any locally connected modules. The server keys will be unique for each locally connected module and each VCM. You can see the complete set of client and server keys by using the command hsmadmin keys show.

If you wish to change the client keys, you can do so using the command hsmadmin keys roll.

If you wish to backup the client keys you can do so with the command hsmadmin keys backup.

If you wish to restore the client keys from backup you can do so with the command hsmadmin keys restore. Once you have restored the keys you will need to manually enroll each

VCM using the command hsmadmin vcm enroll. You will need the configuration file for each VCM to be able to do this.

2.1.6. VCM logs

2.1.6.1. VCM system logs

The VCM has its own system log to record events happening inside the VCM. Handling is the same as for logs at platform level, see System logging (nShield 5 HSMs) except that the commands used are from hsmadmin vcm logs.

2.1.6.2. VCM audit logs

Audit logging is unchanged when operating inside a VCM; see Audit Logging.

The audit segment header in the audit logs, see audit segment header, contains the ESN of the HSM that produced the audit record. Because you can enroll only one VCM on a given HSM at a time, the ESN and timestamp uniquely identify a VCM.

If your VCM is deleted and you enroll another VCM hosted on the same HSM, then the new VCM's audit logs will have the same ESN in the audit segment header as the deleted VCM. However, the two logs can be distinguished by their timestamps.

The service provider cannot delete a VCM that has an unfinalized audit log unless they use the `--force` option. This means that, in normal operation, VCM audit logs should always be finalized before there is any chance of the same ESN being used for another VCM.

2.1.7. VCM deletion

VCMs are deleted by the service provider when you have told them that they are no longer needed, or in exceptional circumstances may be forcibly deleted by the service provider.

Before deleting a VCM that is using audit logging, the audit log should be finalized. See Disabling audit logging.

When a VCM has been deleted, you must remove references to that VCM from your host file system. If you do not do so, the VCM will show up as `failed` in an `enquiry` command and you may get issues whenever you restart your hardserver.

You can remove references to a VCM by using the command hsmadmin vcm unenroll and specifying the `uuid` of the VCM to delete. You can find the `uuid` of a VCM from the output

of an `enquiry` command or by looking in the config file that you received when the VCM was created and which you used when enrolling the VCM.

When a VCM is deleted it is good practice to also delete the config file since it is no longer of any use.

2.1.8. Tenant request key

Messages that you send to the service provider to manage your VCM are signed by the ten ant request key. If this key does not exist it is automatically created when you request your first VCM, see VCM creation. The keypair is stored in the `tenant` directory of `$NFAST`.

The public key is stored by the service provider and used to validate any messages sent to them. Messages that fail validation will be ignored. For this reason it is important to keep a backup of the keypair in case you delete your host-side installation, either accidentally or as part of an upgrade.

You can use standard commands for your operating system to copy the keypair to a safe location and copy it back when required.

If for any reason you wish to change the tenant request key you should first delete any VCMs created using it and then delete the keypair from the `tenant` directory. A new keypair will then be generated when you next create a VCM.

Once you have changed the tenant request key any requests made for VCMs that existed before the change will fail to validate. For this reason it is recommended to delete all VCMs before changing the tenant request key.

2.2. VCM features

2.2.1. Requesting VCM features

Features available within a VCM are provisioned for you by your service provider. For a list of the available features see Available optional features.

To request a feature be made available for your VCM you must generate a request message and send the request to your service provider.

If you are requesting a new VCM you can do this using the command hsmadmin vcm create and specifying the features you want in the `--features` option.

If you want to change the features for an existing VCM you can do this using the command hsmadmin vcm setproperties and specifying the features you want in the `--features` option.

The service provider will need to stop your VCM to apply the new features so you should put your VCM in maintenance mode as described at check and change the mode of operation. Once the service provider has applied the new features they will need to start the VCM. If you specified that the VCM requires authorization to start, see Restrict startup, you should create a start authorization message as described at Authorizing VCM start.

The `--features` option in the commands above specifies the features as a 32 bit word as described at Specification of features as a 32 bit word.

2.2.2. Built-in features

A number of features that were optional on earlier versions of firmware are now built in to the firmware and are always available for use regardless of whether they are requested or not. These are:

- StandardKM
- EllipticCurve
- ECCMQV
- AcceleratedECC
- PostQuantum

Specifying, or not specifying, these features in any commands will have no effect on your ability to make use of these features.

2.2.3. Viewing VCM features

You can view the features that are available in your VCM using the command `enquiry`.

An example `enquiry` output would be:

```
features enabled     ForeignTokenOpen RemoteShare GeneralSEE KISAAlgorithms StandardKM EllipticCurve ECCMQV
AcceleratedECC HSMSpeed2 PostQuantum
```

You can also view the features that are available in your VCM using the command `fet --show-all` but this command does not show the built-in `StandardKM` feature which has never been optional and has always been included in all firmware.

The `fet` command includes a number of options for reading and apply-

> ing feature certificates. This functionality is no longer applicable within a VCM. It is not possible to order any feature certificates that can be applied within a VCM.

3. Using the features of your VCM

3.1. Client software and module configuration: PCIe and USB HSMs

This chapter describes software and module configuration tasks that you can choose to per form after the initial installation of Security World Software and hardware.

You must determine whether particular configuration options are necessary or appropriate for your installation. The additional configuration options described in this chapter can be performed either before or after the creation of a Security World (as described in Create a new Security World) and an OCS (as described in Creating Operator Card Sets (OCSs)).

3.1.1. About user privileges

Cryptographic security does not depend on controlling user privileges or access but main-taining the integrity of your system from both deliberate or accidental acts can be enhanced by appropriate use of (OS) user privileges.

3.1.2. Set up client cooperation

You can allow an nShield HSM to automatically access the remote file system (RFS) belong ing to another nShield HSM and share the Security World and key data stored in the Key Management Data directory. Client hardware security modules that access data in this way are described as *cooperating clients*.

To configure client cooperation for other clients or hardware security modules that are not nShield HSMs, see Client cooperation.

3.1.2.1. Useful utilities

- anonkneti
- rfs-sync

3.1.2.2. Setting environmental variables

This section describes how to set Security World Software-specific environment variables. You can find detailed information about the environment variables used by Security World Software in Environment variables.

Linux

You can set Security World Software-specific environment variables in the file `/etc/nfast.conf`. This file is not created by the installation process: you must create it yourself. `/etc/nfast.conf` is executed by the start-up scripts of nShield HSM services as the root user. This file should only contain shell commands that can safely be run in this context. `/etc/nfast.conf` should be created with access permissions that allow only the root user to write to the file.

Ensure that all variables are exported as well as set.

Windows

You can set Security World Software-specific environment variables as follows:

1. Open the **System** dialog by clicking **System** in the control panel menu.
2. Select the **Advanced** tab and click the **Environment Variables** button.
3. To add a variable, click **New**. Alternatively, to edit an existing variable select an entry in the **System Variables** list and click **Edit**.
4. In the **Variable Name** field, type or edit the name of the environment variable (for example, **NFAST_HOME**).
5. In the **Variable Value** field, type or edit the value to use.
6. Click the **OK** button to set the value, and then click the **OK** button to close the dialog.
7. Open the **Administrative Tools** dialog by clicking the **Administrative Tools** icon in the Control Panel
8. Open the **Services** console by clicking the **Services** icon.
9. From the displayed list of services, select the **nFast Server** icon, and select **Restart the service**.

3.1.2.3. Logging and debugging

Network-attached HSMs: You can view logs generated by the nShield HSM and applications that use it on the unit front panel. Application log messages are handled on the client.

The Security World Software generates logging information that is configured through a set of four environment variables:

- `NFLOG_FILE`
- `NFLOG_SEVERITY`
- `NFLOG_DETAIL`

- `NFLOG_CATEGORIES`

If none of these logging environment variables are set, the default behavior is to log nothing, unless this is overridden by any individual library. If any of the four logging variables are set, all unset variables are given default values.

Detailed information about controlling logging information by means of these environment variables is supplied in Logging, debugging, and diagnostics.

Some components of the Security World Software generate separate debugging information which you can manage differently. On network-attached HSMs, debugging information for applications is handled on the client.

If you are setting up the unit or the client to develop software that uses it, you should configure debugging before commencing software development.

3.1.3. Configuring the hardserver

The hardserver handles secure transactions between the HSMs connected to the host computer and applications that run on the host computer. In addition, the hardserver, for example:

- Controls any Remote Operator slots that the HSM uses
- Loads any SEE (Secure Execution Engine) machines that are to run on the HSM
- Enables Remote Administration and provides the communication channel between the Remote Administration Service and the HSM

The hardserver can handle transactions for multiple HSMs. This does not require configuration of the hardserver. For more information, see Using multiple modules.

The hardserver must be configured to control:

- The way the hardserver communicates with remote HSMs
- The way the hardserver communicates with local HSMs
- The import and export of Remote Operator slots
- The loading of SEE machines on to the HSM when the hardserver starts up
- The number of Dynamic Slots available on the HSM
- The port used to connect to the Remote Administration Service
- Whether a Dynamic Slot needs to be exchanged with slot 0 of an HSM
- Timeout values for nShield Remote Administration Card presence assurance
- Configuring the audit logging destination.

The hardserver configuration file defines the configuration of the hardserver. By default, it is stored in `/opt/nfast/kmdata/config/` (**Linux**) or `%NFAST_KMDATA%\config` (**Windows**), and a default version of this file is created when the Security World Software is installed. See Overview of hardserver configuration file sections for an overview of the hardserver configuration file, and see nShield HSM configuration files for detailed information about the various options available through it.

In some previous releases of the Security World Software, hardserver configuration was controlled by environment variables. The use of these variables has been deprecated. If any of these environment variables are still set, they override the settings in the configuration file.

You must load the configuration file for the changes to the configuration to take effect.

To configure the hardserver, follow these steps:

1. Save a copy of the configuration file, `/opt/nfast/kmdata/config/` (**Linux**) or `%NFAST_KMDATA%\config\config` (**Windows**), so that the configuration can be restored if necessary.
2. Edit the configuration file to contain the required configuration. (See "Hardserver configuration files" for descriptions of the options in the configuration file.)
3. Run cfg-reread to load the new configuration.

   If you changed the server_startup section of the hardserver configuration file, you must restart the hardserver instead of running cfg-reread. For more information, see Stopping and restarting the hardserver.

4. Test that the hardserver is configured correctly by running the `enquiry` command-line utility.

   Check that an HSM with the correct characteristics appears in the output.

5. Test that the client has access to the Security World data by running the `nfkminfo` command-line utility.

   Check that an HSM with the correct ESN appears in the output and has the state `0x2 Usable`.

3.1.3.1. Overview of hardserver configuration file sections

See nShield HSM configuration files for a full explanation of the configuration files.

3.1.3.1.1. Configure remote HSM connections

You configure the hardserver's communications with remote HSMs in the `server_remotecomms` section of the hardserver configuration file. This section defines the port on which the hardserver listens for communications from remote HSMs. You need to edit this section only if the default port (9004) is not available.

For detailed descriptions of the options in this section, see server_remotecomms.

For information about configuring the Remote Operator feature (Remote Operator slots), as opposed to remote HSMs, see Remote Operator.

3.1.3.1.2. Hardserver settings

You configure the hardserver's settings in the `server_settings` section of the configuration file.

This section defines how connections and hardserver logging are handled. These settings can be changed while the hardserver is running.

For detailed descriptions of the options in this section, see server_settings.

3.1.3.1.3. Hardserver performance settings

You configure the hardserver performance settings in the server_performance section of the configuration file.

This section determines whether multi-threaded performance scaling is enabled or not. By default, scaling is not enabled. Any changes you make to the settings in this section do not take effect until after you restart the hardserver.

For detailed descriptions of the options in this section, see server_performance.

3.1.3.1.4. HSM settings

You configure the HSM's settings in the `module_settings` section of the configuration file.

This section defines the settings for the HSM that can be changed while the hardserver is running.

For detailed descriptions of the options in this section, see module_settings.

3.1.3.1.5. Hardserver start-up settings

You configure the hardserver's start-up settings in the `server_startup` section of the configuration file.

This section defines the sockets and ports used by the hardserver. You need to change this section only if the default ports for privileged or unprivileged connections (9000 and 9001) are not available.

Windows only

You should use the `nt_privpipe_users` option to define the name of the user who is allowed to carry out privileged operations, for example, using the `nopclearfail` utility. See nt_privpipe_users for more information.

For detailed descriptions of the options in this section, see server_startup.

3.1.3.1.6. SEE machines

You configure the hardserver to load SEE machines on start-up in the `load_seemachine` section of the configuration file. The SEE Activation feature must be enabled on the HSM, as described in Optional features.

This section defines the SEE machines and optional user data to be loaded, as well any other applications to be run in order to initialize the machine after it is loaded.

For detailed descriptions of the options in this section, see load_seemachine.

For information about SEE machines, see CodeSafe applications.

3.1.3.1.7. Remote Operator slots

You configure Remote Operator slots in the `slot_imports` and `slot_exports` sections of the configuration file. These sections define the slots that are imported to or exported from the HSM. This applies to the Remote Operator feature only.

For detailed descriptions of the options in these sections, see slot_imports and slot_exports.

The Remote Operator feature must be enabled on the HSM, as described in Optional features.

3.1.3.1.8. Remote file system

Each client's remote file system is defined separately in the `remote_file_system` section of

the configuration file with a list of HSMs that are allowed to access the file system on the given client. For information about setting up client cooperation, see Client cooperation.

The `remote_file_system` section is updated automatically when the `rfs-setup` utility is run. Do not edit the `remote_file_system` section manually.

As a reference, for detailed descriptions of the options in this section, see remote_file_system.

3.1.3.1.9. Audit logging

You configure the hardserver's audit logging in the `auditlog_settings` section of the configuration file.

This section defines the host IP address and port used as the destination for the syslog output of the audit logging capability. Optionally, the audit logging messages can be copied to the hardserver's log file.

For further details see Audit Logging.

The hardserver needs to be restarted for these settings to take effect.

3.1.3.2. Using multiple modules

The hardserver can communicate with multiple modules connected to the host. By default, the server accepts requests from applications and submits each request to the first available module. The server can share load across buses, which includes the ability to share load across more than one module.

If your application is multi-threaded, you can add additional modules and expect performance to increase proportionally until you reach the point where cryptography no longer forms a bottleneck in the system.

3.1.3.2.1. Identifying modules

Modules are identified in two ways:

- By serial number
- By `ModuleID`.

You can obtain the `ModuleID` 's and serial numbers of all your modules by running the `enquiry` command-line utility.

3.1.3.2.2. Electronic Serial Number (ESN)

The serial number is a unique 12-digit number that is permanently encoded into each module. Quote this number in any correspondence with Support.

ModuleID

The `ModuleID` is an integer assigned to the module by the server when it starts. The first module it finds is given a `ModuleID` of 1, the next is given a `ModuleID` of 2, and this pattern of assigning `ModuleID` numbers continues for additional modules.

The order in which buses are searched and the order of modules on a bus depends on the exact configuration of the host. If you add or remove a module, this can change the allocation of ModuleIDs to all the modules on your system.

You can use the `enquiry` command-line utility to identify the PCI bus and slot number associated with a module.

All commands sent to nShield modules require a `ModuleID`. Many Security World Software commands, including all acceleration-only commands, can be called with a `ModuleID` of 0. Such a call causes the hardserver to send the command to the first available module. If you purchased a developer kit, you can refer to the developer documentation for information about the commands that are available on nShield modules.

In general, the hardserver determines which modules can perform a given command. If no module contains all the objects that are referred to in a given command, the server returns an error status.

However, some key-management operations must be performed together on the same module. In such cases, your application must specify the `ModuleID`.

To be able to share OCSs and keys between modules, the modules must be in the same Security World.

3.1.3.2.3. Adding a module

If you have a module installed, you can add further modules without reinstalling the server software.

However, we recommend that you always upgrade to the latest server software and upgrade the firmware in existing modules to the latest firmware.

1. Install the module hardware.
2. (**Linux**) Run the script `/opt/nfast/sbin/install`.

3. Add the module to the Security World. Refer to Adding or restoring an HSM to the Security World.

3.1.3.2.4. Module fail-over

The Security World Software supports fail-over: if a module fails, its processing can be transferred automatically to another module provided the necessary keys have been loaded. Depending on the mode of failure, however, the underlying bus and operating system may not be able to recover and continue operating with the remaining devices.

To maximize uptime, we recommend that you fit any additional nShield modules for failover on a bus that is physically separate from that of the primary modules.

3.1.4. Stopping and restarting the hardserver

If necessary, you can stop the hardserver on the client, and where applicable the Remote Administration Service, by running the following command. On Windows, this must be a command window with administrative privileges.

Linux

```
/opt/nfast/sbin/init.d-ncipher stop
```

Windows

```
net stop "nfast server"
```

If the Remote Administration Service is running, you will be warned and given the option of continuing or not.

Similarly, you can start the hardserver on the client, and where applicable the Remote Administration Service, by running the following command. On Windows, this must be a command window with administrative privileges.

Linux

```
/opt/nfast/sbin/init.d-ncipher start
```

You can also restart the hardserver on the client, and where applicable the Remote Administration Service, by running the following command:

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

Windows

```
net start "nfast server"
net start "nfast Remote Administration Service"
```

On Windows, you can also stop, start, or restart the hardserver, and where applicable the Remote Administration Service, from the Windows Control Panel:

1. From the Windows Start menu, open the Windows Control Panel.
2. Double-click Administrative Tools.
3. Double-click Services.
4. Locate **nFast Server** or **nFast Remote Administration Service** in the list of services, and from the **Action** menu, select **Stop**, **Start**, or **Restart** as required.

The **nFast Remote Administration Service**, where applicable, is dependent on the **nFast Server** so should be started or restarted after the **nFast Server**.

3.2. ncoreapi modes of operation

This chapter describes the `ncoreapi` modes of operation:

- Modes of operation
- Check and change the mode of operation

3.2.1. Modes of operation

The status of `ncoreapi` can only be one of the following:

Status	Description
Starting up	The nShield 5s HSM is booting up and performing self tests. After all tests complete successfully, the HSM enters Operational mode.
Operational mode	The nShield 5s HSM is working and ready to perform cryptographic operations. An initialized HSM enters Operation mode automatically after it is powered up and all pre-tests are successfully completed. To enter Operational mode manually, see Check and change the mode of operation.
Emulated maintenance mode	The nShield 5s HSM is ready to receive maintenance commands, or is processing a maintenance command. The HSM remains in Emulated maintenance mode until you change mode manually, see Check and change the mode of operation.

Status	Description
Pre-initialization mode	The nShield 5s HSM is ready to receive initialization commands. For example, initialization commands to set the root-of-trust key (KNSO), to create a Security World, or load an existing Security World. To enter Pre-initialization mode, see Check and change the mode of operation.
Initialization mode	The nShield 5s HSM is processing an initialization command. After the command completes, the HSM will return to Pre-initialization mode.
Uninitialized mode	The nShield 5s HSM was booted with no root-of-trust key (KNSO) set. This typically happens after leaving a factory state, see [tenant:ncoreapi-modes-nshield5s:::factory-state]. To resolve this, switch to Pre-initialization mode, set the KNSO and reboot the HSM.
Error	The nShield 5s HSM is in an error state, see HSM status indicators and error codes (nShield 5s). No cryptographic operations can be performed until this error has been cleared.

3.2.2. Check and change the mode of operation

You must change the mode on the nShield 5s HSM to perform certain maintenance and configuration tasks. The nShield 5s HSM does not have a physical mode switch. Switch between modes using the `nopclearfail` utility.

When changing the mode, you should wait a few seconds before issuing subsequent commands. These commands might fail if issued before `nopclearfail` has completed.

Use the following commands to change the mode of an nShield 5s HSM:

Command	Resulting mode
`nopclearfail --maintenance |-M`	Emulated maintenance mode
`nopclearfail --operational | -O`	Operational
`nopclearfail --initialization | -I`	Pre-initialization

1. Run the `nopclearfail` command specifying the module number and the new mode.

   When finished, the system responds with `OK`. This message is not confirmation that the module has changed mode.

   ```
   nopclearfail --maintenance --module 1
   ```

```
Module 1, command ClearUnitEx: OK
```

2. Confirm the new mode of the module by running the `enquiry` command.

   The `mode` line of the `Module` section displays the current mode.

```
enquiry -m1
Module #1:
enquiry reply flags  none
enquiry reply level  Five
serial number        XXXX-XXXX-XXXX
mode                 Emulated maintenance mode. hsmadmin may be used to perform module management whilst in
this mode.
module type code     14
product name         NC5536E/NC5536N
device name          #1 Secure Shell nshield-XXXX-XXXX-XXXX.local
hardware status      OK
```

3.3. Administration of platform services (nShield 5 HSMs)

nShield 5s platform services are administered through the unified utility `hsmadmin`, which directs the command to the service that implements the command.

Some commands require elevated privileges by default because both the permissions and the protection settings have an impact on the usability of the keys by non-administrative users. Commands that create keys or modify configuration always require elevated privileges. Elevated privileges mean `root` on Linux, and the built-in local Administrators group (running in an elevated shell) on Windows. If a command requires elevated privileges, this is indicated in the command description.

You can modify the permissions and protection options on service keys to allow particular groups of users to execute commands that require the private key for a given service. See Permissions on SSH keys and Setting protection on SSH keys.

All platform services are administered through a unified utility called `hsmadmin`.

3.3.1. hsmadmin

The `hsmadmin` utility manages the administration of nShield HSMs using different subcommands.

```
hsmadmin <subcommand> [-h] [-v] [--no-reset]
```

You can use one of the following subcommands each time you run `hsmadmin`:

- factorystate
- status
- npkginfo
- upgrade
- reset
- enroll
- keys
- logs
- info
- settime
- gettime
- setminvsn
- getenvstats
- cs5
- select
- vcm
- fet
- setnetwork

You can use the following options with `hsmadmin`:

- Reset options:
  - `--no-reset`: Does not trigger an immediate reset of any HSM. If a reset is required you must reboot the host system.
- Help options:
  - `-h`, `--help`: Displays help for `hsmadmin`.
  - `-v`, `--version`: Displays the version number of the Security World Software

3.3.1.1. hsmadmin factorystate

This command requires `root` privileges on Linux and the privileges of the built-in local Administrators group on Windows.

Before running this command, place the unit in maintenance mode using `nopclearfail -M -m <MODULEID> -w`.

This command returns an HSM to the state it was in when it left the factory. This securely erases all user credentials and information. It resets the `sshadmin` SSH credential to the

default.

```
hsmadmin factorystate [-h] [--timeout <TIMEOUT>] [--esn <ESN>] [--verbose]
```

This command takes the following parameters:

Parameter	Description
`--timeout`	Time to wait for service response, in seconds. Default 30 seconds, minimum 3s, maximum 120s.
`--all`	Resets all modules without prompting for confirmation.
`--esn`	Resets specific modules to factory state. You need to add `--esn` before each ESN you include in the command, for example: `hsmadmin factorystate --esn 1A23-BC45-6789 --esn 9Z87-YX65-4321` If no ESNs are specified, the command resets all connected modules. The command will prompt you to confirm this action unless the --all parameter is specified.
`--verbose`	Prints verbose logs.

3.3.1.2. hsmadmin status

This command displays the ESN and currently loaded firmware version for discovered HSMs. It also displays whether the current image is a primary or a recovery image. When used with the `--json` option it displays primary firmware version, recovery firmware version, and uboot version.

```
hsmadmin status [-h] [--esn <ESN>] [--timeout <TIMEOUT>] [--verbose] [--json]
```

This command takes the following parameters:

Parameter	Description
`--verbose`	Prints verbose logs.
`--json`	Prints the HSM firmware version and image version in JSON format.

Parameter	Description
`--esn`	Displays information for specified HSMs. You need to add `--esn` before each ESN you include in the command, for example: `hsmadmin status --esn 1A23-BC45-6789 --esn 9Z87-YX65-4321` If you do not specify any ESNs, the command displays information for all connected HSMs.
`--timeout`	Time to wait for service response, in seconds. Default 30 seconds, minimum 3s, maximum 120s.

3.3.1.3. hsmadmin npkginfo

This command inspects the `npkg` file and displays the metadata.

```
hsmadmin npkginfo [--json] <NPKGFILE>
```

This command takes the following parameters:

Parameter	Description
`--json`	Prints metadata in JSON format.
`<NPKGFILE>`	Specifies the NPKG-format file to inspect.

3.3.1.4. hsmadmin upgrade

This command installs firmware packages in npkg format. The command can install both primary and recovery firmware.

The command requires `root` privileges on Linux and the privileges of the built-in local Administrators group on Windows.

Before running this command, place the module to be upgraded in maintenance mode using `nopclearfail -M -m <MODULEID> -w`. This requirement can be overridden by using the `--force` option.

In a multi-tenant system all VCMs running on the HSM to be upgraded must be stopped for this command to succeed. This cannot be overridden using the `--force` option.

```
hsmadmin upgrade [-h] --esn <ESN> [--timeout <TIMEOUT>] [--dry-run] [--force] [--verbose] [--json] <NPKGFILE>
```

This command takes the following parameters:

Parameter	Description
`--verbose`	Prints verbose logs.
`--json`	Prints metadata in JSON format.
`--dry-run`	Don't load the package, just validate it.
`--force`	Ignore warnings and force upgrade to proceed.
`--esn`	Specifies the HSMs in which to load the NPKG file. You need to add `--esn` before each ESN you include in the command, for example: `hsmadmin upgrade --esn 1A23-BC45-6789 --esn 9Z87-YX65-4321 <NPKGFILE>`
`--timeout`	Time to wait for service response, in seconds. Default 30 seconds, minimum 3s, maximum 120s.
`<NPKGFILE>`	Specifies the npkg file to load in to the HSMs.

3.3.1.5. hsmadmin reset

Before running this command, place the unit in maintenance mode using `nopclearfail -M -m <MODULEID> -w`. If you run the command while in operational mode, it creates a failed state and you will need to run `nopclearfail -r -m <MODULEID>` to correct it.

This command resets the nShield HSM.

```
hsmadmin reset [-h] [--esn <ESN>]
```

This command takes the following parameters:

Parameter	Description
`--esn`	Specifies the HSMs to reset. You need to add `--esn` before each ESN you include in the command, for example: `hsmadmin reset --esn 1A23-BC45-6789 --esn 9Z87-YX65-4321` If you do not specify any ESNs, all connected HSMs will be reset.

3.3.1.6. hsmadmin enroll

This command requires `root` privileges on Linux and the privileges of the built-in local Administrators group on Windows.

Before running this command, place the unit in maintenance mode using `nopclearfail -M -m <MODULEID> -w`.

This command configures the SSH keys for the nShield HSM.

```
hsmadmin enroll [--timeout <TIMEOUT>] [--verbose] [--sshadmin-key <SSHADMIN_KEY>]
```

This command takes the following parameters:

Parameter	Description
`--timeout`	Time to wait for service response, in seconds. Default 30 seconds, minimum 3s, maximum 120s.
`--verbose`	Prints verbose logs.
`--sshadmin-key`	Path to backup of `sshadmin` key to use if not present in the standard location.

3.3.1.7. hsmadmin keys

This command requires `root` privileges on Linux and the privileges of the built-in local Administrators group on Windows.

Before running this command, place the unit in maintenance mode using `nopclearfail -M -m <MODULEID> -w`.

This command is used to manage the SSH keys currently loaded on a module.

```
hsmadmin keys [--timeout <TIMEOUT>] <subcommand>
```

This command takes the following parameter:

Parameter	Description
`--timeout`	Time to wait for service response, in seconds. Default 30 seconds, minimum 3s, maximum 120s.

You can use one of the following subcommands with this command:

- show
- migrate

- roll
- backup
- restore
- remote-set
- remote-remove

3.3.1.7.1. hsmadmin keys show

This subcommand displays the public client and server keys used to communicate with the HSMs. For client keys, it also displays the time stamp held on the associated key file in the host file system.

```
hsmadmin keys show [--json] [--verbose]
```

This subcommand takes the following parameters:

Parameter	Description
`--json`	Prints output in JSON format.
`--verbose`	Prints verbose logs.

3.3.1.7.2. hsmadmin keys migrate

This subcommand changes the SSHAdmin client key on all connected modules to match a public key. The public key is derived from the private key specified in the subcommand.

```
hsmadmin keys migrate --privkeyfile <PRIVKEYFILE> [--json] [--verbose]
```

This subcommand takes the following parameters:

Parameter	Description
`--json`	Prints output in JSON format.
`--verbose`	Prints verbose logs.
`--privkeyfile`	Specifies the file containing the private key to be migrated to.

3.3.1.7.3. hsmadmin keys roll

This subcommand changes the client keys for all services.

See SSH Client Key Protection (nShield 5s HSMs) for information about protection options that can be set on keys during generation.

```
hsmadmin keys roll [--json] [--verbose]
```

This subcommand takes the following parameters:

Parameter	Description
`--json`	Prints output in JSON format.
`--verbose`	Prints verbose logs.

On Linux, the hardserver must be restarted in order to be able to use the new `ncoreapi` SSH client key after performing this operation, for example, with `/opt/nfast/sbin/init.d-ncipher restart`.

3.3.1.7.4. hsmadmin keys backup

This subcommand makes a backup of the private client key for the `sshadmin` service.

The backup key should be protected against unauthorized access. Refer to your security procedures for information on how to store the backup file.

```
hsmadmin keys backup [--passphrase] <FILE>
```

This subcommand takes the following parameters:

Parameter	Description
`--passphrase`, `-p`	Replace host key protection with passphrase protection.
`<FILE>`	Path to file in which to store backup.

If the `--passphrase` option is not supplied, then the existing `sshadmin` key file will be copied verbatim with whatever existing protections it has. By default, the `sshadmin` key is tied to the host machine and OS install, and will not be usable on another machine. A warning about this restriction to the local machine will be printed if the `--passphrase` option is omitted (add `--local` to indicate that this is the explicit choice in order to prevent the warning).

If the `--passphrase` option is used, then the `sshadmin` key will be loaded and re-encrypted using a user passphrase that must be supplied at the prompt. If the existing `sshadmin` key was also protected with a user passphrase (this is not the case by default), then there will

be a prompt for that key's passphrase too. The backup key will not be tied to the host machine in this case, and can be used to re-install the HSM on another machine.

On Linux, the backup file will be generated with owner and group matching the directory in which it is created, and readable by owner only.

3.3.1.7.5. hsmadmin keys restore

This subcommand restores the private client key for the `sshadmin` service from a backup file that has previously been created with the `hsmadmin keys backup` command.

Once the private client key for the `sshadmin` service has been successfully restored, this command will automatically configure all other SSH keys for the HSM.

```
hsmadmin keys restore <FILE>
```

This subcommand takes the following parameter:

Parameter	Description
`<FILE>`	Path to file previously created by `hsmadmin keys backup`

3.3.1.7.6. hsmadmin keys remote-set

This subcommand installs a specific SSH public key for remote access to one HSM service.

```
hsmadmin keys remote-set <SERVICE> <KEYTYPE> <KEYDATA>
```

This subcommand takes the following parameters:

Parameter	Description
`<SERVICE>`	HSM service to be accessed remotely
`<KEYTYPE>`	SSH public key type
`<KEYDATA>`	SSH public key

3.3.1.7.7. hsmadmin keys remote-remove

This subcommand removes a specific SSH public key that had previously been set for remote access and restores the local client key.

```
hsmadmin keys remote-remove <SERVICE>
```

This subcommand takes the following parameter:

Parameter	Description
`<SERVICE>`	HSM service from which to remove remote access

3.3.1.8. hsmadmin logs

This command manages the system logs of connected HSMs. These logs are separate from the `ncoreapi` logs. See Platform services (nShield 5 HSMs) for more information about platform services and `ncoreapi`.

For more information about system logs, see System logging (nShield 5 HSMs).

For more information about managing `ncoreapi` logs, see Audit Logging.

```
hsmadmin logs <subcommand>
```

You can use one of the following subcommands with this command:

- get
- clear
- export
- expire
- getkey

3.3.1.8.1. hsmadmin logs get

This subcommand retrieves logs from a connected HSM.

HSMs running firmware version 13.5 or later can produce logs in either a signed or unsigned format. This subcommand will retrieve unsigned logs. To retrieve logs in a signed format, use the `export` subcommand.

```
hsmadmin logs get [-h] [--verbose] [--timeout <TIMEOUT>] --esn <ESN> --log <LOG> [--json | --out <OUTFILE>]
```

This subcommand takes the following parameters:

Parameter	Description
`--timeout`	Time to wait for service response, in seconds. Default 30 seconds, minimum 3s, maximum 120s.

Parameter	Description
`--esn`	Specifies the HSM from which to retrieve logs. Only one ESN can be used in the command to retrieve the logs of one specific HSM.
`--json`	Prints output in JSON format.
`--out`	Write logs to file specified by OUTFILE
`--verbose`	Prints verbose logs.
`--log`	Selects log to be retrieved. Options are `system`, `init`

3.3.1.8.2. hsmadmin logs clear

This subcommand clears logs from connected HSMs.

The `system` log can only be cleared using this command on firmware versions earlier than 13.5. The `system` log on HSMs running firmware version 13.5 or later is cleared using the `expire` command. See Logging, debugging, and diagnostics for more information. The `init` log can be cleared on all firmware versions using this command.

Before running this command, place the unit in maintenance mode using `nopclearfail -M -m <MODULEID> -w`.

```
hsmadmin logs clear [-h] [--verbose] [--timeout <TIMEOUT>] [--esn <ESN] --log <LOG> [--json]
```

This subcommand takes the following parameters:

Parameter	Description
`--timeout`	Time to wait for service response, in seconds. Default 30 seconds, minimum 3s, maximum 120s.
`--esn`	Specifies the HSMs from which to clear logs. You need to add `--esn` before each ESN you include in the command, for example: `hsmadmin logs clear --esn 1A23-BC45-6789 --esn 9Z87-YX65-4321 --log <LOG>` If you do not specify any ESNs, logs will be cleared from all connected HSMs.
`--json`	Prints output in JSON format.
`--verbose`	Prints verbose logs.

Parameter	Description
`--log`	Selects log to be cleared. Options are `system`, `init`

3.3.1.8.3. hsmadmin logs export

This subcommand retrieves and validates signed logs from a connected HSM.

The directory used for storing the log files must exist before running this command.

```
hsmadmin logs export [-h] [--verbose] [--timeout <TIMEOUT>] [--esn <ESN>] [--saved] [--expire] [--json | --outdir
<OUTDIR>]
```

This subcommand takes the following parameters:

Parameter	Description
`--timeout`	Time to wait for service response, in seconds. Default 30 seconds, minimum 3s, maximum 120s.
`--esn`	Specifies the HSM from which to export logs.
`--json`	Prints metadata in JSON format.
`--outdir`	Write logs to directory specified by OUTDIR
`--verbose`	Prints verbose logs.
`--expire`	Expire the log after exporting it
`--saved`	If not expired, re-export a previously saved log

3.3.1.8.4. hsmadmin logs expire

This subcommand expires saved system logs from a connected HSM.

```
hsmadmin logs expire [-h] [--verbose] [--timeout <TIMEOUT>] [--esn <ESN>] --seq <SEQ_NO> [--json]
```

This subcommand takes the following parameters:

Parameter	Description
`--timeout`	Time to wait for service response, in seconds. Default 30 seconds, minimum 3s, maximum 120s.
`--esn`	Specifies the HSM from which to expire logs.

Parameter	Description
`--json`	Prints output in JSON format.
`--seq`	expire the log identified by `<SEQ_NO>`
`--verbose`	Prints verbose logs.

3.3.1.8.5. hsmadmin logs getkey

This subcommand retrieves the system log signing key from a connected HSM.

```
hsmadmin logs getkey [-h] [--verbose] [--timeout <TIMEOUT>] [--esn <ESN>] [--json | --out <OUTFILE>]
```

This subcommand takes the following parameters:

Parameter	Description
`--timeout`	Time to wait for service response, in seconds. Default 30 seconds, minimum 3s, maximum 120s.
`--esn`	Specifies the HSM from which to retrieve the log signing key
`--json`	Prints output in JSON format.
`--out`	Write key to file specified by `<OUTFILE>`.
`--verbose`	Prints verbose logs.

3.3.1.9. hsmadmin info

This command requires `root` privileges on Linux and the privileges of the built-in local Administrators group on Windows.

This command returns information that was loaded in the HSM during manufacturing. This information is persistent even after returning the HSM to factory state.

```
hsmadmin info [-h] [--timeout <TIMEOUT>] [--esn <ESN>] [--verbose] [--json]
```

This command takes the following parameters:

Parameter	Description
`--timeout`	Time to wait for service response, in seconds. Default 30 seconds, minimum 3s, maximum 120s.

Parameter	Description
`--esn`	Returns information for the HSM identified by <ESN>. You need to add `--esn` before each ESN you include in the command, for example: `hsmadmin info --esn 1A23-BC45-6789 --esn 9Z87-YX65-4321` If no ESNs are specified, the command returns information for all connected modules.
`--verbose`	Prints verbose logs.
`--json`	Prints output in JSON format.

3.3.1.10. hsmadmin settime

This command is used to synchronize the HSM system clock with the clock in the host PC.

See Setting the system clock for more information on managing the system clock.

This command requires `root` privileges on Linux and the privileges of the built-in local Administrators group on Windows.

To use this command without the `--adjust` parameter, the HSM must be in maintenance mode.

Setting the system date and time without the `--adjust` parameter automatically resets the HSM.

```
hsmadmin settime [-h] [--adjust <adjust>] [--timeout <TIMEOUT>] [--esn <ESN>] [--verbose]
```

This command takes the following parameters:

Parameter	Description
`--adjust`	Optional parameter. If specified an HSM System clock drift calibration is executed.
`--timeout`	Time to wait for service response, in seconds. Default 30 seconds, minimum 3s, maximum 120s.

Parameter	Description
`--esn`	Sets the system date and time of specific modules. You need to add `--esn` before each ESN you include in the command, for example: `hsmadmin settime --esn 1A23-BC45-6789 --esn 9Z87-YX65-4321` If no ESNs are specified, the command resets all connected modules. If the adjust parameter is specified, a module reset is not required.
`--verbose`	Prints verbose logs.

3.3.1.11. hsmadmin gettime

This command requires `root` privileges on Linux and the privileges of the built-in local Administrators group on Windows.

It returns the system date and time of the HSM.

```
hsmadmin gettime [-h] [--timeout <TIMEOUT>] [--esn <ESN>] [--verbose] [--json]
```

This command takes the following parameters:

Parameter	Description
`--timeout`	Time to wait for service response, in seconds. Default 30 seconds, minimum 3s, maximum 120s.
`--esn`	Returns information for the HSM identified by <ESN>. You need to add `--esn` before each ESN you include in the command, for example: `hsmadmin gettime --esn 1A23-BC45-6789 --esn 9Z87-YX65-4321` If no ESNs are specified, the command returns the HSM system date and time for all connected modules.
`--verbose`	Prints verbose logs.
`--json`	Prints output in JSON format.

3.3.1.12. hsmadmin setminvsn

This command requires `root` privileges on Linux and the privileges of the built-in local

Administrators group on Windows.

Before running this command, place the unit in maintenance mode using `nopclearfail -M -m <MODULEID> -w`.

This command sets the minimum VSN number of the firmware which the HSM will in the future accept as an upgrade.

```
hsmadmin setminvsn [-h] [--timeout <TIMEOUT>] [--esn <ESN>] [--verbose] [--json] <VSN>
```

This command takes the following parameters:

Parameter	Description
`--timeout`	Time to wait for service response, in seconds. Default 30 seconds, minimum 3s, maximum 120s.
`--esn`	Sets the minimum VSN on the HSM identified by <ESN>. You need to add `--esn` before each ESN you include in the command, for example: `hsmadmin setminvsn --esn 1A23-BC45-6789 --esn 9Z87-YX65-4321 2` If no ESNs are specified, the command sets the minimum VSN on all connected HSMs.
`--verbose`	Prints verbose logs.
`--json`	Prints output in JSON format.
`<VSN>`	The minimum VSN to set. Once this command is executed, the HSM will no longer accept a command to upgrade to a firmware with a VSN lower than <VSN>. The new minimum VSN cannot be lower than the HSM's current VSN, and cannot be higher than the VSN of the firmware currently installed on the HSM.

3.3.1.13. hsmadmin getenvstats

This command returns the environmental monitoring statistics of the HSM.

Environmental monitoring statistics available depend on the model of the HSM, the hardware revision and the version of the firmware installed on the HSM.

For the nShield5 with firmware version 13.3 the available statistics are:

uptime	The time since the HSM was last rebooted, in seconds.
current_time	The current system time of the HSM.
mem_total	Total amount of physical RAM, in kilobytes.
msp_temp	Temperature recorded by the MSP sensor, in degrees C.
cpu_temp	Temperature recorded by the CPU sensor, in degrees C.
crypto_co_proc_temp	Temperature recorded by the cryptographic co-processor sensor, in degrees C.
voltage_t1022_core	Voltage drawn by the T1022 core chip.
voltage_t1022_ifc_io	Voltage drawn by the T1022 IFC I/O chip.
voltage_t1022_serdes	Voltage drawn by the T1022 SERDES chip.
voltage_t1022_serdes_io	Voltage drawn by the T1022 SERDES I/O chip.
voltage_c292_serdes	Voltage drawn by the C292 SERDES chip.
voltage_fpga_serdes	Voltage drawn by the FPGA SERDES chip.
voltage_c292_serdes_io	Voltage drawn by the C292 SERDES I/O chip.
voltage_fpga_serdes_io	Voltage drawn by the FPGA SERDES I/O chip.
voltage_msp_avcc	MSP Analogue Vcc.
voltage_ddr4_io_access	Voltage drawn by the DDR4 I/O access chip.
voltage_ddr4_io	Voltage drawn by the DDR4 I/O chip.
voltage_battery	Voltage supplied by the on-board battery.
voltage_pci_bus	Voltage drawn by the PCI bus.
max_temp	Highest temperature recorded by any temperature sensor since statistics were reset.
min_temp	Lowest temperature recorded by any temperature sensor since statistics were reset.
ais31_preliminary_alarm_count	AIS31 (RNG) preliminary alarm count.
spi_retries	SPI protocol failure count.
sp_i2c_total_failures	MSP430 I2C total failures.
sp_i2c_slave_failures	MSP430 I2C slave failures.
sp_temp_failures	MSP430 temperature failures.
sp_voltage_failures	MSP430 voltage failures.
host_bus_exceptions	PCI0 (Host) NPE and PE error count.

crypto_bus_exceptions	PCI1 (Crypto) NPE error count.
sp_sensor_cmd_failures	Read security processor handshake line failure count.
nvm_free_space	Free space on user NVRAM.
nvm_wear_level	Wear level on user NVRAM.
nvm_worn_blocks	Worn block count on user NVRAM.
bios_code	Not used; always reports 'None'
dfs_throttling	Whether CPU performance is currently degraded due to excessive heat.

```
hsmadmin getenvstats [-h] [--timeout <TIMEOUT>] [--esn <ESN>] [--verbose] [--json]
```

This command takes the following parameters:

Parameter	Description
`--timeout`	Time to wait for service response, in seconds. Default 30 seconds, minimum 3s, maximum 120s.
`--esn`	Returns information for the HSM identified by <ESN>. You need to add `--esn` before each ESN you include in the command, for example: `hsmadmin getenvstats --esn 1A23-BC45-6789 --esn 9Z87-YX65-4321` If no ESNs are specified, the command returns the environmental monitoring statistics of all connected modules.
`--verbose`	Prints verbose logs.
`--json`	Prints output in JSON format.

3.3.1.14. hsmadmin cs5

CodeSafe 5 is not supported in multi-tenant systems (v14.1 firmware).

This command is used to manage some aspects of CodeSafe SEE machines running on the HSM.

See also csadmin for additional commands related to managing CodeSafe SEE machines.

```
hsmadmin cs5 <subcommand>
```

You can use the following subcommand with this command:

- stats

The following subcommands are only relevant to the nShield 5c. See CodeSafe setup for the nShield 5c for more detail about the subcommands.

- clientinfo
- genclientinfo
- enroll
- unenroll
- list

3.3.1.14.1. hsmadmin cs5 stats

This subcommand gets statistics from active SEE machines.

```
hsmadmin cs5 stats [--timeout TIMEOUT] [-u UUID] [--esn ESN] [--json]
```

This subcommand takes the following parameters:

Parameter	Description
`--timeout`	Time to wait for service response, in seconds. Default 30 seconds, minimum 3s, maximum 120s.
`--u`	UUID of SEE machine from which to obtain statistics. If no UUID is specified, statistics will be retrieved for all running SEE machines.
`--esn`	Returns statistics for the HSM identified by <ESN>. If no ESNs are specified, the command returns statistics for all connected modules.
`--json`	Prints output in JSON format.

3.3.1.15. hsmadmin select

This command is used to select configuration options that are not controlled by licenses.

This command can only be used when the unit is in maintenance mode. It requires `root` privileges on Linux and the privileges of the built-in local Administrators group on Windows.

```
hsmadmin select <option> [--esn ESN] [--json] [--timeout]
```

All select <option> commands support the following parameters:

Parameter	Description
--esn	Select this option on the HSM identified by <ESN>. If no ESNs are specified, the selection is applied to all connected modules.
--json	Prints output in JSON format. You can use the following options with this command:
--timeout	Time to wait for service response, in seconds. Default 30 seconds, minimum 3s, maximum 120s.

3.3.1.15.1. hsmadmin select acceleration

This option selects which algorithms will be accelerated.

```
hsmadmin select acceleration [--set ID | --show]
```

This option takes the following parameters:

Parameter	Description
--set	Set the accelerator to use. The module must be reset for the change to take effect.
--show	Show the available, and the currently selected accelerators.

3.3.1.16. hsmadmin vcm

This command allows the tenants to manage their VCMs.

```
hsmadmin vcm <subcommand>
```

You can use one of the following subcommands with this command:

- create
- start
- enroll

- unenroll
- setproperties
- inspect-request
- info
- getenvstats
- single-setup
- logs

3.3.1.16.1. hsmadmin vcm create

This subcommand generates a request to create a VCM. The request will be written to the file specified in the command. This file should be sent to your service provider.

The file is signed by a signing key unique to you. This key is automatically generated when you create your first request and will then be used for all subsequent requests.

```
hsmadmin vcm create [-h] [--verbose] --request-file <REQUEST_FILE> [--request-validity <REQUEST_VALIDITY>] [--
restrict-startup | --autostart] [--serial-cardreader] [--features] [--name <NAME>]
```

This subcommand takes the following parameters:

Parameter	Description
`--request-file`	Specifies the name of the file to which the creation request will be written
`--verbose`	Prints verbose logs.
`--request-validity`	The time period in seconds for which the request is valid
`--restrict-startup`	If this option is chosen, the service provider will be unable to start the VCM without a valid start request
`--autostart`	A VCM with this option will automatically start after a reboot of the HSM on which it is hosted. The service provider may choose to ignore this request when creating the VCM if, for example, more tenants request this option than the service provider has a license for.
`--serial-cardreader`	A VCM with this option will have access to the serial card reader attached to the HSM. Only one VCM can have access to the card reader at any one time so the service provider may choose to ignore this request.

Parameter	Description
`--features`	The ncoreapi features you would like your VCM to have, expressed as a 32 bit hexadecimal word. The service provider may choose to provision a different set of features
`--name`	The name you would like the VCM to have. This name is only visible to the service provider so this option is mainly for use where the service provider is part of your own organisation

3.3.1.16.2. hsmadmin vcm start

This subcommand generates an authorization to start a VCM. This is needed if the VCM was created with the `--restrict-startup` option. The subcommand will write the authorization to a file specified in the command. This file must be sent to the service provider. There is an option to specify the validity period for the authorization and if this is selected the service provider must action the request before the end of the validity period and if not a new authorization must be generated.

```
hsmadmin vcm start [-h] --request-file <REQUEST_FILE> [--request-validity <REQUEST_VALIDITY>] [--uuid <UUID>]
```

This subcommand takes the following parameters:

Parameter	Description
`--request-file`	Specifies the name of the file to which the start authorization will be written
`--request-validity`	The time period in seconds for which the authorization is valid
`--uuid`	The UUID of the VCM for which the request is valid. A VCMs UUID is contained in the configuration file that was received from the service provider when the VCM was created and can also be seen in the `enquiry` command as part of the `device name`.

3.3.1.16.3. hsmadmin vcm enroll

This subcommand enrolls a VCM using the configuration file received from the service provider.

```
hsmadmin vcm enroll [-h] [--verbose] --config <CONFIG_FILE> [--no-service-restart]
```

This subcommand takes the following parameters:

Parameter	Description
`--config`	Specifies the configuration file that was received from the service provider
`--no-service-restart`	If this option is specified the hardserver and other services will not be restarted during enrollment
`--verbose`	Prints verbose logs.

3.3.1.16.4. hsmadmin vcm unenroll

This subcommand removes details of an enrolled VCM from the host file system.

```
hsmadmin vcm unenroll [-h] [--no-service-restart] [--verbose] uuid
```

This subcommand takes the following parameters:

Parameter	Description
`uuid`	Specifies the uuid of the VCM to unenroll
`--no-service-restart`	If this option is specified the hardserver and other services will not be restarted during enrollment
`--verbose`	Prints verbose logs.

3.3.1.16.5. hsmadmin vcm setproperties

This subcommand generates a request to change the properties of a VCM. The request is written to a file specified in the command and should be sent to the service provider. There is an option to specify the validity period for the request and if this is selected the service provider must action the request before the end of the validity period and if not a new request must be generated.

```
hsmadmin vcm setproperties [-h] --request-file <REQUEST_FILE> [--request-validity <REQUEST_VALIDITY>] [--uuid
<UUID>] [--restrict-startup | --no-restrict-startup] [--autostart | --no-autostart] [--serial-cardreader | --no
-serial-cardreader] [--features <FEATURES>] [--name <NAME>]
```

This subcommand takes the following parameters:

Parameter	Description
`--request-file`	Specifies the name of the file to which the request will be written
`--request-validity`	The time period in seconds for which the request is valid

Parameter	Description
`--uuid`	The UUID of the VCM for which the request is valid. A VCMs UUID is contained in the configuration file that was received from the service provider when the VCM was created and can also be seen in the `enquiry` command as part of the `device name`.
`--restrict-startup`	Enable the restrict-startup property
`--no-restrict-startup`	Disable the restrict-startup property
`--autostart`	Enable the autostart property
`--no-autostart`	Disable the autostart property
`--serial-cardreader`	Enable the serial-cardreader property
`--no-serial-cardreader`	Disable the serial-cardreader property
`--features`	Modify the VCM features. The new value is expressed as a 32 bit hexadecimal word.
`--name`	Modify the VCM name

3.3.1.16.6. hsmadmin vcm inspect-request

This subcommand displays the contents of a request or authorization file.

```
hsmadmin vcm inspect-request [--verify] [--key-file <KEY_FILE>] <REQUEST_FILE>
```

This subcommand takes the following parameters:

Parameter	Description
`--verify`	Verify the request signature
`--key-file`	The public key file for verifying the request signature

3.3.1.16.7. hsmadmin vcm info

This subcommand retrieves information about the HSM on which the VCM is hosted.

```
hsmadmin vcm info [-h] [--timeout <TIMEOUT>] [--uuid <UUID>] [--json]
```

This subcommand takes the following parameters:

Parameter	Description
`--timeout`	Time to wait for service response, in seconds. Default 30 seconds, minimum 3s, maximum 120s.
`--uuid`	Specifies the UUID of the VCM to retrieve info for. If omitted, information will be retrieved for all enrolled VCMs.
`--json`	Prints output in JSON format.

3.3.1.16.8. hsmadmin vcm getenvstats

This subcommand retrieves some statistics for a VCM, including the system time.

Times are reported in seconds using 'Unix time', also known as 'Epoch time'. There are readily available converters that can convert Unix time to other time formats.

```
hsmadmin vcm getenvstats [-h] [--verbose] [--timeout <TIMEOUT>] [--uuid <UUID>] [--json]
```

This subcommand takes the following parameters:

Parameter	Description
`--timeout`	Time to wait for service response, in seconds. Default 30 seconds, minimum 3s, maximum 120s.
`--uuid`	Specifies the UUID of the VCM from which to retrieve statistics.
`--json`	Prints output in JSON format.
`--verbose`	Prints verbose logs.

3.3.1.16.9. hsmadmin vcm single-setup

This subcommand will automatically create, start and enroll a single VCM hosted on the same machine as the HSM host.

```
hsmadmin vcm single-setup [-h] [--verbose] [--timeout <TIMEOUT>] [--esn <ESN>] [--json] [--name <NAME>]
```

This subcommand takes the following parameters:

Parameter	Description
`--timeout`	Time to wait for service response, in seconds. Default 30 seconds, minimum 3s, maximum 120s.
`--esn`	Specifies the HSM on which to create the VCM

Parameter	Description
`--json`	Prints output in JSON format.
`--name`	The name of the VCM which will be created
`--verbose`	Prints verbose logs.

3.3.1.16.10. hsmadmin vcm logs

This command manages the VCM system logs.

```
hsmadmin vcm logs <subcommand>
```

You can use one of the following subcommands with this command:

- export
- expire
- getkey

3.3.1.16.11. hsmadmin vcm logs export

This subcommand retrieves and validates signed logs from a connected VCM.

The directory used for storing the log files must exist before running this command.

```
hsmadmin vcm logs export [-h] [--verbose] [--timeout <TIMEOUT>] --uuid <UUID> [--saved] [--expire] [--json |
--outdir <OUTDIR>]
```

This subcommand takes the following parameters:

Parameter	Description
`--timeout`	Time to wait for service response, in seconds. Default 30 seconds, minimum 3s, maximum 120s.
`--uuid`	Specifies the uuid of the VCM from which to export logs.
`--json`	Prints metadata in JSON format.
`--outdir`	Write logs to directory specified by OUTDIR
`--verbose`	Prints verbose logs.
`--expire`	Expire the log after exporting it

Parameter	Description
--saved	If not expired, re-export a previously saved log

3.3.1.16.12. hsmadmin vcm logs expire

This subcommand expires saved system logs from a connected VCM.

```
hsmadmin vcm logs expire [-h] [--verbose] [--timeout <TIMEOUT>] --uuid <UUID> --seq <SEQ_NO> [--json]
```

This subcommand takes the following parameters:

Parameter	Description
--timeout	Time to wait for service response, in seconds. Default 30 seconds, minimum 3s, maximum 120s.
--esn	Specifies the uuid of the VCM from which to expire logs.
--json	Prints output in JSON format.
--seq	Expire the log identified by <SEQ_NO>
--verbose	Prints verbose logs.

3.3.1.16.13. hsmadmin vcm logs getkey

This subcommand retrieves the system log signing key from a connected VCM.

```
hsmadmin vcm logs getkey [-h] [--verbose] [--timeout <TIMEOUT>] --uuid <UUID> [--json | --out <OUTFILE>]
```

This subcommand takes the following parameters:

Parameter	Description
--timeout	Time to wait for service response, in seconds. Default 30 seconds, minimum 3s, maximum 120s.
--esn	Specifies the uuid of the VCM from which to retrieve the log signing key
--json	Prints output in JSON format.
--out	Write key to file specified by <OUTFILE>.
--verbose	Prints verbose logs.

3.3.1.17. hsmadmin fet

This subcommand applies a feature licence to the HSM.

```
hsmadmin fet [--timeout TIMEOUT] [--verbose] [--esn ESN] [--json] <FILENAME>
```

This subcommand takes the following parameters:

Parameter	Description
`--timeout`	Time to wait for service response, in seconds. Default 30 seconds, minimum 3s, maximum 120s.
`--esn`	ESN of the HSM on which to apply the license This must match the ESN contained in the license.
`--json`	Prints output in JSON format.
`FILENAME`	Certificate file containing licence to be applied

3.3.1.18. hsmadmin setnetwork

This command allows you to specify the subnetwork for this HSM on which VCMs will be created.

After applying this command, you must reset your HSM using hsadmin reset for the settings to take effect.

```
hsmadmin setnetwork <subcommand>
```

You can use one of the following subcommands with this command:

- default
- ipv4static
- ipv6static

3.3.1.18.1. hsmadmin setnetwork default

This subcommand sets the network to use IPv6 link local addresses, which is the default setting. The main use of this is where all tenants are on the same machine as the HSM. For example when the HSM is used for single tenant operation or if multiple tenants are hosted in containers on the same machine.

```
hsmadmin setnetwork default [-h]
```

This subcommand takes no parameters.

3.3.1.18.2. hsmadmin vcm setnetwork ipv4static

This subcommand sets the IPv4 subnetwork on which VCMs will be created.

```
hsmadmin setnetwork ipv4static --address <ADDRESS> --gateway <GATEWAY>
```

This subcommand takes the following parameters:

Parameter	Description
address	IPv4 address and network mask in CIDR format
gateway	IPv4 address of network gateway

3.3.1.18.3. hsmadmin vcm setnetwork ipv6static

This subcommand sets the IPv6 subnetwork on which VCMs will be created.

```
hsmadmin setnetwork ipv6static --address <ADDRESS> --gateway <GATEWAY>
```

This subcommand takes the following parameters:

Parameter	Description
address	IPv6 address and network mask in CIDR format
gateway	IPv6 address of network gateway

3.4. Using nShield commands from PowerShell

PowerShell is a powerful console tool for scripting operations on Windows. nShield applications can be run from PowerShell, locally or remotely, interactively or non-interactively (batch mode). nShield library code implements a set of commands for reading text and passphrases.

3.4.1. Install and configure PowerShell

1. Install PowerShell, see https://docs.microsoft.com/en-us/powershell/.
2. Ensure that executing PowerShell scripts is enabled in the system.

Script execution is not enabled by default on Windows clients. The default permissions usually allow script execution on Windows Server operating systems, but it may be nec essary to enable this in a custom Windows Server configuration.

Open PowerShell and set the signing property and scope of script execution.

The support files for running nShield commands from PowerShell are Authenticode-signed, so the execution can be restricted to only signed scripts:

```
Set-ExecutionPolicy -ExecutionPolicy AllSigned
```

If unsigned PowerShell scripts are to be executed, you may want to relax this so that locally created scripts can be run without signing:

```
Set-ExecutionPolicy -ExecutionPolicy RemoteSigned
```

The above commands can be run with the additional parameter `-Scope CurrentUser` to restrict the changes to the currently logged-in user. For example to permit the current user to run locally-created scripts, run:

```
Set-ExecutionPolicy -ExecutionPolicy RemoteSigned -Scope CurrentUser
```

3. By default, PowerShell commands are created for all executables in `$env:NFAST_HOME\bin`. If there are additional directories containing nShield executables that you wish to include in the nShield PowerShell module support, you can specify those directories with a semicolon separated list of paths in the `NC_PS_ADDITIONAL_DIRECTORIES` environment variable.

```
$env:NC_PS_ADDITIONAL_DIRECTORIES="C:\Path1;C:\Path2"
```

4. To load support for nShield commands in PowerShell, import the `nShieldTools.psd1` module located in `$env:NFAST_HOME\bin`.

```
Import-Module 'C:\Program Files\nCipher\nfast\bin\nShieldTools.psd1'
```

The support module is installed in the active shell.

5. To load the `nShieldTools.psd1` module automatically every time PowerShell is opened, you can add it to your PowerShell profile with the `Add-nShieldToProfile` command, included in the `nShieldTools.psd1` module.

Parameter	Description
`AllHosts`	(Recommended) Profile is available for all PowerShell hosts, for example both ConsoleHost and ISE, rather just the currently running host.
`AllUsers`	Profile is available for all users on the local machine rather than just the cur rent user.

Example:

```
Add-nShieldToProfile -AllHosts
```

3.4.2. Calling nShield commands at the PowerShell prompt

nShield commands can be called with their usual names from PowerShell, for example `enquiry`, `cardpp`, `generatekey`, or `new-world`. Aliases for the `.exe` variants, for example `new-world.exe`, are also registered so that the nShield executables in the PATH are only called with PowerShell support.

Do not call the executables directly (for example, do not call & 'C:\Program Files\nCipher\nfast\bin\new-world.exe') because this will not enable the PowerShell support.

Command-line parameters to nShield PowerShell commands can be provided in the same way as the corresponding command under regular Windows consoles. If you can run `cardpp --check` in a regular Windows console, you can run `cardpp --check` in PowerShell.

The global variable `$LASTEXITCODE` of PowerShell contains the exit code (0 for success) immediately after execution of the nShield command.

3.4.3. PowerShell modes: interactive and batch

nShield commands can be run in either interactive or batch (non-interactive) mode.

In the default **interactive mode**, the output is displayed incrementally on the screen in the PowerShell host user interface. UI prompts are displayed when the command attempts to read user input, for example passphrases or confirmations. Output is not written to a Power Shell pipeline in this mode.

Batch mode is intended for automation. Commands can be run from a script and an output can be redirected to a file. Batch mode does not prompt for input in the host UI. Input can be supplied programmatically with a PowerShell input pipeline. If input is needed but no suit able pipeline was supplied, the command fails rather than stall program execution to wait for user interaction. Standard output and standard error text printed by the underlying

nShield program is written to output and error pipelines that can be redirected or piped. If the program fails, that is, it returns a non-zero exit code, the error code is also thrown as an exception that appears in the error pipeline. In batch mode, output and error texts are visible only at the very end of execution, because such texts are objects that the command returns to the pipeline instead of writing them incrementally to the host UI.

If the mode is not explicitly set, PowerShell normally defaults to interactive mode. However, if any input pipeline objects are supplied, PowerShell defaults to batch mode. You can therefore switch to batch mode without setting it explicitly simply by piping `$null`:

```
$null | enquiry > enquiry.txt
```

You can change the default mode to batch mode by setting the `NC_PS_INTERACTIVE` environment variable:

```
$env:NC_PS_INTERACTIVE=0
```

Mode change commands provided in the `nShieldTools.psd1` module:

Command	Notes
`Set-nShieldBatchMode`	
`Set-nShieldInteractiveMode`	
`Reset-nShieldCommandMode`	Can be used to restore the default PowerShell behavior based on presence or absence of pipeline input.

To restrict a setting to a particular PowerShell scope, you can use the PowerShell variable `$nShieldInteractiveCommandMode`, which can be set to `$True` or `$False`.

3.4.4. Input pipelines

In both interactive and batch mode, nShield commands support input pipelines with the PowerShell pipe ("|") syntax. The input pipeline can be used to automate the execution of nShield commands that would otherwise have to prompt for user input. For example, a passphrase check on an OCS card can be performed automatically by executing the following command:

```
Set-nShieldBatchMode
$passphrase | cardpp --check
```

`$passphrase` is a variable in the command or script, and contains the card's passphrase.

Multiple values can be supplied to provide the input to successive prompts. For example, the `generatekey` command can be automated to provide passphrases for operator cards, softcards, or administrator cards. In the following example, the passphrase variables are passed to the input pipeline, and the remaining key generation parameters are passed on the command-line:

```
PS C:\temp\test> $acs_passphrase, $softcard_passphrase | generatekey --batch pkcs11 protect=softcard
softcard=mysoftcard plainname=mykeyname nvram=yes
key generation parameters:
operation    Operation to perform              generate
application  Application                       pkcs11
module       Module to use                     1
protect      Protected by                      softcard
slot         Slot to read cards from           0
softcard     Soft card to protect key          mysoftcard
recovery     Key recovery                      yes
verify       Verify security of key            yes
type         Key type                          RSA
size         Key size                          2048
pubexp       Public exponent for RSA key (hex)
logkeyusage  Log key usage                     no
plainname    Key name                          mykeyname
nvram        Blob in NVRAM (needs ACS)         yes

Load Admin Card (for KNV):
Module 1 slot 0: Admin Card #1
Module 1 slot 0: Enter passphrase:
Module 1 slot 0:- passphrase supplied - reading card
Module #1 Slot #0: Processing ...
Card reading complete.

Please enter the passphrase for softcard 'mysoftcard':
Please wait........
Key successfully generated.
Path to key: C:\ProgramData\nCipher\Key Management Data\local\key_pkcs11_uce586891...
```

3.4.5. Secure strings

A passphrase or other sensitive string can be read into a variable in PowerShell using `$passphrase = Read-Host -AsSecureString`. `$passphrase` in this case is an instance of `System.Security.SecureString` and not the `System.String` type used for normal strings. The contents of a `SecureString` cannot be read directly. If you print the value of `$passphrase` in PowerShell, only the type name appears, not the value entered at the `Read-Host -AsSecureString` prompt.

nShield commands support using `SecureString` instances both for the input pipeline and as parameters to the nShield command. This helps reduce the visibility of plaintext passphrases or other sensitive values in scripts or in the shell. This is useful when using the input pipeline to automate the presentation of passphrases to the prompts in card-loading commands. It also means that nShield commands that take a passphrase as a command-line parameter can be presented that string without the string becoming directly visible.

Example:

```
ppmk --new --newpp $passphrase newsoftcard
```

3.5. Preload Utility

3.5.1. Overview

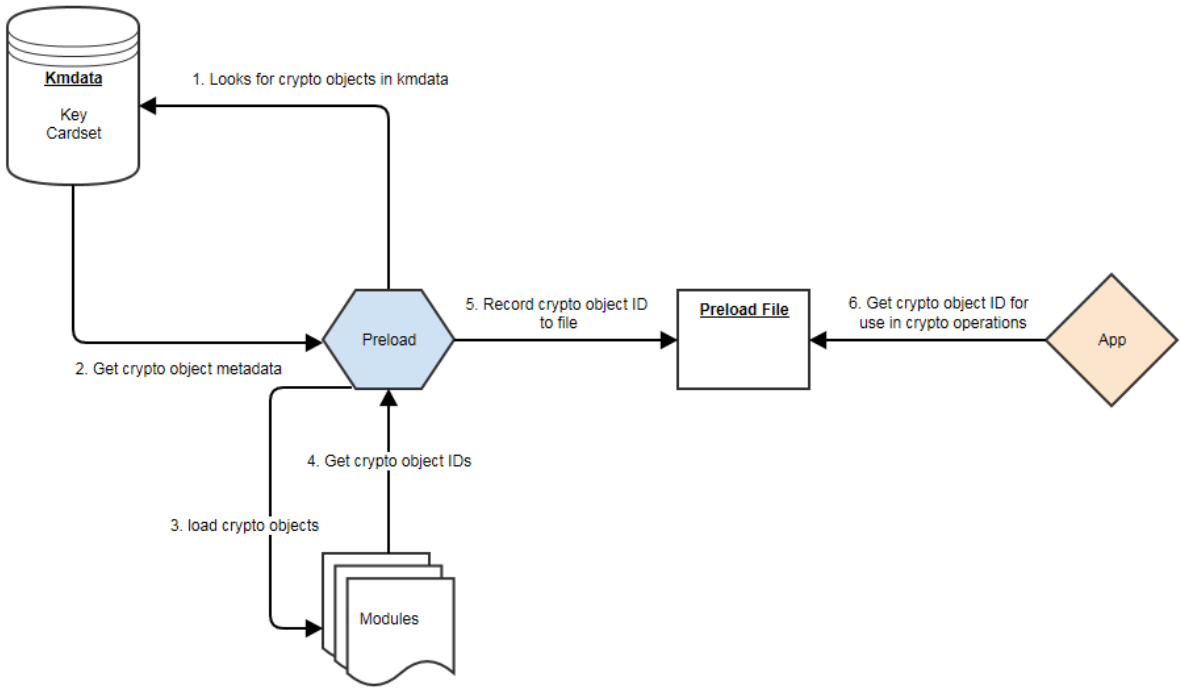
The preload utility loads persistent cryptographic objects (keys/OCS/softcards) onto a cho sen set of modules, then makes those objects available for use by applications. This removes the need for applications to load keys/cards themselves, and allows for easy shar- ing of keys/cards between multiple applications. Additionally, `preload` can manage keys, such that they are reloaded/maintained on modules to provide high availability.

Preloading is achieved via keys/cardsets being loaded, then once loaded the IDs of these objects are recorded persistently to a file (the preload file), which can be read via another application sharing the same session, and subsequently used.

Keys/cardsets must have previously been created before they can be preloaded, and all modules participating in a preload session must be in the same security world.

The preload binary can be found in `/opt/nfast/bin` (**Linux**) or `%NFAST_HOME%\bin` (**Windows**). This binary calls the `preload.py` script found in `/opt/nfast/python/scripts` (**Linux**) or `%NFAST_HOME%\python\scripts` (**Windows**).

The image below shows the relationship between preload, modules and applications:



3.5.2. Using Preload

3.5.2.1. Preload Commands

A command is needed in order to run preload. This command needs to be specified after the preload arguments.

The purpose of this command is to decide what needs to be done after preload has found and loaded all its crypto objects (OCS/softcards/keys).

```
> preload [arguments] command
```

Preload has a choice of 3 commands:

1. `pause` - continue to run the preload process forever. This is useful to load keys in one session and use them in another.
2. `exit` - exit preload gracefully. This is useful to add keys to the preload session. Not avail able in combination with high availability mode.
3. `subprocess` - execute this subprocess and exit once the subprocess has finished

The only exception to this is the `--list-admin` option that does not require a command.

The preload session remains open, and thus the preloaded keys remain loaded, as long as at

least one instance of preload continues to run. If/when the final preload instance terminates, all loaded objects will be cleaned up.

Example showing a single key, of type simple, being loaded and then an application being launched:

```
> preload -A simple -K key1 myapplication.py
```

3.5.2.2. Preload file location

The environment variable NFAST_NFKM_TOKENSFILE holds the path to the preload file. If it is not set, then the default location is used. A non-default location can also be set via the --preload-file option when invoking preload.

3.5.2.3. Preload Command Line Arguments

Argument	Effect
--version	show program's version number and exit
-h|--help	show help message and exit
-m MODULE_NUMBER|--module=MODULE_NUMBER	Load on specified module (may be repeated; default = all).
-c IDENT|--cardset=IDENT	Load all cardsets matching IDENT. If IDENT looks like a hash it will be interpreted as that, otherwise it will be interpreted as a name. If it is definitely a name, use --cardset-name.
--cardset-name=NAME	Load cardset(s) named NAME.
-s IDENT|--softcard=IDENT	Load all softcards matching IDENT. If IDENT looks like a hash it will be interpreted as that, otherwise it will be interpreted as a name.
--softcard-name=NAME	Load softcard(s) named NAME.
-o|--any-one	Load a single cardset.
-i|--interactive	Load cardsets interactively until told to stop.
-A APP|--appname=APP	Choose the appname for subsequent -K options.
-K IDENT|--key-ident=IDENT	Load keys with ident matching IDENT.
-n PATTERN|--name-pattern=PATTERN	Load keys with name matching PATTERN. Use * for wildcard.
--name-exact=NAME	Load keys with name NAME.
-M|--module-prot	Load all module protected keys, in addition to any others requested.

Argument	Effect
`--no-cardset-keys`	Do not automatically load keys protected by requested cardsets. Deprecated.
`--no-token-keys`	Do not automatically load keys protected by requested tokens.
`--admin=KEYS`	Load admin keys (separate with commas, or use `all`).
`--list-admin`	List available admin key names (for `--admin`).
`-F|--require-fips`	Require FIPS-auth to be loaded.
`-N|--no-fips`	Do not record FIPS auth, even if available. (overrides `-F`).
`-H|--high-availability`	High availability mode.
`--polling-interval=POLLING_INTERVAL`	Interval (s) between polls for changes to the module list (default=60). High availability mode only.
`-f PRELOAD_FILE|--preload-file=PRELOAD_FILE`	Use specified preloaded objects file, instead of the default.
`-R|--reload-everything`	Reload keys and tokens that are already loaded.
`--show-key-info`	Display key information for keys as they are loaded.
`-l|--file-logging`	Log to file.
`-S|--no-stderr-logging`	Do not log to `stderr`, this is independent of file logging.
`--log-file=LOG_FILE`	The file destination for the log, defaults to `preload_%pid.log` in the `nfast` log directory.
`--log-level=LOG_LEVEL`	The log level to log, options: `DEBUG`, `INFO`, `WARNING`, `ERROR`. Default is `INFO`, if unrecognized option it will fall back to default.

3.5.2.4. Pattern Matching

Options to preload that use pattern matching, namely `--name-exact` and `--key-ident`, can accept the following wildcards:

Wildcard	Definition
`*`	matches everything
`?`	matches any single character
`[seq]`	matches any character in seq
`[!seq]`	matches any character not in seq

It is advised that all arguments that using wildcards are surrounded by quotations to ensure

that they are passed to preload as intended. For example, to load all keys whose names start with `keyname`, the following pattern could be used:

```
> preload --name-pattern 'keyname*' exit
```

3.5.3. Preload File

The IDs of preloaded crypto objects are persistently stored in a preload file.

Each entry has the following format:

Element	Description
`Hash`	The `sha1 hash` of the crypto object.
`module`	The module which this object is present.
`objectid`	The id reference as a `M_KeyID`.
`generation`	This element is reserved for internal use.

Example `nfkminfo` output with preloaded crypto objects:

```
Pre-Loaded Objects ( 10):  objecthash    module     objectid   generation
c29da3ac0d99a7c01477831ac31a4bebe283c4f8      1   0xac57be2e            1
c29da3ac0d99a7c01477831ac31a4bebe283c4f8      2   0xac57be2d            1
1080cca2be9588e6e47bcd870ebcbb133ea0561b      1   0xac57be2c            1
1080cca2be9588e6e47bcd870ebcbb133ea0561b      2   0xac57be13            1
```

By default the preload file location is `/tmp` (**Linux**) or the current user's temporary folder (**Windows**):

Linux

```
/tmp/nfpriv_<username>/default
```

Windows

```
<current user's temporary folder>\nfpriv_<username>\default
```

This location can be changed by using the command line option `-f PRELOAD_FILE`|`--pre-load-file=PRELOAD_FILE`.

3.5.4. Softcard Support

Softcards are now supported in preload, along with module protected keys and OCS card-

sets.

In order to preload a softcard and the corresponding keys being protected by said softcard the `-s` or `--softcard-name` arguments can be used.

The `-s` option can be used with the softcard name or the hash of the softcard:

```
> nfkminfo -s
...
Operator logical token hash               name
3768b8efb7c7324dd8a1edbe2650c2015281c877  test
```

```
> nfkminfo -k simple aes128simplesoftcard1
...
name                  "aes128simplesoftcard1"
hash                  07c8110498dc0315455457f25564fc288c7da304
...
softcard              3768b8efb7c7324dd8a1edbe2650c2015281c877
```

```
> preload -s test nfkminfo
...
Pre-Loaded Objects ( 4):  objecthash   module objectid  generation
07c8110498dc0315455457f25564fc288c7da304   1 0xa411c0ab 1
07c8110498dc0315455457f25564fc288c7da304   2 0xa411c09e 1
3768b8efb7c7324dd8a1edbe2650c2015281c877   1 0xa411c09d 1
3768b8efb7c7324dd8a1edbe2650c2015281c877   2 0xa411c0a0 1
```

This shows the softcard is loaded on modules 1 and 2. It additionally shows that the key pro tected by the softcard has been loaded on both modules.

3.5.4.1. No Cardset Keys

The `--no-cardset-keys` command line option can also be used for softcards.

This command line option will ensure that only the softcard is preloaded, and no keys pro- tected by that cardset:

```
> preload -s test --no-cardset-keys
...
Pre-Loaded Objects (  2):  objecthash   module objectid   generation
3768b8efb7c7324dd8a1edbe2650c2015281c877     2 0xa9ba32a9 1
3768b8efb7c7324dd8a1edbe2650c2015281c877     1 0xa9ba32aa 1
```

3.5.5. FIPS Auth

FIPS Auth can be made available via preload.

The command line `-F` will ensure FIPS auth is preloaded everywhere.

The command line `-N` will ensure FIPS auth is not recorded, and will negate `-F`.

FIPS auth is also an admin key, see Admin Key section for more information.

3.5.6. Admin Keys

3.5.6.1. Listing

Admin keys can be listed using the `--list-admin` command line option.

This should be run without a command:

```
> preload --list-admin
```

Available admin keys are `NSO`, `M`, `RA`, `P`, `NV`, `RTC`, `FIPS`, `MC`, `RE`, `DSEE`, `FTO`.

3.5.6.2. Loading

Admin keys can be loaded using the `--admin=KEYS` command line option, supplying the value `--admin=ALL` to load all available admin keys. Note that admin key loading will require an ACS card being present in a slot of each module that is to be used.

Also note that the logical token of the admin key is preloaded alongside the key itself, for example, kfips and ltfips.

3.5.7. High Availability

Preload provides a high availability mode. When this mode is invoked Preload will load all requested keys, and will then periodically check for modules added or removed from the security world, or for keys becoming unloaded on existing modules. Should old or new mod ules be found to not have the specified keys/cardsets loaded, then preload will attempt to load them. This ensures that all available/usable modules have the requested keys loaded at all times, available for use by applications. Merged keyIDs are used to ensure applications can continually use these keys without interruption or changing key IDs. Preloaded keys are not only available to one application, but to any/all applications that share the preload ses- sion.

When preload is invoked with the `--high-availability` or `-H` option, it does the following differently:

1. Whenever preload loads a key onto the HSMs, it creates a Merged Key to represent the

set of (HSM, key ID) pairs. Applications will then use these merged IDs to address the keys.
    - As discussed below, this in itself provides failback, resilience and increased availability: the Merged Key ID remains usable even if some HSMs fail or are removed from the security world.
2. For as long as preload is running, it does the following repeatedly, once per polling interval:
    - Consult the hardserver to get a list of operational HSMs which are in the relevant security world.
    - For each Merged Key that was loaded by this instance of preload:
    - Ensure there is a valid current entry for each usable HSM.
    - To achieve this, check HSMs and load (or re-load) keys onto them as necessary, and update Merged Key contents.
    - Ensure that the individual key IDs within each Merged Key are valid: Remove any that are no longer valid and usable (such as those for a removed HSM).
    - Update the preload file to reflect changes, if any.
    - When finished, sleep for an interval of time, then repeat.

In summary, this mode attempts to keep preloaded crypto objects present on all usable modules in a security world (or a set of modules if requested via the -m argument) for as long as preload is running, with a keyID that remains constant, so that keys are available for use by any applications sharing the preload session.

3.5.7.1. Prerequisites for high availability mode

Users should not mix and match instances of preload with and without the `-H` high availability option, if those instances are sharing a session.

Managing OCS cardset-protected keys requires the following:

- the OCS protecting the key(s) be a 1/N quorum
- the passphrase for each card of the OCS set be identical
- one card of the OCS set be left inserted in a slot (local or remote) for each module
- if the card is non-persistent, it must be left in a local slot.

3.5.7.2. Differences from legacy behaviour

When running in high availability mode, certain behaviours of may differ from those outside of high availability mode. This includes the prompts for PIN entry and error messages. This

is due to a necessary difference in implementation between the two modes, and is expected.

3.5.7.3. Conditions for Management/Reloading

As mentioned above, preload in high availability mode will (re)load keys onto modules when a module is usable. A module will be considered usable if that module is in operational mode and in the correct world (and in the case of OCS protected keys, if a card from the OCS set is inserted into the module, locally or remotely). Preload will not attempt to perform actions that involve world administration, such as world loading or client enrolment. Users are responsible for managing worlds and client enrolment, and thus for bringing modules into a usable state.

The automatic loading/reloading of keys onto usable modules is not to be confused with forced reloading of keys provided by the -R option.

3.5.7.4. Merged Keys in the Preload File

When high availability mode is activated, all keys are represented in the preload file as Merged keys; cardsets and softcards are represented in the same way as non-high-availabil ity mode.

Due to the fact that in high availability mode keys are represented as MergedKeys, which do not correspond to any one particular module, the module element of the preload file is no longer relevant for keys. However, for cardsets, the module field is still utilized.

For symmetric and private halves of asymmetric keys the module number is represented as a `-1` and for public halves of asymmetric keys the module number is represented as a `-2`.

This is evident in the output from `nfkminfo`. (Note that `nfkminfo` ignores the 32-bit two's-complement representation, thus displaying -1 and -2 as (2^{32} - 1) and (2^{32} - 2) respectively: 4294967295 and 4294967294):

```
Pre-Loaded Objects (  4):  objecthash   module objectid  generation
84749a62d0f71db7f80c5df6469c11685f7f1b78   1 0xb5c0c7fa 1
84749a62d0f71db7f80c5df6469c11685f7f1b78   2 0xb5c0c7fd 1
28dcee51dfc53387f4dc4d55538d8b5253ee85d1 4294967295 0xb5c0c7f7 1
c2afe833ae6e823a37777c633a5b3a18a9e5dfbd 4294967294 0xb5c0c7f8 1
```

As shown above, cardsets/softcards are still module specific.

To make `nfkminfo` show the preloaded objects, run it as a `subprocess` as part of the `preload` command. See the section above on using preload.

Merged Key IDs (just like single-key IDs) are shared between multiple instances of preload that are invoked by the same client, that is, using the same `ClientID`. As such, applications must ensure that they perform no operations that delete or replace the merged key ID, or alter the keys that are part of that merged key ID.

3.5.7.5. Polling Interval

Preload manages its crypto objects by polling available modules, based on a polling interval.

Once per interval, if preload detects modules (new or existing) without the relevant crypto objects (keys/cards) present, it will attempt to load those missing objects.

This polling interval is configurable via the command line option `--polling-interval=SECONDS`

By default the polling interval is 60 seconds.

3.5.7.6. Key timeouts and use limits

It is advised to not use OCSs or keys with timeouts in high availability mode, as preload will be unable to reload objects once their timeouts have expired.

In high availability mode, there are situations where OCS/keys that have previously timed out, or reached maximum use limits, may be reloaded (and thus their limits reset) without user interaction. In general within high availability mode keys that have timed out or reached their use limits will be left in place, unusable, respecting the limits. However if the module containing those keys reboots or resets then, upon the module's return, preload will notice that the keys are not loaded and will load them. This reloading of keys will necessarily reset timeouts and use limits. If the timeout on an OCS has reached its limit, any keys protected by that OCS will not be reloaded on newly-indoctrinated modules in the security world.

3.5.7.7. Multiple Preload instances in high availability mode

As described above, keys will be maintained by the preload instance that first introduces them, and will cease being maintained when that instance ends. (Here maintained means reloaded automatically onto relevant HSMs that lack them.)

Therefore when `preload` is invoked with `exit` (or a short-lived `subprocess` command) it will load the specified keys but then exit, leaving those keys unmaintained.

If a `preload` process is already running under high availability mode, any new preload process (with the same preload file) will gain access to the preloaded keys. As such that later instance must also be run in high availability mode (and preload will reject an attempt to run it in plain mode in this situation).

The `pause` command may be useful for setting up availability of keys for subsequent use by multiple applications:

First, a long-running preload instance to load keys and maintain them indefinitely:

```
$ preload --high-availability [...other options...] pause
```

Then run applications (possibly short-lived) that use those keys:

```
$ preload --high-availability [...other options...] app --args --for --app
```

3.5.7.7.1. Managing Keys

Given multiple preload processes run under high availability, the process that will manage the keys is the first process to find them, based on command line options.

For example, Security World crypto objects:

crypto object	name	protected by
Softcard	softcard1	N/a
Key	simple_softcard1	softcard1
Key	simple_module1	module

First preload process started:

```
> preload -H -s softcard1 pause
```

This would load the softcard `softcard1` on all modules as well as the key `simple_softcard1`:

```
> preload -H nfkminfo
...
Pre-Loaded Objects (  3):  objecthash   module objectid  generation
29235f2a0b77fc1e18641b0820fe3c93e030a02e 4294967295 0x44313d41 1
5bccb6f540802ef1da3828f6b8b0f3fc985272e6   2 0x44313d47 1
5bccb6f540802ef1da3828f6b8b0f3fc985272e6   1 0x44313d46 1
...
> nfkminfo -k simple simplesoftcard1
...
name                 "simple_softcard1"
hash                 29235f2a0b77fc1e18641b0820fe3c93e030a02e
```

```
...
> nfkminfo -s
...
Operator logical token hash               name
5bccb6f540802ef1da3828f6b8b0f3fc985272e6  softcard1
```

Second preload process started:

```
> preload -H -n simple pause
```

This would load the key `simple_module` on all modules:

```
> preload -H nfkminfo
...
Pre-Loaded Objects (  4):  objecthash   module objectid  generation
600bcc26336c13f2371bdbb54b1cde293ded9a15 4294967295 0x44313d29 1
29235f2a0b77fc1e18641b0820fe3c93e030a02e 4294967295 0x44313d41 1
5bccb6f540802ef1da3828f6b8b0f3fc985272e6   2 0x44313d47 1
5bccb6f540802ef1da3828f6b8b0f3fc985272e6   1 0x44313d46 1
...
> nfkminfo -k simple simplemodule1
...
name                 "simple_module1"
hash                 600bcc26336c13f2371bdbb54b1cde293ded9a15
```

The evidence that the first preload process is still managing the key `simple_softcard1`, even though the second preload process could have loaded it, is in the `objectid`.

The object id for key `simple_softcard1` has not changed (**0x44313d41**).

3.5.7.8. FIPS Auth in High Availability mode

FIPS auth can be preloaded when running preload in high availability mode. In this scenario fips auth will be loaded as a high availability key (ie, reloaded/maintained on modules, as with other preloaded keys).

To enable FIPS auth use the command line option `-F`.

However, note that FIPS auth is represented differently, in comparison to other high availability mode keys, within the preload file.

The FIPS auth key is represented in the preload file multiple times: once for each module it is loaded on, and one extra time with a negative module ID as with other merged IDs. However the `objectid` is still a Mergedkey so will remain the same across those entries. This duplication of entries is to maintain compatibility with legacy behaviour/applications.

The following shows the pre-loaded FIPS auth objects on an estate of 3 modules - note there are 4 entries, each with the same `objectid`:

```
Pre-Loaded Objects (  4):  objecthash   module objectid  generation
aa462d0dd9dfeaa80968aadda2610ac0f6f94352   3 0xa824b9ab 1
aa462d0dd9dfeaa80968aadda2610ac0f6f94352   2 0xa824b9ab 1
aa462d0dd9dfeaa80968aadda2610ac0f6f94352   1 0xa824b9ab 1
aa462d0dd9dfeaa80968aadda2610ac0f6f94352 4294967295 0xa824b9ab 1
...
hkfips      aa462d0dd9dfeaa80968aadda2610ac0f6f94352
```

3.5.7.9. PKCS #11 and JCE

Both PKCS #11 and JCE applications are compatible with the high availability mode of preload, provided the PKCS #11 or JCE library that the application uses is from the 12.60 release or later. Flags or environment variables only need to be set to enable this when PKCS #11 is used for key reloading.

3.5.7.9.1. Use PKCS #11 for key reloading

PKCS #11 key reloading requires preload to be run in high availability mode, with the following options enabled:

- `--high-availability`.
- `--no-token-keys`.
- `--preload-file=PRELOAD_FILE`, where PRELOAD_FILE must match the location given to PKCS #11 with the NFAST_NFKM_TOKENSFILE environment variable.
- Either `--cardset=<IDENT>` or `--softcard=<IDENT>` (depending on whether using card set or softcard protected keys), where `<IDENT>` is the identifier of the card set or softcard, respectively.

  PKCS #11 key reloading is also supported for module-protected keys, but the PKCS #11 application must still be run under a preload application that is reloading tokens for another key.

  Using preload in high availability mode with Operator Card Sets has a set of restrictions, see Overview.

- Additionally, the following option is not required, but recommended:

  `--polling-interval=<POLLING_INTERVAL>`, where `<POLLING_INTERVAL>` also determines how often PKCS #11 will attempt to reload keys. The default is 60 seconds.

For more information, see PKCS#11 with key reloading.

3.5.7.10. Unsupported options

The `-H --high-availability` option may not be used in conjunction with any of the following options:

- `-o --any-one`
- `-i --interactive`
- `exit`
- `--admin`
- `--reload-everything`

3.5.8. Logging

By default `preload` logs to `stderr`.

Logs follow the format: `yyyy-mm-dd hh:mm:ss: [pid]: LogLevel: message`

For example:

```
2019-03-27 09:45:50: [439]: INFO: loading objects
```

Preload can also log to a file, this behaviour is separate from `stderr` logging. Therefore we can disable logging or log to `stderr` and/or a file.

To disable `stderr` logging, use the command line option `-S`. To enable file logging use the command line option `-l`.

The default file location for logs is `/opt/nfast/logs/preload_log_pid.log` (**Linux**) or `%NFAST_HOME%\logs\preload_log_pid.log` (**Windows**).

To change the file location, use the command line option `--log-file=FILE`.

As standard, preload has different log levels. These are:

- `DEBUG`
- `INFO`
- `WARNING`
- `ERROR`
- `CRITICAL`

The log level is by default: `INFO` and can be changed via the command line option `--log-level=LEVEL`.

3.5.9. Using preloaded objects - Worked example

In order to use preloaded objects, an application needs to create a connection that reads in the preload file:

Python:

```
import nfkm

conn = nfkm.connection(existingobjects="") # Reads file from default location

# If no existingobjects parameter is specified,
# the connection will not attempt to read any preload file:
conn_no_preload = nfkm.connection()
```

If the `existingobjects` argument is the empty string, the connection will use the file from the default location.

Any other string should be a path to different preload file. It can then call `NFKM_GetInfo` to get the security world info:

Python:

```
world_info = nfkm.getinfo(conn)
```

This results in a data structure with all the preloaded objects (this list is static and created at the time of connection creation):

Python:

```
import nfkm

conn = nfkm.connection(existingobjects="")

world_info = nfkm.getinfo(conn)

print world_info.existingobjects
```

Result:

```
[
ExistingObjectInfo
.module= 2
.hash= 84749a62 d0f71db7 f80c5df6 469c1168 5f7f1b78
.change= 1
.id= 0xffffffff88afd208,
ExistingObjectInfo
.module= 1
.hash= 84749a62 d0f71db7 f80c5df6 469c1168 5f7f1b78
.change= 1
.id= 0xffffffff88afd20b
```

```
]
```

Once an application has the M_KeyID references, it can use those cryptographic objects:

```
objid = world_info.existingobjects[0].id

cmd = nfkm.Command(["GetLogicalTokenInfo", 0, objid])

print conn.transact(cmd)
```

Result:

```
Reply.cmd= GetLogicalTokenInfo
.status= OK
.flags= 0x0
.reply.state= Present
.hkt= 84749a62 d0f71db7 f80c5df6 469c1168 5f7f1b78
.shares= empty
.sharesneeded= 0
```

3.6. Audit Logging

3.6.1. Aims of audit logging

Audit Logging on nShield HSMs provides the means to log administrative operations and key usage events across your estate of HSMs.

In a multi-tenant system audit logging will log administrative actions and key usage events for VCMs in the same way.

Some events are always logged and some events are logged only if certain conditions are met. Details of which events are logged, the conditions for logging the event, and the main information in those logs is described in Commands audited.

Logs are accurately time-stamped to allow correlation of these logs with other systems that may form part of your security or network environments.

Logs are signed and can be verified by the tools provided so that any attempts to alter the contents of the logs can be detected.

Audit logging is a compliance requirement of some certification schemes such as Common Criteria but can optionally be used with any Security World. The option must be specified when the World is created and cannot be added later.

3.6.1.1. Audit logging and CEF audit logging

There are two different audit log formats which differ depending on the firmware loaded on the HSM.

HSMs loaded with firmware versions prior to 13.5 will produce audit logs in CEF format and this guide refers to the production of logs in that format as 'CEF audit logging'.

HSMs loaded with firmware versions of 13.5 or later will produce audit logs in a new format and this guide refers to the production of logs in this format as 'audit logging'.

For a description of CEF audit logging and how to manage it, see the following v13.4 guides:

- Audit Logging (nShield Connect v13.4.5 User guide)
- Audit Logging (nShield Solo v13.4.5 User guide)
- Audit Logging (nShield Edge v13.4.5 User guide)

Since CEF was the only format of audit logs available prior to the release of Security World 13.5 older versions of the User Guides refer to CEF audit logging simply as 'audit logging'.

If your Security World uses a mix of HSMs with firmware before and after 13.5 you will receive logs in both formats. These logs are essentially independent and you must manage them separately.

For a detailed description of the audit logging format for firmware versions 13.5 and later and how it differs from CEF audit logging, see Understanding the New Audit Logging Format.

3.6.2. Configuring audit logging

3.6.2.1. Enabling audit logging

Audit Logging is enabled on an HSM when it is added to a Security World that is configured for audit logging.

Since accurate time-stamps are important for audit logging Entrust recommends that you ensure that the system clock has been accurately set before enabling audit logging. See Manage the system clock of an nShield 5s for help managing the system clock on nShield 5s HSMs.

The Security World is configured for audit logging when it is created. This can be done in one of two ways:

- Specifying the `--audit-logging` or `-G` option in the `new-world` command (this is

enabled by default in v13.9 and later Security World software)

- Specifying the `--mode=common-criteria-cmts` option in the `new-world` command

For the overall procedure, see Creating a Security World using new-world. Audit logging can also be enabled when creating a Security World via the nShield Connect front panel or with the CNG configuration wizard.

The `--mode=common-criteria-cmts` option will create a Security World supporting Common Criteria PP 419 221-5 which imposes other restrictions not related to audit logging. Only use this option if you require compliance to Common Criteria

There are some differences in the conditions that must be met for particular events to be audited depending on which of the two options above has been used to create the Security World. These differences are explained in Commands audited.

3.6.2.2. Disabling audit logging

Audit Logging is set for the lifetime of the Security World.

To disable Audit Logging on an HSM:

1. Reinitialize the HSM using `initunit`. See Erasing a module with initunit

Before removing the HSM from the Security World you should check that you have exported and verified all the logs that you require. Once the HSM has been erased it may not be possible to obtain and verify any logs that had not yet been exported.

3.6.2.3. Key usage logging

By default Audit Logging does not log usage of keys for cryptographic operations such as sign, verify, encrypt and decrypt or their usage in channels for these purposes. The capability to log these operations is determined on a per-key basis by the `LogKeyUsage` permission group flag on the ACL group authorizing the operation for which logging is desired.

See the *nCore Developer Tutorial* for further information on ACLs.

The `generatekey` utility (see Key generation options and parameters) provides the ability to set this permission group flag when a key is generated by either:

- Specifying `logkeyusage=yes` as an option on the command line
- Answering **yes** to the `logkeyusage` question if the command is being used interactively.

When `generatekey` is used this flag is applied to all permission groups but is only checked by the HSM on the group authorizing the desired action.

The following example shows this set on permission group 0 of a key's ACL.

```
groups[    0].flags= LogKeyUsage
                     .n_limits= 0
                     .n_actions= 2
                     .actions[    0].type= OpPermissions
                                         .details.oppermissions.perms= DuplicateHandle
ExportAsPlain GetAppData SetAppData
   ReduceACL ExpandACL Encrypt Verify UseAsBlobKey GetACL
```

The table in Commands audited shows which commands implement conditional key usage logging.

3.6.3. Commands audited

The table below shows which nCore commands may produce audit logs. The column titled 'Logged?' specifies if the logging is conditional or not. An entry in that column of 'ALWAYS' means that the command is unconditionally logged. An entry in that column of 'CONDITIONAL' means that logging is conditional according to the criteria below:

Conditional items:

(a) means the command is logged if an ACL check on a supplied key involves the `LogKeyUsage` bit, see Key usage logging

(b) means the command may need NSO permissions or (for files/shares with their own ACLs) some other certifier. If the command is certified using a `CertType_SigningKey` entry, a log record may be generated as a result of an ACL check (for `UseAsCertificate` permission) on that key.

(c) means the command is logged if the object involved is "loggable". Note that keys are always loggable if the Security World was created with `--mode=common-criteria-cmts`, see Enabling audit logging

(d) means this command becomes 'Always Logged' if the Security World was created with `--mode=common-criteria-cmts`, see Enabling audit logging

(e) means this command is logged if it accesses a (local or remote) smartcard

Commands marked "(a),(c)" will therefore generate audit-log records if either one of the "input" keys requires logging as part of an ACL check, or one of the "output" keys is a loggable object.

The other two columns in the table refer to the information that is logged for that command. See Audit log contents for more information.

Command	Logged?	CmdAuditInfo contents	Extra AuditInfo entries
Cmd_ChangeShareGroupPIN	ALWAYS	SlotID slot table: ShareInfo shares	ObjectUse(cert) TokenID
Cmd_ChangeSharePIN	ALWAYS	SlotID slot ShareInfo share	ObjectUse(cert) TokenID
Cmd_ChannelOpen	CONDITIONAL (a)	ChannelMode mode	ObjectUse(key,cert)
Cmd_CheckUserAction	CONDITIONAL (a)	-	ObjectUse(key,cert)
Cmd_CreateSEEConnection	ALWAYS	UUID containerid	-
Cmd_Decrypt	CONDITIONAL (a)	-	ObjectUse(key,cert)
Cmd_DeriveKey	CONDITIONAL (a),(c),(d)	-	ObjectUse(key,cert) ObjectNew
Cmd_Destroy	CONDITIONAL (c)	AuditObjectID objid Word refcount	-
Cmd_Duplicate	CONDITIONAL (a),(c),(d)	-	ObjectUse(key,cert) ObjectNew
Cmd_DynamicSlotCreateAssociation	ALWAYS	SlotID slot	-
Cmd_DynamicSlotsConfigure	ALWAYS	Word slot	-
Cmd_Encrypt	CONDITIONAL (a)	-	ObjectUse(key,cert)
Cmd_EraseFile	CONDITIONAL (b),(e)	-	ObjectUse(cert) TokenID
Cmd_EraseShare	ALWAYS	SlotID slot Word i ShortHash hkt	ObjectUse(cert) TokenID
Cmd_Export	CONDITIONAL (a),(d)	-	ObjectUse(key,cert)
Cmd_FileCopy	CONDITIONAL (b),(e)	-	ObjectUse(cert) TokenID
Cmd_FileCreate	CONDITIONAL (b),(e)	-	ObjectUse(cert) TokenID
Cmd_FileErase	CONDITIONAL (b),(e)	-	ObjectUse(cert) TokenID
Cmd_FileOp	CONDITIONAL (b),(e)	-	ObjectUse(cert) TokenID
Cmd_FormatToken	ALWAYS	SlotID slot optional: KeyHashEx auth_key	ObjectUse(cert) TokenID
Cmd_GenerateKey	CONDITIONAL (c),(d)	-	ObjectUse(cert) ObjectNew

Command	Logged?	CmdAuditInfo contents	Extra AuditInfo entries
Cmd_GenerateKeyPair	CONDITIONAL (c),(d)	-	ObjectUse(cert) ObjectNew (x 2)
Cmd_GenerateLogicalToken	ALWAYS	-	ObjectUse(key,cert) ObjectNew
Cmd_GetACL	CONDITIONAL (a)	-	ObjectUse(key,cert)
Cmd_GetAppData	CONDITIONAL (a)	-	ObjectUse(key,cert)
Cmd_GetRTC	CONDITIONAL (b)	-	ObjectUse(cert)
Cmd_GetTicket	CONDITIONAL (c)	AuditObjectID objid	-
Cmd_ImpathKXBegin	ALWAYS	optional: ASCIIString esn	ObjectNew
Cmd_ImpathKXFinish	ALWAYS	-	ObjectUse(imp)
Cmd_Import	CONDITIONAL (c),(d)	-	ObjectNew
Cmd_InitialiseUnit	ALWAYS	=InitialiseUnitEx	-
Cmd_InitialiseUnitEx	ALWAYS	InitModeFlags initmodeflags KMLType kmltype	Startup
Cmd_InsertSoftToken	ALWAYS	SlotID slot	-
Cmd_LoadBlob	CONDITIONAL (a),(c),(d)	-	ObjectUse(lt,key,cert) ObjectNew
Cmd_LoadLogicalToken	ALWAYS	-	ObjectUse(lt,cert) ObjectNew
Cmd_MakeBlob	CONDITIONAL (a)	-	ObjectUse(key,cert)
Cmd_NVMemAlloc	CONDITIONAL (b)	-	ObjectUse(cert)
Cmd_NVMemFree	CONDITIONAL (b)	-	ObjectUse(cert)
Cmd_NVMemOp	CONDITIONAL (b)	-	ObjectUse(cert)
Cmd_ReadFile	CONDITIONAL (b),(e)	-	ObjectUse(cert) TokenID
Cmd_ReadShare	ALWAYS	SlotID slot Word i Word sharesleft	ObjectUse(cert,lt) TokenID
Cmd_ReceiveKey	ALWAYS	-	ObjectNew ObjectUse(imp)
Cmd_ReceiveShare	ALWAYS	Word i Word sharesleft	ObjectUse(imp,lt)
Cmd_RedeemTicket	CONDITIONAL (c)	AuditObjectID objid Word refcount	-
Cmd_RemoveKM	ALWAYS	=RemoveKMEx	ObjectUse(cert)

Command	Logged?	CmdAuditInfo contents	Extra AuditInfo entries
Cmd_RemoveKMEx	ALWAYS	KeyHashEx hkm	ObjectUse(cert)
Cmd_RemoveSoftToken	ALWAYS	SlotID slot	-
Cmd_SendKey	ALWAYS	-	ObjectUse(imp,key,cert)
Cmd_SendShare	ALWAYS	SlotId slot Word i Key-HashEx hkm TokenHash hkt	ObjectUse(imp,cert) TokenID
Cmd_SetACL	CONDITIONAL (a),(d)	-	ObjectUse(key,cert)
Cmd_SetAppData	CONDITIONAL (a),(d)	-	ObjectUse(key,cert)
Cmd_SetKM	CONDITIONAL (a),(d)	-	ObjectUse(key,cert)
Cmd_SetNSOPerms	ALWAYS	=SetNSOPermsEx	-
Cmd_SetNSOPermsEx	ALWAYS	NSOPermsModeFlags nsoflags KeyHashEx hknso NSOPerms publicperms	-
Cmd_SetRTC	ALWAYS	RTCTime oldtime RTC-Time newtime	-
Cmd_Sign	CONDITIONAL (a)	-	ObjectUse(key,cert)
Cmd_SignModuleState	CONDITIONAL (a)	-	ObjectUse(key,cert)
Cmd_StartAuditLogging	ALWAYS	ByteBlock nonce	Startup
Cmd_StaticFeatureEnable	ALWAYS	FeatureInfo info	-
Cmd_StopAuditLogging	ALWAYS	-	Shutdown
Cmd_Verify	CONDITIONAL (a)	-	ObjectUse(key,cert)
Cmd_WriteFile	CONDITIONAL (b),(e)	-	ObjectUse(cert) TokenID
Cmd_WriteShare	ALWAYS	SlotID slot Word i	ObjectUse(lt,cert) TokenID
Cmd_CreateBuffer (Solo-XC only)	CONDITIONAL (a)	-	ObjectUse(key,cert)
Cmd_CreateSEEWorld (Solo-XC only)	ALWAYS	-	-
Cmd_ForeignTokenOpen (Solo-XC only)	ALWAYS	SlotID slot	-
Cmd_SetSEEMachine (Solo-XC only)	ALWAYS	-	-

3.6.4. Audit log contents

Audit log information is recorded in audit records. Audit records are sent from the HSM within a data structure called an audit segment. Each audit segment starts with an `AuditSegmentHeader` and ends with a signature block signed by a key `KML`. See Audit log verification for further information on this signature and how to verify it.

The number of audit records contained within an audit segment is variable and depends upon factors such as the processing load of the HSM and the number of auditable events being triggered.

Audit logs exported by the system are not in human readable format unless converted using a suitable tool such as `nshieldaudit`. For more information, see Read the audit log databases. The examples shown below have been converted to JSON format. A text format is also available.

3.6.4.1. Audit segment header contents

The `AuditSegmentHeader` contains the following information:

- The `ESN` of the HSM that produced the audit segment
- The `logID` (KML hash) of the HSM that produced the audit segment
- The `runID`. This changes each time the system boots
- The start and end audit indices
- A timestamp of when the audit segment was started in milliseconds since January 1 1970
- A timestamp of when the audit segment was signed in milliseconds since January 1 1970
- A hash of the audit logs contained within the segment

In a multi-tenant system the audit logs for a VCM will reference the ESN of the HSM upon which the VCM is hosted. See Identification of a VCM in the audit logs for more information.

An example `AuditSegmentHeader` is shown below:

```
"header": {
            "v": 0,
            "flags": "0x0",
            "esn": "14BD-B089-E078",
            "logID": "33133940 fb686dc5 4184a726 b670b7c1 508a8ea1",
            "runID": 4,
            "startIndex": 3458,
```

```
            "endIndex": 3458,
            "starttime": 1690801099591,
            "signtime": 1690801099591,
            "datahash": {
                "mech": "SHA256Hash",
                "data": {
                    "h": "2dcd7912 b9f0edd3 82df65f7 9ccdf289 4470a569 078638bf 20158eb1 e4c6aa6f"
                }
            }
```


3.6.4.1.1. Audit index

Each audit log record has an audit log index which increments by one for each log created since the start of audit logging. The `AuditSegmentHeader` shows the index for the first and last audit log contained in the segment.

In normal operation no audit records are lost and there will be no gaps in the audit index sequence.

If an abnormal shutdown occurs there is a possibility that some records may have been lost. This will be indicated by a gap in the audit index sequence and also by an `AbnormalShutdown` entry associated with the `StartAuditLogging` command.

It is not possible to tell exactly how many audit records were lost due to an abnormal shutdown. The gap in the audit index sequence indicates the maximum number that could have been lost based on the load of the HSM at the time of the abnormal shutdown but the actual number lost could be fewer than this and could even be zero.

3.6.4.2. Audit log contents

Each audit log produced will contain information as shown in the `CmdAuditInfo contents` column of the table in Commands audited. All audit logs contain basic information to indicate the name of the command that triggered the audit log and a status that indicates the outcome of that command, as in the extract below from an audit log produced by a `GenerateKeyPair` command.


```
{
      "type": "Command",
      "body": {
            "flags": "0x0",
            "cmd": "GenerateKeyPair",
            "info": {},
            "status": "OK"
      }
}
```


If the `CmdAuditInfo contents` column shows other information that will appear in the `info`

section.

If the `Extra AuditInfo entries` column of the table in Commands audited contains entries these will appear as separate log entries, as in the extract below.

```
{
     "type": "ObjectUse",
    "body": {
         "v": 0,
        "objid": 137,
        "action": {
            "type": "MakeBlob",
            "details": {
                "flags": "0x0"
            }
        }
    }
}
```

3.6.4.2.1. Object identification

Objects are identified by an `Objid`. A new ID is assigned when the object is created and will appear in the `ObjectNew` record as shown in the example below.

An object identifier is unique whilst the system runs uninterrupted but values may be reused if the system reboots. The combination of `RunID` from the audit segment header and `ObjID` is always unique.

```
 {
     "type": "ObjectNew",
    "body": {
        "v": 0,
        "objid": 135,
        "type": "Key",
        "details": {
            "type": "RSAPrivate",
            "hash": {
                "mech": "SHA1Hash",
                "data": {
                    "hash": "bb6b1e71 1cc06123 853af0d6 ff781cb0 b7a4b515"
                }
            }
        }
    }
}
```

The same ID will appear in `ObjectUse` records and other command records relating to the same object allowing you to trace the object through the audit log. If an object is copied the new object will have its own ID so that you can keep track of each copy. When the copy is created you can identify the object being copied by its hash.

3.6.4.3. Audit log verification

As described in Audit log contents audit log records are protected by a signature across the audit segment in which they are delivered. The signature is made using `KML` which is a secret key unique to the HSM and protected by the HSM.

Each time the HSM is re-initialized (e.g. after upgrading firmware and re-loading the Security World) the previous `KML` is destroyed and a new `KML` is created. The hash of `KML` is used as the `logID` for the audit log.

You can print the `KML` hash of the current Security World using `nfkminfo` and looking for `hkml`. If you have more than one HSM connected `nfkminfo` will show the `hkml` for each HSM.

3.6.5. Audit log administration

3.6.5.1. Reading the audit logs

Audit logs are read as described in the nShield Audit Log Service.

3.6.5.2. Audit logging and firmware upgrade

The audit log should be finalized before upgrading the firmware on an HSM enrolled in an audit logging Security World. To finalize the audit log, see Disabling audit logging

If you do not finalize the audit log before attempting to upgrade the firmware of a PCIe or USB HSM, you will receive an error message. You can either finalize the audit log and repeat the command, or use one of the following commands to override the warning:

- **nShield Solo, Solo XC, and Edge HSMs** The `loadrom` command with the `--noauditcheck` option.
- **nShield 5s HSMs** The `hsmadmin upgrade` command with the `--force` option.

Upgrading the firmware without finalizing the audit log will result in there being no final audit log record. This means it will not be possible to prove completeness of the audit log.

3.6.5.3. Audit logging and the system clock

It is important that the timestamps in the audit logs are accurate so that events can be correlated across the whole network in which the HSM is operating.

If the system clock is lost, for instance due to the HSM running on battery power for an extended period of time, ncoreapi commands that would result in an audit log being generated will be inhibited until the system clock is restored:

See Setting the system clock for more information on setting the system clock on nShield 5s HSMs.

3.6.6. Audit logging and VCMs

3.6.6.1. Identification of a VCM in the audit logs

In a multi-tenant system, the audit logs for a VCM reference the ESN of the HSM on which the VCM is hosted.

If a VCM is deleted and a new VCM is hosted on the same HSM, the audit logs for both VCMs contain the same ESN reference. However, the logs can still be uniquely traced to a specific VCM because each host machine can enroll only one VCM from a given HSM at a time.

3.6.6.2. Deleting a VCM and finalizing the audit log

The audit log should be finalized before deleting any VCM that is enrolled in an audit logging Security World. To finalize the audit log, see Disabling audit logging

If you do not finalize the audit log, the service provider will receive an error message when they attempt to delete the VCM.

You can either finalize the audit log and ask the service provider to repeat the command, or the service provider can follow the instructions at Delete a VCM using the `--force` option.

Deleting the VCM without finalizing the audit log will result in there being no final audit log record. This means it will not be possible to prove completeness of the audit log.

3.7. nShield Audit Log Service

3.7.1. Introduction

The nShield Audit Log Service (`nshieldauditd`) retrieves and removes audit logs from HSMs that are available to the local hardserver and are listed in the nShield audit log service's configuration file.

In a multi-tenant system the nShield Audit Log Service will retrieve and remove audit logs from VCMs in the same way.

There are two types of audit logs: nCore audit logs and signed system logs.

The nShield Audit Log Service retrieves nCore audit logs, see Audit Logging, from PCIe HSMs running firmware version 13.5 or later and from network-attached HSMs running firmware version 13.6 or later.

nShield Audit Log Service and the new audit databases support the new audit logs for 13.5 firmware onwards only. It does not retrieve the CEF-format audit logs from earlier firmware versions (the legacy audit configuration settings in `[auditlog_settings]` continue to be supported for those logs, and verification thereof can be done with the separate `cef-audit-verify` tool).

It also retrieves signed system logs from nShield 5s HSMs running firmware version 13.5 or later and nShield 5c HSMs running firmware version 13.6 or later. See System logging (nShield 5 HSMs).

Both types of logs are saved in separate local database files stored by default in a subdirectory of `NFAST_KMDATA`.

The service must be configured to enable log fetching from available HSMs. Once configured, the service runs automatically in the background without user intervention.

Connect XC and 5c HSMs enrolled in a Security World with audit-logging enabled (including a Common-Criteria (CMTS) compatible Security World) will continuously generate nCore audit-logs and, if these audit-logs are not transferred and removed by the nshieldauditd service, the HSMs may run out of disk space and will not process certain commands until their audit-logs are transferred and removed.

5s and 5c HSMs generate signed system logs internally in v13.5 onwards. These logs also need to be transferred and removed by the nShield Audit Log Service. This is true even when the Security World does not have audit-logging enabled.

3.7.2. nShield Audit Log Service configuration

The nShield Audit Log Service is configured by editing a YAML-format configuration file. Configuration changes can be applied by restarting the service.

3.7.2.1. Restarting nshieldauditd on Linux

The nShield Audit Log Service can be restarted on Linux by running the following command as `root`:

```
/opt/nfast/scripts/init.d/nshieldauditd restart
```

Instead of `restart` use `stop` followed by `start` parameters to explicitly stop and start the service respectively.

3.7.2.2. Restarting nshieldauditd on Windows

The service can be stopped and started from an elevated (Administrator) Windows command prompt using these commands:

```
net stop "nShield Audit Log Service"
net start "nShield Audit Log Service"
```

To restart the service as a single operation, run the following PowerShell command from an elevated (Administrator) PowerShell shell:

```
Restart-Service "nShield Audit Log Service"
```

3.7.2.3. Editing the configuration file on Linux

The configuration file is located at `/opt/nfast/kmdata/auditlogs/nshieldauditd.conf`. Editing the file requires membership of the `nfast` group.

3.7.2.4. Editing the configuration file on Windows

The configuration file is located at `C:\ProgramData\nCipher\Key Management Data\auditlogs\nshieldauditd.conf` by default. Editing the file requires the privileges of the built-in local Administrators group.

3.7.2.5. Default configuration file

The default configuration is created when the service first starts, and looks as follows:

```
# nShield Audit Log Service config file
#
# This configuration file is in yaml format.

#---------------------------------------------------------------------
# modules is the list of ESNs of HSMs and UUIDs of VCMs from which
# to fetch nCore audit-logs and (if applicable) signed system logs.
```

```
# By default, no modules are included in the list.
# Set modules: ["AAAA-AAAA-AAAA", "BBBB-BBBB-BBBB", "bf15b49c-9e8f-4072-b871-c147aed1c470"]
# to enable fetching of audit from modules with ESNs AAAA-AAAA-AAAA and BBBB-BBBB-BBBB
# and VCM "bf15b49c-9e8f-4072-b871-c147aed1c470"
# Comment out or delete the modules: [] line to fetch audit from all modules.

modules: []

#------------------------------------------------------------------
# syslog_fetch_interval_hours specifies the time in hours to wait
# between fetching the nShield 5 HSM syslog audit.
# It is always fetched when the service first starts.
# Default value is 12.

#syslog_fetch_interval_hours: 12

#------------------------------------------------------------------
# suppress_repeat_logs_interval specifies the time
# to wait for repeated log messages to be logged again.
# Set interval values in 's'(seconds), 'm'(minutes), or 'h'(hours),
# for example: '0s', '5m', '10h' etc.
# If set to '0s' then no logs will be suppressed.
# Recommended to set it to '0s' while in debug mode.
# Default value is '5m'.

#suppress_repeat_logs_interval: '5m'

#------------------------------------------------------------------
# logging_level controls the verbosity of log messages from the nShield Audit Log Service's own operations (not
the audit logs themselves)
# Available levels in increasing order of severity are DEBUG, INFO, WARNING, ERROR, or CRITICAL (default is INFO)

#logging_level: INFO
```

3.7.2.6. Configurable options

3.7.2.6.1. modules

The nShield Audit Log Service only fetches logs from the modules that are specified in the configuration file. If the `modules` field is commented out or absent then all HSMs available to the local hardserver will be monitored by the nShield Audit log service; this provides a short hand if the goal is to fetch audit from all enrolled modules.

In a multi-tenant system VCMs are specified by their UUID in the same way that an HSM is specified by its ESN.

By default, the `modules` field is an empty list, so this must be configured to enable the fetch ing of logs. Only one client system of the modules should enable log fetching, otherwise the audit log services will conflict with each other; that is why the user must explicitly opt-in to enable the fetching, to prevent any conflicts from default configuration. You can add or remove a module by editing the configuration file and restarting the nShield Audit Log Service. The `ESN` of each module to be enrolled should be specified within quotes and separated with a comma.

To fetch nCore or signed system logs from a network-attached HSM (Connect XC or 5c) the client machine where the nShield Audit Log Service has been configured must have been enrolled as a privileged client of that HSM.

Security World HSM estates with multiple client machines must configure just one client's nShield Audit Log Service to monitor any given HSM, although different nShield Audit Log Services may be configured to monitor different subsets of HSMs. If multiple services fetch logs from the same HSM, then each client's databases will only contain some of the log records.

3.7.2.6.2. syslog_fetch_interval_hours

This option configures how often the service should fetch the signed system logs from the HSM.

By default, `syslog_fetch_interval_hours` is commented out and the default value of 12 hours will be used.

The interval in the configuration file is in hours. The interval must be set in between 1 and 168 hours.

Signed system logs are produced at a relatively low rate, and fetching and removing the logs itself produces an audit record. This is why fetching of the logs is infrequent, unlike the continuous streaming of records that is done for nCore audit.

Signed system logs are not produced by nShield Solo XC and nShield Connect XC HSMs.

3.7.2.6.3. suppress_repeat_logs_interval

This option configures how often the service should show repeated log messages.

By default, `suppress_repeat_logs_interval` is commented out and the default value of 5 minutes will be used,

The interval for log suppression can be set in seconds, minutes or hours.

It is recommended to set the `suppress_repeat_logs_interval` to 0 when debugging.

3.7.2.6.4. logging_level

This option configures how much debug information is recorded in the service's application log. This is for monitoring the nShield Audit Log Service's operation in fetching nCore and signed system logs, and does not control the verbosity of that actual audit log data.

By default the service logging level is set to `INFO`, which will also report any higher severity messages. This is the recommended setting, although logging can be reduced by setting `WARNING`, `ERROR`, or `CRITICAL` to restrict only to messages at that log level or above.

If additional logging is desired, the `DEBUG` setting can be set, but note that this will write considerably more information to the service log, and will also slow down service operation as it will also include a human-readable representation of the nCore audit logs as they arrive. This is only recommended when investigating issues or if advised to do so by nShield Support.

On Linux, the service debug log is `/opt/nfast/log/nshieldauditd.log`. On Windows, the service debug log can be viewed in the Windows Event Viewer (`nCipherAuditSvc` event source) or can be queried using `nshieldeventlog -s nCipherAuditSvc -c 10` (for example) to view the most recent 10 lines of the service Event Log (adjust `-c` parameter for a larger number of lines, or use the `-f` parameter to write to a file).

When first configuring the system, it is recommended to check the service debug log to ensure that it reports that it has detected the intended modules and enabled log fetching for them.

3.7.3. Warrants

The nShield Audit Log Service attempts to retrieve the KLF2 warrant for each configured module automatically. It stores the warrants in the nCore audit log database for their module.

In nShield 5 HSMs, this warrant is embedded in the module and is always available.

In nShield XC HSMs, the KLF2 warrant must be present in one of the following locations in the filesystem of the machine running the nShield Audit Log Service, depending on how you are interacting with the HSM:

- `/opt/nfast/kmdata/warrants` (**Linux**) or `C:\ProgramData\nCipher\Key Management Data\warrants` (**Windows**)
  This is the same location as where a warrant should be placed if using nShield Remote Administration smartcards.
- `/opt/nfast/kmdata/hsm-<ESN>/warrants` (**Linux**) or `C:\ProgramData\nCipher\Key Man-`

agement Data\hsm-<ESN>\warrants (**Windows**)
Where <ESN> is the ESN of the relevant module.
This is where nShield Connect XC warrants are automatically placed on the RFS. If the warrant is not here, you will need to reconfigure (see rfs-setup) and reboot the HSM to copy the warrant to the correct location.

nShield Connect XC

To run the nShield Audit Log Service on a different client machine to the RFS, you must first copy the warrant file to /opt/nfast/kmdata/warrants or C:\ProgramData\nCipher\Key Management Data\warrants.

If there was no warrant present when the nShield Audit Log Service was started, you can still verify the chain up to the root of trust post-hoc with nshieldaudit. Copy the warrant to /opt/nfast/kmdata/warrants or C:\ProgramData\nCipher\Key Management Data\warrants on the machine where you are running nshieldaudit. When it displays the information, for example in nshieldaudit ncore query, it always re-runs the verification of the chain and will pick up the warrant from the aforementioned location.

3.7.4. Log databases

The nShield Audit Log Service saves the nCore audit logs and signed system logs in local database files. Logs for each module are saved in their own database in a subdirectory of the auditlogs directory named after the ESN of that module.

To back-up the log databases, first stop the nShield Audit Log Service, recursively copy the auditlogs directory, then start the service again. Stopping the service during backup ensures that a consistent state of the database files is copied, with no temporary transaction files being present. If restoring from backup, the original permissions and ownership of files and directories should be restored.

3.7.4.1. Linux

The databases are located in /opt/nfast/kmdata/auditlogs.

The directory and its contents have read and write permissions for nfast group by default. To restrict access to a different group, run the /opt/nfast/sbin/install script with NSHIELDAUDITD_GROUP environment variable set to the name of the alternate group. This must be set every time install is run in order to retain this setting. The nshieldauditd service user

will be added to the alternate group automatically, and the group will be created during install if it does not already exist.

3.7.4.2. Windows

The databases are located in `C:\ProgramData\nCipher\Key Management Data\auditlogs`.

Access is restricted to members of the built-in Administrators group (with elevated privilege) and the nShield Audit Log Service itself. To allow access to other users or groups without elevated privilege, the DACL of the directory may be modified, but note that it will be reset to defaults when the nShield software is re-installed.

3.7.4.3. Alternative log database directory

The above log database locations can be overridden by setting the environment variable `NFAST_AUDIT_LOGDIR` to the alternative location. This can be another local directory, or a network share.

If `NFAST_AUDIT_LOGDIR` is set, that also changes the service configuration file location to be inside that directory, that is, `NFAST_AUDIT_LOGDIR/nshieldauditd.conf`. Set the environment variable `NFAST_AUDIT_SERVICE_CONF` if a different location is desired for the configuration file.

These environment variables must be set in the environment of the service, and of any tools which read the logs. On Windows, set the environment variables in the Windows system variables, and on Linux in the `/etc/nfast.conf` file, in order to make them visible to the service. The service must be restarted for the change to take effect.

If you already have audit logs stored in the default location, it is recommended that the nShield Audit Log Service be stopped, the existing contents moved to the new location, and then the service started after setting the environment variables, so that the existing portion of the logs are not lost.

3.7.4.4. Read the audit log databases

The nShield Audit Log Service databases are accessed with the command line tool `nshield-audit` which manages logs using different subcommands as explained in the following sections. By default, running the `nshieldaudit` subcommands requires membership of `nfast` group on Linux and an elevated (Administrator) command prompt on Windows.

In a multi-tenant system use of `ESN` as an identifier on the command line will retrieve logs

for all VCMs hosted on the HSM identified by that ESN.

```
nshieldaudit <subcommand>
```

The audit logs in the databases are not in a human-readable format. Therefore, to be made human-readable, they must be exported to a file using the nshieldaudit tool.

Timestamps reported by nshieldaudit and in the audit logs themselves are UTC.

nshieldaudit supports the following subcommands:

- syslogs
- ncore
- serveradmin

3.7.5. Reading signed system logs

The nshieldaudit syslogs subcommand allows the user to access the signed system logs stored in the database.

```
nshieldaudit syslogs [-h] {query,export,info}
```

This subcommand takes the following parameters:

Parameter	Description
--help or -h	Show the help message and exit
query	Show the contents of the signed system logs database
export	Export the audit data to a file in JSON format
info	An overview of one or more log entries

3.7.5.1. Querying available signed system logs

Use the nshieldaudit syslogs query subcommand to quickly see what signed system logs databases are available, an overview of the date and sequence ranges of their content, and a "Log Status" to summarize any detected verification errors or missing data.

```
nshieldaudit syslogs query [-h] [-e ESN(s) [ESN(s) ...]] [-u UUID(s) [UUID(s) ...
```

The subcommand `nshieldaudit syslog query` takes the following parameters:

Parameter	Description
--help or -h	Show the help message and exit
-e ESN(s) [ESN(s) ...] or --esn ESN(s) [ESN(s) ...]	One or more ESN(s) of the HSM(s) that generated the audits
-u UUID(s) [UUID(s) ...] or --uuid UUID(s) [UUID(s) ...]	One or more UUID(s) of the VCM(s) that generated the audits

A typical output looks like the following:

```
$ nshieldaudit syslogs query
--esn AAA1-BBB2-CCC3
    Sequence numbers:          --start 1 --end 7775
    Time:                      --from 2024-01-13_12:11:03 --to 2024-05-28_08:31:40
    Malformed Logs:            0
    Unverified Logs:           0
    Unattested Logs:           0
    Missing Sequences:         None
    Log Status:                OK
--esn 1ABC-1DEF-1GHI
    Sequence numbers:          --start 13 --end 125
    Time:                      --from 2024-05-23_13:05:48 --to 2024-05-28_07:48:33
    Malformed Logs:            0
    Unverified Logs:           0
    Unattested Logs:           0
    Missing Sequences:         1-12
    Log Status:                Issues found
```

3.7.5.1.1. nshieldaudit syslogs info

This subcommand allows the user to see a more detailed summary of the available signed syslogs, for a given ESN, than the `nshieldaudit syslogs query` subcommand. It allows the time ranges and number of log lines in each of the sequence numbers in the signed syslogs to be listed, in support of constraining the range of interest when retrieving the log contents using the `export` subcommand.

```
nshieldaudit syslogs info [-h] -e ESN(s) [ESN(s) ...] -u UUID(s) [UUID(s) ...] [--start START_INDEX] [--end
END_INDEX] [--from FROM_DATE] [--to TO_DATE]
```

The subcommand `nshieldaudit syslog info` takes the following parameters:

Parameter	Description
--help or -h	Show the help message and exit
-e ESN(s) [ESN(s) ...] or --esn ESN(s) [ESN(s) ...]	One or more ESN(s) of the HSM(s) that generated the audits

Parameter	Description
-u UUID(s) [UUID(s) ...] or --uuid UUID(s) [UUID(s) ...]	One or more UUID(s) of the VCM(s) that generated the audits

The subcommand `nshieldaudit syslog info` allows the user to add the following filters:

Parameter	Description
--start START_INDEX	Start index of the audits desired
--end END_INDEX	End index of the audits desired
--from FROM_DATE	Start date of the audits desired (UTC)
--to TO_DATE	End date of the audits desired (UTC)

3.7.5.2. Exporting signed system logs to JSON format or text format

Use the subcommand `nshieldaudit syslogs export` to export the signed system logs data to a file. The output file is written in JSON format (default), or in the human-readable text format if the `-t` or `--text` parameter is added.

```
nshieldaudit syslogs export [-h] -e ESN(s) [ESN(s) ...] -u UUID(s) [UUID(s) ...] [-v] -f FILE [--overwrite] [--malformed] [-t] [--start START_INDEX] [--end END_INDEX] [--from FROM_DATE] [--to TO_DATE]
```

The subcommand `nshieldaudit syslog export` takes the following parameters:

Parameter	Description
--help or -h	Shows the help message and exits
-e ESN --esn ESN	One or more ESN(s) of the HSM(s) that generated the audits
-u UUID(s) [UUID(s) ...] or --uuid UUID(s) [UUID(s) ...]	One or more UUID(s) of the VCM(s) that generated the audits
-v, --verify	Re-verify signatures whilst exporting; this significantly increases export time. (default is to report cached verification status)
-f FILE, --file FILE	Required: Store output in this file (in JSON format)
--overwrite	Overwrite the output file if it exists
--malformed	Export malformed logs
-t, --text	Export of human-readable summary of logs in the text format rather than full JSON

The subcommand `nshieldaudit syslog export` allows the user to add the following filters:

Parameter	Description
--start START_INDEX	Start index of the audits desired
--end END_INDEX	End index of the audits desired
--from FROM_DATE	Start date of the audits desired (UTC)
--to TO_DATE	End date of the audits desired (UTC)

Run `nshieldaudit syslogs query` first in order to get parameters that can be copied to produce the `nshieldaudit syslogs export` command-line.

By default, the output format is JSON format (it is formatted with whitespace indentation to also be human-readable). To emit in a more compact human-readable text format, add the `--text` parameter. In many cases `--text` will be the more convenient format to use for browsing the data, and it is faster to export and produces a smaller output file.

Example command (text output):

```
nshieldaudit syslogs export --esn 5323-D01E-6DC0 -t
```

The output file will contain content like the following.

```
2024-11-12 06:51:19 HDR to=2024-11-12 07:51:33 esn=5323-D01E-6DC0 seq_no=1231
2024-11-12 06:51:19 nshield5 monitor[3987]: New log file: current seq-no=1231
2024-11-12 06:51:19 nshield5 monitor[3987]: Last log: {'signature': {'key': 'AAAA...', 'sig': 'AAAA...', 'mech':
'ecds...', 'hash': 'a684...'}, 'seq-no': 1230}
2024-11-12 06:51:19 nshield5 monitor[3987]: Command response: {'log-data': '2024-11-12 05:51:06 nshield5
moni...', 'signature': {'key': 'AAAA...', 'sig': 'AAAA...', 'mech': 'ecds...', 'hash': 'a684...'}, 'seq-no':
1230}
2024-11-12 06:51:28 nshield5 monitor[4002]: Command args: {'subcommand': 'expirelog', 'seq_no': 1230}
2024-11-12 06:51:28 nshield5 monitor[4002]: Removed saved log file messages.saved.1230
2024-11-12 06:51:28 nshield5 monitor[4002]: Command response: None
2024-11-12 07:51:33 nshield5 monitor[4017]: Command args: {'subcommand': 'exportlog', 'saved_log': False}
```

Run `nshieldaudit syslogs export --malformed` to export malformed logs into a file. By default the export file will be in JSON format unless `-t` or `--text` parameter is added.

```
nshieldaudit syslogs export --malformed -f ./log_export -e 5323-D01E-6DC0 --text --overwrite
```

3.7.6. Reading nCore audit logs

The `nshieldaudit ncore` subcommand allows the user to read nCore audit logs from the database.

```
nshieldaudit ncore [-h] {query,export}
```

This subcommand takes the following parameters:

Parameter	Description
--help or -h	Show the help message and exit
query	Shows the contents of the audit logs database
export	Export the audit data to a file in JSON (default) or text format

3.7.6.1. Querying available nCore audit logs

Use the `nshieldaudit ncore query` subcommand to quickly see what nCore audit log databases are available, an overview of the date and index ranges of their content, and a "Log Status" to summarize any detected verification errors or missing data.

```
nshieldaudit ncore query [-h] [-e ESN(s) [ESN(s) ...]] [-u UUID(s) [UUID(s) ...]] [-l LOGID [LOGID ...]]
```

This subcommand takes the following parameters:

Parameter	Description
--help or -h	Show the help message and exit
-e ESN(s) [ESN(s) ...] or --esn ESN(s) [ESN(s) ...]	One or more ESN(s) of the HSM(s) that generated the audits
-u UUID(s) [UUID(s) ...] or --uuid UUID(s) [UUID(s) ...]	One or more UUID(s) of the VCM(s) that generated the audits
-l LOGID [LOGID ...] or --logid LOGID [LOGID ...]	One or more LogID(s) of the nCore audit-log(s)

A typical output looks like this:

```
$ nshieldaudit ncore query
--esn A1A1-B2B2-C3C3 --logid f515236b5a4970f0e0ac7a3c2852c6b53fee28e3
    Indexes:                --start 1 --end 6546650
    Time:                   --from 2024-05-30_12:21:56 --to 2024-06-04_13:16:25
    KML Status:             Verification chain OK
    Finalized Status:       Finalized
    Malformed Segments:     0
    Unverified Segments:    0
    Missing indexes:        None
    Log Status:             OK
```

A brief description of each field:

Parameter	Description
Indexes	First and last Audit Index present in the local database for this log.
Time	UTC timestamps of the first and last audit records for this log.
KML Status	Status of verification chain of the KML public key (used to sign audit records) up to the nShield HSM root of trust.
KML Error	This field only appears if there is an error in KML Status, to give more details on KML verification error.
KML Warning	This field only appears if it was not possible to find a warrant to verify the chain of trust.
Finalized Status	`Finalized` if the module has been re-initialized (with `initunit` or by loading a Security World again) and `Unfinalized` otherwise.
Malformed Segments	Count of the number of audit log segments that could not be processed.
Unverified Segments	Count of the number of audit log segments that failed to verify with the KML key.
Missing indexes	List of the ranges of Audit Indexes that are missing in the database. This is inclusive counting, for example `1-2` means both Audit Index `1` and `2` are missing.
Log Status	`OK` if there are no detected issues with this log, and `Issues Found` if there are any errors or warnings reported in the other fields.

3.7.6.2. Exporting nCore audit to JSON or text format

Use the `nshieldaudit ncore export` subcommand to export the audit data to a file. The output file is written in JSON format (default), or in the human-readable text format if the `-t` or `--text` parameter is added.

```
nshieldaudit ncore export [-h] -e ESN(s) [ESN(s) ...] [-u UUID(s) [UUID(s) ...]] -l LOGID [LOGID ...] [-v] -f
FILE [--overwrite] [-t] [--start START_INDEX] [--end END_INDEX] [--from FROM_DATE] [--to TO_DATE]
```

The `nshieldaudit ncore export` command can take some time if there is a very large amount of data to export.

The subcommand `nshieldaudit ncore export` takes the following parameters:

Parameter	Description
--help or -h	Show the help message and exit
-e ESN or --esn ESN	One or more ESN(s) of the HSM(s) that generated the audits

Parameter	Description
-u UUID(s) [UUID(s) ...] or --uuid UUID(s) [UUID(s) ...]	One or more UUID(s) of the VCM(s) that generated the audits
-l LOGID or --logid LOGID	Required: The specific LogID from this module to export
-t, --text	Export of human-readable summary of logs in text format rather than full JSON
-v, --verify	Re-verify signatures whilst exporting; this significantly increases export time. (default is to report cached verification status)
-f FILE, --file FILE	Required: Store output in this file
--overwrite	Overwrite the output file if it exists
--malformed	Export malformed logs

The subcommand `nshieldaudit ncore export` allows the user to add the following filters:

Parameter	Description
--start START_INDEX	Start index of the audits desired
--end END_INDEX	End index of the audits desired
--from FROM_DATE	Start date of the audits desired (UTC)
--to TO_DATE	End date of the audits desired (UTC)

Run `nshieldaudit ncore query` first in order to get parameters that can be copied to produce the `nshieldaudit ncore export` command-line.

The `--esn` and `--logid` options must be supplied to specify the log whose data to export (the `nshieldaudit ncore query` command can be used to get these parameters), along with the `--file` parameter to specify the output file path.

By default, the output format is JSON format (it is formatted with whitespace indentation to also be human-readable). To emit in a more compact human-readable text format, add the `--text` parameter. In many cases `--text` will be the more convenient format to use for browsing the data, and it is faster to export and produces a smaller output file.

Example command (text output):

```
$ nshieldaudit ncore export --esn 4210-02E0-D947 --logid 9455993eca529189f44a4861d2c23ef359f549ff --text --file
ncore.audit.4210-02E0-D947.txt
100%
```

The output file will contain content like the following. Note that identical repeated opera-

tions within each "segment" of log like this are collated with the "REPEAT=+17" field which indicates that this command was repeated a further 17 times, that is, 18 times total. This reduces the verbosity of data when the system is under load.

```
2024-06-07 22:43:59.876 HDR logid=9455993eca529189f44a4861d2c23ef359f549ff runid=2 start=36398476 end=36398546
signtime=2024-06-07_22:43:59.883
2024-06-07 22:43:59.876 idx=36398476 src=Host cmd=Sign rc=OK obj=162 REPEAT=+17
2024-06-07 22:43:59.877 idx=36398477 src=Host cmd=Sign rc=OK obj=161 REPEAT=+19
2024-06-07 22:43:59.877 idx=36398480 src=Host cmd=Sign rc=OK obj=163 REPEAT=+15
2024-06-07 22:43:59.878 idx=36398484 src=Host cmd=Sign rc=OK obj=160 REPEAT=+16
```

Run `nshieldaudit ncore export --malformed` to export malformed logs into a file. By default the export file will be in JSON format unless `-t` or `--text` parameter is added.

```
nshieldaudit ncore export --malformed -f ./log_export_ncore -e 5323-D01E-6DC0 --text --overwrite -l
0032bf17c0aeb84aefa31c29f846886d39710eff
```

Audit records are verified automatically as they arrive by the nShield Audit Log Service. To re-verify records when exporting, pass the `--verify` option, otherwise the cached verification status is reported in the output file. Re-verification will significantly increase the time to export the logs.

3.7.7. Monitoring and managing audit data sources

3.7.7.1. Hardserver audit database storage limit

The hardserver has its own audit database as the temporary staging area for nCore audit records from modules that are local to it. This ensures that the audit records are not lost whilst waiting to be handed over to the nShield Audit Log Service.

- For Linux it is located at `/opt/nfast/hardserver.d/audit.db`.
- For Windows it is located at `C:\ProgramData\nCipher\hardserver.d\audit.db`.
- For network-attached HSMs (Connect XC and 5c), it is stored internally on the device.

This is not the same database as the nShield Audit Log Service database which is located as described in Log databases.

There is a configuration section `auditdb_settings`, which is present on both client-side and in network-attached HSMs configuration. If the section is missing, then the defaults will be applied. To learn more about the configuration section, see Audit Database setting.

A default `auditdb_settings` configuration section is as follows:

```
[auditdb_settings]
# Start of the auditdb_settings section
# Hardserver settings for (new format v13.5 and later) nCore audit logging
# database.
# Each entry has the following fields:
#
# Minimum available megabytes of free-space for audit DB to operate.
# (default=0 for auto-configured)
#  free_space_min_mb=MB
#
# Maximum total size of audit DB in megabytes. (default=0 for no limit)
#  db_size_max_mb=MB
#
# Maximum number of audit indexes in audit DB. (default=0 for no limit)
#  audit_indexes_max=COUNT
```

This is not the same configuration file as the nShield Audit Log Service configuration file (which is described in nShield Audit Log Service configuration).

By default network-attached HSMs (5c or Connect XC) pause HSM auditing when 500 MB of free space is left in the appliance, and client-side hardservers pause auditing of local HSMs when 50 MB of free space is left in the local storage. When HSM auditing is paused, any nCore commands which require the creation of an audit record will be denied. The paused status of an HSM will also be visible in the output of the `enquiry` tool in the `hardware status` field.

The configuration section changes can be applied dynamically by pushing configuration to a network-attached HSM. For local modules run the `cfg-reread` command to apply a change.

3.7.7.2. Checking nCore audit temporary storage usage

In normal usage, when the nShield Audit Log Service is operating correctly, the amount of temporary storage consumed for audit on HSMs should be small, as the data is being offloaded to the long-term databases. The following commands can be used to monitor to help detect issues.

Checking temporary storage for nCore audit usage for on a network-attached HSM (Connect XC or 5s) can be checked using the command `stattree RemoteServers 1 ServerGlobals` (replacing `1` with whatever the module number is).

Local temporary storage for nCore audit usage for Solo XC or 5s can be checked using the command `stattree ServerGlobals`.

The `AuditDBUsedSpaceMB` field states the number of megabytes of space consumed by the temporary database (where data is staged before being fetched by the nShield Audit Log

Service). The `AuditDBFreeSpaceMB` field states the number of megabytes of available space on the same partition as that temporary database.

Example output (for local storage usage):

```
stattree ServerGlobals | grep AuditDB
   -AuditDBFreeSpaceMB   102694
   -AuditDBUsedSpaceMB   45
```

3.7.7.3. Checking or repairing hardserver audit database

The `nshieldaudit serveradmin` subcommand enables the system administrator to check or repair, if needed, the temporary database of nCore audit logs not yet processed by the nShield Audit Log Service. It requires a privileged connection to the hardserver (membership of `nfast` group by default on Linux, and elevated Administrator command prompt by default on Windows). If managing a network-attached HSM, that must also have been enrolled as privileged.

This command only interacts with the hardserver to query its internal staging database that temporarily stores nCore audit data, and does not relate to the long-term databases that the `nshieldaudit ncore` and `nshieldaudit syslogs` subcommands provide access to.

To learn more about the hardserver internal `audit.db`, see hardserver audit database.

```
nshieldaudit serveradmin [-h] [-m MODULE] [-q] [-e ESN] [-l LOGID] [--delete] [--end END] [--force] [--delete-
status] [--recreate-db] [--no-confirm]
```

This subcommand takes the following parameters:

Parameter	Description
-m or --module	Module ID of source data server. Use 0 to refer to local hardserver.
-e or --esn	ESN of the log. Required when deleting.
-q or --query	query status of logs
-l or --logid	Log id of audit logs. Required when deleting.
--delete	Manually delete logs. Log deletion is normally automatic. This option should only be used in a recovery situation.
--end	When deleting audit logs the index of the last logs to be deleted.
--force	When deleting override check on whether logs have already been exported.

Parameter	Description
--delete-status	Delete the audit export status information. Not recommended unless the LogID is reported as finalized.
--recreate-db	Deletes entire database and creates a new empty database. Not recommended except in a recovery situation.
--no-confirm	Omits the interactive confirmation for --delete and --recreate-db options.

3.7.7.3.1. Querying hardserver database contents

Run `nshieldaudit serveradmin --query` to query the contents of the hardserver's temporary staging database.

The required hardware capabilities of the host machine running the nShield Audit Log Service depend on the amount of audit-log data that it needs to export, verify and expire. By running `nshieldaudit serveradmin --query` the user can monitor that the nShield Audit Log Service is successfully keeping up with offloading nCore audit records.

`nshieldaudit serveradmin --query` reports on the local system's hardserver by default (this will show information for audit from Solo XC and 5s modules). To query the temporary staging database of a network-attached HSM (Connect XC or 5c), add the `-m` or `--module` parameter to specify the module number of that HSM.

Example output (showing both the completed offload of a "Finalized" log as well as the in-progress offload of the log for the currently loaded Security World):

```
nshieldaudit serveradmin --query -m1
--esn=4210-02E0-D947 --logid=1330c21f241db5b2d86f5ddb99bccaacf12453df
    Indexes still on source data server (inclusive): --start=0 --end=0
    Total remaining index count: 0
    Exported to index: 218871
    Log creation time: 2024-04-08 10:12:56.145
    Log status: Finalized
--esn=4210-02E0-D947 --logid=9455993eca529189f44a4861d2c23ef359f549ff
    Indexes still on source data server (inclusive): --start=749879020 --end=749880109
    Total remaining index count: 1090
    Exported to index: 749879019
    Log creation time: 2024-06-07 20:57:03.480
    Log status: Unfinalized
```

Placeholder records of previously encountered logids seen by the system are kept by default (as in the "Finalized" log example above). If you wish to delete these placeholders for a "Finalized" log, you may issue the command `nshieldaudit serveradmin --delete --delete-status --esn=4210-02E0-D947 --logid=1330c21f241db5b2d86f5ddb99bccaacf12453df -m1` (replacing `1` with the module number in `-m1` - this may be omitted if it is a local module, and passing the actual `--esn` and `--logid` values for the log status information

to be deleted).

3.7.7.3.2. Forcibly deleting records or re-creating hardserver database

The `nshieldaudit serveradmin --delete` command can be used to delete records up to a specified `--end` value.

The `nshieldaudit serveradmin --recreate-db` command can be used to delete the entire database, that is, all status tracking information and any nCore audit records not yet offloaded will be lost.

In both cases, if interacting with a remote module, add the `-m1` (or appropriate module number) parameter to specify which hardserver the command should be sent to.

These commands will prompt for confirmation by default. They are not intended to be issued except in non-production test systems or as emergency recovery options if advised to do so by nShield Support.

3.7.7.4. Checking signed system logs HSM storage

The percentage free space for signed system logs temporary storage for a 5s or 5c can be queried using the command `appliance-cli -m1 gethsmenvstats`, replacing `1` with the module number. This is not applicable to XC modules.

Example output showing 90% free space, that is, only 10% used:

```
appliance-cli -m1 gethsmenvstats | grep log_free_space
           "log_free_space": 90,
```

Large consumption of this storage space is not expected in normal usage as signed system logs are only written occasionally, and the HSM stores the logs in a compressed form.

3.8. Configure the hardserver to export the module for guest VM usage

This feature is not available on nToken models, but works with all other local HSM models. If you previously configured this feature using rserverperm, you may wish to update to using these instructions using the config file to specify the permissions for guest VMs to access the HSMs in a persistent manner.

1. Configure the host hardserver to permit guest VM hardservers to share access to the module:
   a. Edit the host hardserver config file `NFAST_KMDATA/config/config` (**Linux**) or `NFAST_KMDATA\config\config` (**Windows**).
   b. Add a new entry in the `hs_clients` section to contain the details of the client to be added.

      If your config file does not already contain a `hs_clients` section you may add it yourself with a line containing only `[hs_clients]`.

      The `addr` and `clientperm` fields are required for each client, and `keyhash` is recommended for authentication: :

      ```
      [hs_clients]
      addr=<client_IP>
      clientperm=permission_type
      keyhash=software_keyhash
      ```

      Where:

      `<client_IP>` can be either the IP address of the guest VM or any of `0.0.0.0`, `::`, or blank if the host hardserver is to accept clients identified by their key hash instead of their IP address.

      If you set both the `<client_IP>` field (the guest VM's IP address) and the key hash, client connections will be restricted based on both values.

      `permission_type` defines the type of commands the client can issue (`unpriv` for unprivileged only, `priv` for privileged or `priv_lowport` for privileged connections restricted to low port numbers).

      `software_keyhash` is the hash of the software-generated authentication key that the client should authenticate itself with.

      If there is more than one client being configured, the fields for each client must be separated by line consisting of one or more hyphens (e.g. `----`).

      It is recommended that the firewall on the host be configured so that only connections from intended network interfaces can be made to the host hardserver on its Impath port (port 9004 by default).

   c. Load the updated configuration file in the host hardserver. To do this, run the fol-

lowing command:

```
hsc_nethsmexports
```

This command only needs to be run when the config is added or modified. The permissions for guest VMs will be re-applied automatically when the host hardserver is restarted.

2. Configure the hardserver in the guest VM to enroll to the host hardserver with an IP address using the virtual switch. Enter the following command for each guest hardserver that should have unprivileged access:

```
nethsmenroll <host-hardserver-ip>
```

Run the following command if the guest hardserver should have privileged access for mode change and administration:

Not all administration operations will be permitted from a privileged guest VM, such as firmware updates, which must be carried out from the host.

```
nethsmenroll -p <host-hardserver-ip>
```

You will be asked to confirm your entries. You should then see the following message:

```
OK configuring hardserver's nethsm imports
```

3. Confirm the connection from the guest VMs by running `enquiry`.

3.9. Logging, debugging, and diagnostics

3.9.1. Logging and debugging

When using a network-atttached HSM, you can view log files using the front panel. Application log messages are handled on the client.

The current release of Security World Software uses controls for logging and debugging that differ from those used in previous releases. However, settings you made in previous releases to control logging and debugging are still generally supported in the current release, although in some situations the output is now formatted differently.

Some text editors, such as Notepad, can cause NFLOG to stop working if the NFLOG file is open at the same time as the hardserver is writing the logs.

Debug logs may include sensitive data.

CKNFAST_DEBUG	All severities from DL_Call to DL_DetailMutex may result in sensitive data being logged
JCECSP_DEBUG	No further guidance
NFJAVA_DEBUG	No further guidance
NFLOG_DETAIL	The 0x00010000 flag may result in sensitive data being logged
NFLOG_SEVERITY	All the DEBUGn settings may result in sensitive data being logged

3.9.1.1. Environment variables to control logging

The Security World for nShield generates logging information that is configured through a set of four environment variables:

NFLOG_FILE

This environment variable specifies the name of a file (or a file descriptor, if prefixed with the & character) to which logging information is written. The default is stderr (the equivalent of &2).

Ensure that you have permissions to write to the file specified by NFLOG_FILE.

NFLOG_SEVERITY

This environment variable specifies a minimum severity level for logging messages to be written (all log messages less severe than the specified level are ignored). The level can be one of (in order of greatest to least severity):

1. FATAL
2. SEVERE
3. ERROR
4. WARNING
5. NOTIFICATION
6. `DEBUG`*N*, where *N* can be an integer from 1 to 10 inclusive that specifies increasing levels of debugging detail, with 10 representing the greatest level of detail, although the type of output is depends on the application being debugged.

The increasingly detailed information provided by different levels of `DEBUG`*N* is only likely to be useful during debugging, and we recommend not setting the severity level to `DEBUG`*N* unless you are directed to do so by Support.

The default severity level is `WARNING`.

`NFLOG_DETAIL`

This environment variable takes a hexadecimal value from a bitmask of detail flags as described in the following table (the `logdetail` flags are also used in the hardserver configuration file to control hardserver logging; see server_settings).

Hexadecimal flag	Function	logdetail flags
0x00000001	This flag shows the external time (that is, the time according to your machine's local clock) with the log entry. It is on by default.	external_time
0x00000002	This flag shows the external date (that is, the date according to your machine's local clock) with the log entry.	external_date
0x00000004	This flag shows the external process ID with the log entry.	external_pid
0x00000008	This flag shows the external thread ID with the log entry.	external_tid
0x00000010	This flag shows the external `time_t` (that is, the time in machine clock ticks rather than local time) with the log entry.	external_time_t
0x00000020	This flag shows the stack backtrace with the log entry.	stack_backtrace
0x00000040	This flag shows the stack file with the log entry.	stack_file
0x00000080	This flag shows the stack line number with the log entry.	stack_line
0x00000100	This flag shows the message severity (a severity level as used by the `NFLOG_SEVERITY` environment variable) with the log entry. It is on by default.	msg_severity
0x00000200	This flag shows the message category (a category as used by the `NFLOG_CATEGORIES` environment variable) with the log entry.	msg_categories

Hexadecimal flag	Function	logdetail flags
0x00000400	This flag shows message writeables, extra information that can be written to the log entry, if any such exist. It is on by default.	msg_writeable
0x00000800	This flag shows the message file in the original library. This flag is likely to be most useful in conjunction with Security World Software-supplied example code that has been written to take advantage of this flag.	msg_file
0x00001000	This flag shows the message line number in the original library. This flag is likely to be most useful in conjunction with example code we have supplied that has been written to take advantage of this flag.	msg_line
0x00002000	This flag shows the date and time in UTC (Coordinated Universal Time) instead of local time.	options_utc
0x00004000	This flag shows the full path to the file that issued the log messages.	options_fullpath
0x00008000	This flag includes the number of microseconds in the timestamp.	options_time_us
0x00010000	This flag enables logging of potentially secret values in generic stub log output.	msg_secrets

NFLOG_CATEGORIES

This environment variable takes a colon-separated list of categories on which to filter log messages (categories may contain the wild-card characters * and ?). If you do not supply any values, then all categories of messages are logged. This table lists the available categories:

Category	Description
nflog	Logs all general messages relating to nflog.
nflog-stack	Logs messages from `StackPush` and `StackPop` functions.
memory-host	Logs messages concerning host memory.
memory-module	Logs messages concerning module memory.
gs-stub	Logs general generic stub messages. (Setting this category works like using the `dbg_stub` flag with the logging functionality found in previous Security World Software releases.)

Category	Description
gs-stubbignum	Logs bignum printing messages. (Setting this category works like using the `dbg_stubbignum` flag with the logging functionality found in previous Security World Software releases.)
gs-stubinit	Logs generic stub initialization routines. (Setting this category works like using the `dbg_stubinit` flag with the logging functionality found in previous Security World Software releases.)
gs-dumpenv	Logs environment variable dumps. (Setting this category works like using the `dbg_dumpenv` flag with the logging functionality found in previous Security World Software releases.)
nfkm-getinfo	Logs `nfkm-getinfo` messages.
nfkm-newworld	Logs messages about world generation.
nfkm-admin	Logs operations using the Administrator Card Set.
nfkm-kmdata	Logs file operations in the `kmdata` (**Linux**) or `%NFAST_KMDATA%` (**Windows**) directory.
nfkm-general	Logs general NFKM library messages.
nfkm-keys	Logs key loading operations.
nfkm-preload	Logs `preload` operations.
nfkm-ppmk	Logs softcard operations.
serv-general	Logs general messages about the local hardserver.
serv-client	Logs messages relating to clients or remote hardservers.
serv-internal	Logs severe or fatal internal errors.
serv-startup	Logs fatal startup errors.
servdbg-stub	Logs all generic stub debugging messages.
servdbg-env	Logs generic stub environment variable messages.
servdbg-underlay	Logs messages from the OS-specific device driver interface
servdbg-statemach	Logs information about the server's internal state machine.
servdbg-perf	Logs messages about the server's internal queuing.
servdbg-client	Logs external messages generated by the client.
servdbg-messages	Logs server command dumps.
servdbg-sys	Logs OS-specific messages.
pkcs11-sam	Logs all security assurance messages from the PKCS #11 library.

Category	Description
pkcs11	Logs all other messages from the PKCS #11 library.
rqcard-core	Logs all card-loading library operations that involve standard message passing (including slot polling).
rqcard-ui	Logs all card-loading library messages from the current user interface.
rqcard-logic	Logs all card-loading library messages from specific logics.

You can set a minimum level of hardserver logging by supplying one of the values for the `NFLOG_SEVERITY` environment variable in the hardserver configuration file, and you can likewise specify one or more values for the `NFLOG_CATEGORIES` environment variable. For detailed information about the hardserver configuration file settings that control logging, see server_settings.

If none of the four environment variables are set, the default behavior is to log nothing, unless this is overridden by any individual library. If any of the four variables are set, all unset variables are given default values.

3.9.1.2. Logging from the nShield CSP and CNG (Windows)

By default, logging is disabled for the nShield CSP and CNG.

To enable logging, use a suitable registry editor such as `regedit`.

Depending on whether the program you wish to debug is 64-bit or 32-bit based, you will have to create respective registry keys if they do not already exist.

For a 64-bit program on a 64-bit OS, create the following registry key if it does not already exist:

`HKEY_LOCAL_MACHINE\SOFTWARE\nCipher\Cryptography`

For a 32-bit program on a 64-bit OS, create the following registry key if it does not already exist:

`HKEY_LOCAL_MACHINE\SOFTWARE\WOW6432Node\nCipher\Cryptography`

Open the registry at the required `Cryptography` key as described above, and under the key create the following variables.

1. Create a new `string` variable named `PathName`.
2. Open the `PathName` variable and provide a value which is a suitable path to where you want the log file(s) to be placed (for example, `C:\Users\MyName\Documents`.) Do not

give a log file name. The log file name(s) will be created automatically when logging starts.

It must be possible for the provider to write to the specified path.

3. Create a new `DWORD (32 bit)` variable named `LogLevel`.
4. Open the `LogLevel` variable and provide a suitable value (for example, `2`).

   Permitted values for `LogLevel` are:

Value	Output
0	Messages are sent to the event log.
1	Error messages are sent to the log file.
2	Function calls and error messages are sent to the log file.
3	All information, including debugging information, is sent to the log file.

Do not set this value to 3 unless either you are asked to do so by Support or you are debugging your own code. At this level of logging, a single SSL connection generates approximately 30 kilobytes of log messages. In addition, sensitive information may appear in the log file.

If `LogLevel` is not set, then the provider only logs messages of warning severity or greater (equivalent to setting `NFLOG_SEVERITY=warning`).

If neither `PathName` nor `LogLevel` are set for the CSP or CNG, logging remains disabled.

If logging is successfully enabled, the log file(s) should appear at the location specified in `PathName`, and will have names similar to:

- `nfdebug.txt`
- `ncspdddebug.txt`
- `nckspswdebug.txt`

3.9.1.3. Logging and debugging information for PKCS #11

In order to get PKCS #11 logging and debugging output, you must set the `CKNFAST_DEBUG` variable. A debug output file (with path) can also be set using the `CKNFAST_DEBUGFILE` variable. These variables can be set in the environment or the `/opt/nfast/cknfastrc` (**Linux**) or `%NFAST_HOME%\cknfastrc` (**Windows**) file. Normally settings in the environment should over-

ride the same settings (if any) in the `cknfastrc` file. You can specify any appropriate PKCS #11 categories using the `NFLOG_CATEGORIES` environment variable.

To produce PKCS #11 debug output, the `CKNFAST_DEBUG` variable can be a given value from 1 through to 11, where the greater the value the more detailed debug information is provided. A value of 7 is a reasonable compromise between too little and too much debug information. A value of 0 switches the debug output off.

This environment variable takes a colon-separated list of categories on which to filter log messages (categories may contain the wildcards characters * and ?).

The following table maps PKCS #11 debug level numbers to the corresponding `NFLOG_SEVERITY` value:

PKCS #11 debug level	PKCS #11 debug meaning	NFLOG_SEVERITY value	Output in log
0	DL_None	NONE	
1	DL_EFatal	FATAL	`"Fatal error:"`
2	DL_EError	ERROR	`"Error:"`
3	DL_Fixup	WARNING	`"Fixup:"`
4	DL_Warning	WARNING	`"Warning:"`
5	DL_EApplic	ERROR	`"Application error:"`
6	DL_Assumption	NOTIFICATION	`"Unsafe assumption:"`
7	DL_Call	DEBUG2	`">> "`
8	DL_Result	DEBUG3	`"< "`
9	DL_Arg	DEBUG4	`"> "`
10	DL_Detail	DEBUG5	`"D "`
11	DL_DetailMutex	DEBUG6	`"DM "`

3.9.1.4. Hardserver debugging (PCIe and USB HSMs)

Hardserver debugging is controlled by specifying one or more `servdbg-*` categories (from the `NFLOG_CATEGORIES` environment variable) in the hardserver configuration file; *server_settings* in the *Hardserver configuration file* chapter. However, unless you also set the `NFAST_DEBUG` environment variable to a value in the range 1 – 7, no debugging is produced (regardless of whether or not you specify `servdbg-*` categories in the hardserver configuration file). This behavior helps guard against the additional load debugging places on the CPU usage. You can set the desired `servdbg-*` categories in the hardserver configuration

file, and then enable or disable debugging by setting the NFAST_DEBUG environment variable.

The NFAST_DEBUG environment variable controls debugging for the general stub or hardserver. The value is an octal number, in the range 1 – 7. It refers bitwise to a number of flags:

Flag	Result
1	Generic stub debugging value.
2	Show bignum values.
4	Show initial NewClient or ExistingClient command and response.

For example, if the NFAST_DEBUG environment variable is set to 6, flags 2 and 4 are used.

If the NFAST_DEBUG environment variable value includes flag 1 (Generic stub debugging value), the logdetail value in the hardserver configuration file (one of the values for the NFLOG_DETAIL environment variable) controls the level of detail printed.

Do not set the NFAST_DEBUG environment variable to a value outside the range 1 – 7. If you set it to any other value, the hardserver does not start.

3.9.1.5. Debugging information for Java

This section describes how you can specify the debugging information generated by Java.

3.9.1.5.1. Setting the Java debugging information level

In order to make the Java generic stub output debugging information, set the Java property NFJAVA_DEBUG. The debugging information for NFJAVA, NFAST, and other libraries (for example, KMJAVA) can all use the same log file and have their entries interleaved.

You set the debugging level as a decimal number. To determine this number:

1. Select the debugging information that you want from the following list:

```
NONE = 0x00000000 (debugging off)
MESS_NOTIFICATIONS = 0x00000001 (occasional messages including important errors)
MESS_VERBOSE = 0x00000002 (all messages)
MESS_RESOURCES = 0x00000004 (resource allocations)
FUNC_TRACE = 0x00000008 (function calls)
FUNC_VERBOSE = 0x00000010 (function calls + arguments)
REPORT_CONTEXT = 0x00000020 (calling context e.g ThreadID and time)
FUNC_TIMINGS = 0x00000040 (function timings)
NFJAVA_DEBUGGING = 0x00000080 (Output NFJAVA debugging info)
```

2. Add together the hexadecimal value associated with each type of debugging information.

   For example, to set NFJAVA_DEBUGGING and MESS_NOTIFICATIONS, add 0x00000080 and 0x00000001 to make 0x00000081.

3. Convert the total to a decimal and specify this as the value for the variable.

   For example, to set NFJAVA_DEBUGGING and MESS_NOTIFICATIONS, include the line:

   ```
   NFJAVA_DEBUG=129
   ```

   For NFJAVA to produce output, NFJAVA_DEBUG must be set to at least NFJAVA_DEBUGGING + MESS_NOTIFICATIONS. Other typical values are:

   - 255: All output
   - 130: nfjava debugging and all messages (NFJAVA_DEBUGGING and MESS_VERBOSE)
   - 20: function calls and arguments and resource allocations (FUNC_VERBOSE and MESS_RESOURCES)

3.9.1.5.2. Setting Java debugging with the command line

You can set the Java debug options by immediately preceding them with a -D. Use the NJAVA_DEBUGFILE property to direct output to a given file name, for example:

```
java -DNFJAVA_DEBUGFILE=myfile -DNFJAVA_DEBUG=129 -classpath .... classname
```

Do not set NFJAVA_DEBUG or NFJAVA_DEBUGFILE in the environment because Java does not pick up variables from the environment.

If NFJAVA_DEBUGFILE is not set, the standard error stream System.err is used.

Set these variables only when developing code or at the request of Sup port.

Debug output contains all commands and replies sent to the hardserver in their entirety, including all plain texts and the corresponding cipher texts as applicable.

3.9.1.6. appliance-cli: system logs utility

See appliance-cli

3.9.2. Diagnostics and system information

Besides the diagnostic tools described in this section, we also supply a performance tool that you can use to test Web server performance both with and without an nShield HSM. This tool is supplied separately. If you require a copy, contact your Sales representative.

3.9.2.1. NIC Reporting (network-attached HSMs)

Networking information regarding NIC (Network Interface Card) status and address details can be reported via the `stattree` output.

NIC status and address details are disabled by default. To enable or disable NIC details in the `stattree` output is done via the configuration menu, **System** > **System Configuration** > **NIC exposure config** > **Device Status** and select the appropriate setting (**Yes** or **No**). Similarly for enabling or disabling the NIC addressing which is done via the **Address Reporting** option.

Reporting NIC addressing is only performed when device status is enabled.

A typical `stattree` report with NIC status and address details enabled looks like:

```
+#HostSysInfo:
   +SystemFans:
      +#1:
         -CurrentFanRPM        6240
      +#2:
         -CurrentFanRPM        6240
      +#3:
         -CurrentFanRPM        6240
      +#4:
         -CurrentFanRPM        6300
   +NetworkInterfaces:
      +#1:
         -InterfaceName        eth0
         -InterfaceLinkState   up
         +NetworkAddresses:
            +#1:
               -InterfaceNetworkAddress 172.23.135.127
      +#2:
         -InterfaceName        eth1
         -InterfaceLinkState   down
         +NetworkAddresses:
```

3.9.2.2. nShield HSM tamper log (network-attached HSMs)

The nShield HSM tamper log contains:

- The date and time of any tamper events
- Whether the tamper detection functionality was ever disabled or re-enabled.

It is no longer possible to disable tamper detection.

You view the tamper log using the nShield HSM front panel, by selecting **System** > **System information** > **View tamper log**. You cannot erase the tamper log.

3.9.3. How data is affected when a module loses power and restarts

nShield modules use standard RAM to store many kinds of data, and data stored in such RAM is lost in the event that a module loses power (either intentionally, because you turned off power to it, or accidentally because of a power failure).

Therefore, after restoring power to a module, you must reload any keys that had been loaded onto it before it lost power. After reloading, the `KeyID`s are different.

Likewise, after restoring power to a module, you must reload any cards that were loaded onto it before it lost power.

However, data stored in NVRAM is unaffected when a module loses power.

If you are using multiple nShield modules in the same Security World, and have the same key (or keys) loaded onto each module as part of a load-sharing configuration, loss of power to one module does not affect key availability (as long as at least one other module onto which the keys are loaded remains operational). However, in such a multiple-module system, after restoring power to a module, you must still reload any keys to that module before they can be available from that module.

3.10. System logging (nShield 5 HSMs)

This section describes how you can access the logging information generated by the platform. This information is separate from that produced by `ncoreapi`. See Platform services (nShield 5 HSMs) for more information about platform services and `ncoreapi`.

For information about the logging and debugging information generated by `ncoreapi`, see Logging, debugging, and diagnostics and Audit Logging.

Two system logs are automatically created by the system. These are the `init` log and the `system` log.

The `init` log records information created by the system when it boots. Its primary purpose

is for use by Entrust Support personnel.

The `system` log continuously records system level information whenever the system is running.

Both logs record information automatically and there is no user configuration required. The information recorded is determined by the system and there is no user configuration of the level of information recorded.

The commands used to retrieve and clear logs are described in hsmadmin logs.

3.10.1. Maximum log size

The `init` and `system` log share a common non-volatile storage area within the HSM. This storage area has the capacity to store the logs for a long period of normal usage but, if the logs are not periodically retrieved and cleared, it could eventually become full.

The `system` log is normally much larger than the `init` log but Entrust recommend that you clear both logs at the same time. This is because there are no specific warnings produced by actions that write to the `init` log. From firmware versions 13.5 onwards the `init` log sized is fixed and it is not necessary to clear the `init` log.

For HSMs running firmware version 13.5 or later, the system will detect when the logs have exceeded a safe working capacity and will prohibit the following actions:

- Factory state, see Return to Factory State
- Firmware upgrade, see Firmware upgrade
- Setting the minimum VSN, see Version Security Number
- Setting SSH keys, see Set up communication between host and module (nShield 5s HSMs)
- Setting or adjusting system time, see Setting the system clock and Adjusting the system clock
- Adminstration of CodeSafe, see The csadmin tool

These actions will continue to result in errors until the log volume is reduced back to a safe working level.

If the logs are not cleared promptly when the safe working capacity is exceeded the log volume may reach a critical capacity and at this point the system will enter an error state and display an error code on the LED. If this happens all actions are prevented other than retrieving and clearing the logs and it will be necessary to reboot the HSM to clear the

error state.

It is not normally possible to clear the `system` log from HSMs running v13.5 or later firmware without first exporting the entries. However it is possible to do this in recovery mode. See Recovery Mode.

3.10.2. Interaction with the system clock

It is important that the timestamps in the system logs are accurate so that events can be correlated across the whole network in which the HSM is operating.

For HSMs running firmware version 13.5 or later, if the system clock is lost, for instance due to the HSM running on battery power for an extended period of time, a number of administrator actions will be prohibited until the system clock is restored, including:

- Log expiry, see Expiring signed logs
- Log export, see Retrieving signed logs

See System interaction with the system clock for more information.

3.10.3. Init log

The `init` log records information that is produced each time the system boots.

The information is intended for use by Entrust Support to diagnose any issues that cause an HSM to fail to boot and thus the log is normally retrieved from recovery mode, see Recovery Mode.

The amount of information recorded depends on the firmware version loaded on the HSM. From firmware versions v13.5 onwards, after a successful boot, the `init` log will contain only a message indicating that the boot was successful.

3.10.3.1. Retrieving the init log

The `init` log can be retrieved with the following command:

```
hsmadmin logs get --esn <ESN> --log init [--json | --out <OUTFILE>]
```

Entrust recommend that you always direct the output to a file using the `--out` parameter and save the file for forwarding to Entrust Support.

3.10.3.2. Clearing the init log

The `init` log can be cleared with the following command:

```
hsmadmin logs clear [--esn <ESN>] --log init
```

Before running this command, place the unit in maintenance mode using `nopclearfail -M -m <MODULEID> -w`.

From firmware versions v13.5 onwards it is not necessary to clear the `init` log because it is cleared automatically each time the system successfully boots.

From firmware version v13.7 onwards this command will fail unless the unit is in recovery mode, see Recovery Mode.

3.10.4. System log

The `system` log records system level events and warnings.

For HSMs running firmware version 13.5 or later these logs are produced in a signed format. HSMs running firmware earlier than 13.5 produce logs in an unsigned format.

The procedures for managing logs differ depending on whether the logs are signed or unsigned.

3.10.4.1. Signed system logs

Signed system logs are produced by firmware version 13.5 or later and have a number of benefits.

Signed logs can be verified to ensure that they have not been tampered with. See Verifying Signed Logs.

Signed logs contain a sequence number that prevents unintentional deletion of logs and can be used to help order stored logs and identify any gaps. See Log Sequence Number.

3.10.4.1.1. Log sequence number

Signed system logs use a sequence number to prevent the unintentional loss of logs and to aid in sorting logs.

The HSM holds an internal sequence number for the system log currently being written. This sequence number is persistent over both reboots and factory state operations.

When the system log is exported see Retrieving Signed Logs the sequence number is incremented and a new log is started with the new sequence number. The previous log is not deleted by the `export` command but remains stored internally within the HSM.

To prevent unintentional duplication of logs, signed logs can normally only be exported once. If it is required to export the same log for a second time the `--saved` option must be used with the `export` command.

Once a log has been successfully exported it can be cleared from the HSM as described in Expiring Signed Logs. The use of the sequence number in this command ensures that no logs are deleted that have not been exported.

It is possible to export and expire logs in a single command but Entrust do not recommend doing this because if the command fails for any reason there is a risk that logs may be lost. The recommended procedure is to export the logs first and then expire the logs as a separate procedure only once the export has completed successfully.

Each exported log begins with one entry showing its own sequence number and another entry showing the sequence number of the preceeding log. This allows logs to be chained together to identify any missing gaps.

3.10.4.2. Retrieving the system log

3.10.4.2.1. Retrieving unsigned logs

The `system` log can be retrieved with the following command:

```
hsmadmin logs get --esn <ESN> --log system [--json | --out <OUTFILE>]
```

This command will retrieve the logs in unsigned format and will work for HSMs running any firmware.

3.10.4.2.2. Retrieving signed logs

For HSMs running firmware version 13.5 or later Entrust recommend that system logs are automatically retrieved by configuring the nShield Audit Log service as described in nShield Audit Log Service. If you wish to retrieve the logs manually you can do so with the following command:

```
hsmadmin logs export --esn <ESN> [--json | --outdir <OUTDIR>] [--saved] [--expire]
```

This command automatically verifies the logs as part of the export process

Logs should normally be exported only once. If an attempt is made to export a log that has already been exported the command will fail with a warning. If you wish to export a log that has previously been exported you should use the `--saved` option with the above command.

When upgrading to firmware version 13.5 or later from a firmware version lower than 13.5, there may initially be a saved log in the system created by the previous firmware. You should export this log using the `--saved` option and then expire it.

It is possible to automatically expire logs whilst exporting them by use of the `--expire` option but this is not recommended as it may result in loss of logs should the command fail for any reason.

3.10.4.3. Clearing the system log

The procedures for clearing the system log differ depending on whether the logs are produced in signed or unsigned format. If your HSM is running firmware version 13.5 or later your logs are produced in signed format. If your HSM is running a firmware version earlier than 13.5 your logs are produced in unsigned format.

The process of clearing a signed log is referred to as expiring.

3.10.4.3.1. Clearing unsigned logs

For HSMs running firmware versions earlier than 13.5 the log can be cleared with the following command:

```
hsmadmin logs clear [--esn <ESN>] --log system
```

For HSMs running firmware versions later than 13.5 this command will fail. You must follow the procedures described in Expiring Signed Logs.

It is possible to use this command to clear signed logs if the HSM is in recovery mode. See Recovery Mode. Entrust do not recommend this procedure unless instructed to do do by Entrust Support.

3.10.4.3.2. Expiring signed logs

If your HSM is running a firmware version of 13.5 or later the logs will be in signed format. To prevent unintentional loss of logs, signed logs must be exported before they can be cleared. You can export signed logs by following the procedures at Retrieving Signed Logs.

Logs that have previously been exported can be expired with the following command:

```
hsmadmin logs expire --esn <ESN> --seq SEQ_NO
```

This will expire the log with the sequence number `SEQ_NO`. The sequence number is included as the first line of any exported log.

It is possible to automatically expire logs whilst exporting them by use of the `--expire` option on the export command but this is not recommended as it may result in loss of logs should the command fail for any reason.

3.10.4.4. Verifying signed logs

Signed logs are automatically verified as part of the export process.

It is also possible to verify exported logs at any time in the future should you wish to do so, provided that you have access to the verification key. This verification can be conducted without access to the HSM.

The verification key is persistent over reboots but will be changed by a factory state operation. Therefore it is recommended that you record the verification key as soon as possible after any factory state operation. See Return to Factory State for information about factory state.

The system log will contain an entry for the factory state event. This log entry will contain the value of the verification key so it will always be possible to obtain the verification key if you forget to record it.

The current log verification key can be obtained with the following command:

```
hsmadmin logs getkey --esn <ESN> [--json | --out <OUTFILE>]
```

This command automatically validates the certificate protecting the verification key and produces a warning if it fails to find a certificate for the key. This warning is expected if the HSM:

- is in recovery mode

- has been upgraded to a firmware version of 13.5 or later but has not performed a factory state operation since the upgrade.

If you receive this warning in any other circumstance, contact https://trustedcare.entrust.com/.

If you have upgraded to firmware version 13.5 or later, but have not performed a factory state operation since the upgrade, perform a factory state as soon as possible. This generates an internal certificate for the log signing key, which allows validation of the key all the way back to the root of trust.

3.11. Application Performance Tuning

3.11.1. Job Count

To achieve the best throughput of cryptographic jobs (such as Sign or Decrypt) in your application, arrange for multiple jobs to be on the go at the same time, rather than doing them one at a time. This is true even when using only a single HSM in your system.

When using an nShield HSM, Entrust recommend that you set the number of outstanding jobs within the `rec. queue` (recommended queue) range specified by the `enquiry` output for the module.

If you are sending single jobs synchronously from each thread of your client application, try to keep the number of threads within this `rec. queue` range for best throughput.

When using higher-level APIs, such as PKCS#11, your application could benefit from increasing the thread count above the `rec. queue` range or the number that gives the best throughput when using nCore directly.

If you are load-balancing across multiple HSMs and want to maximize throughput across all of them, then use the sum of all `rec. queue` ranges for each of the modules to set the target for the outstanding jobs.

The `ncperftest` utility supports performance measurements of a range of cryptographic operations with different job counts and client thread counts. You may find this useful to inform tuning of your application. Run `ncperftest --help` to see the available options.

3.11.2. Client Configuration

If your application is coded directly against nCore, you have a choice of sending multiple

jobs asynchronously from a single client connection to the hardserver, or having multiple threads each with their own client connection to the hardserver with a single job sent synchronously in each. You can use the `--threads` parameter to the `ncperftest` utility to experiment with the performance impact of having more threads/connections with fewer jobs outstanding in each, or having fewer or just one thread/connection with more jobs outstanding in that connection.

When using higher-level APIs such as PKCS#11, all cryptographic operations are synchronous, so larger numbers of threads must be used to increase the job count and make full use of HSM resources. These APIs automatically create a hardserver connection for each thread. If many HSMs are being used, a great many threads may be required to achieve best throughput. You can adjust the thread counts in the performance test tools for these APIs (for example, `cksigtest` for PKCS#11) to gauge how much concurrency is required for best throughput in your application.

3.11.3. Highly Multi-threaded Client Applications

If your application is highly multi-threaded, operating system defaults may not be optimal for best performance:

You may benefit from using a scalable memory allocator that is designed to be efficient in multi-threaded applications, examples include `tcmalloc`.

On some systems the default operating system scheduling algorithm is also not optimized for highly multi-threaded applications. A real-time scheduling algorithm such as the POSIX round-robin scheduler may yield noticeable performance improvements for your application.

3.11.4. File Descriptor Limits (Linux)

On Linux systems, large numbers of threads each with their own hardserver connection will require your application to make use of large numbers of file descriptors. It may be necessary to increase the file descriptor limit for your application. This can be done using `ulimit -n NewLimit` on most systems, but you may need to increase system-wide hard limits first.