nShield Security World

nShield 5s v13.3 User Guide (Linux)

18 October 2024

Table of Contents

1. Introduction

1.1. Read this guide if ...

Read this guide if you need to configure or manage:

- An Entrust Hardware Security Module (HSM).
- An associated *Security World*. nShield hardware security modules use the Security World paradigm to provide a secure environment for all your HSM and key management operations.

All nShield HSMs support standard cryptography frameworks and integrate with many stan dards based products.

This guide assumes that:

- You are familiar with the basic concepts of cryptography and Public Key Infrastructure (PKI)
- You have read the *Installation Guide*.
- You have installed your nShield HSM.

Throughout this guide, the term *Installation Guide* refers to the particu lar Installation Guide for your product.

1.2. Terminology

The nShield 5s is also referred to as the *hardware security module* or the *nShield HSM*.

This guide refers to other nShield HSMs by type:

nShield HSMs	nShield HSM type
Connect, Connect +, Connect XC, 5c	Network-attached HSMs
Solo, Solo +, Solo XC, 5s	PCIe HSMs
Edge	USB-attached HSMs

1.3. Model numbers

Model numbering conventions are used to distinguish different nShield hardware security devices.

Model number	Used for
NC5536E-B	nShield 5s Base
NC5536E-M	nShield 5s Medium
NC5536E-H	nShield 5s High

1.4. Security World Software

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 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 nShield HSM is supplied with the latest version of the HSM firmware installed. For more information about:

- Upgrading the firmware, see Upgrading firmware.
- Installing and configuring the software on each client computer, see the Installation Guide and Client Software and module configuration.
- The supplied utilities, see Supplied utilities.
- Maintenance of your nShield hardware, see Maintenance of nShield Hardware.

1.4.1. 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 `NFAST_KMDATA` 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 Creating and managing a Security World.

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`.

1.4.2. 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
nShield Installation	/opt/nfast/
Key Management Data	/opt/nfast/kmdata/
Dynamic Feature Certificates	/opt/nfast/femcerts/
Static Feature Certificates	/opt/nfast/kmdata/hsm-ESN/features
Log Files	/opt/nfast/log
User Log Files	/home/<user>/nshieldlogs
Remote Static Feature Certificates	
opt/nfast/kmdata/hsm-ESN/features	Remote Dynamic Feature Certificates
	opt/nfast/kmdata/hsm-ESN/features

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.

The instructions in this guide refer to the locations of the software installation and program data directories by their names (for example, Key Management Data) or absolute paths (for example, /opt/nfast/kmdata).

If the software has been installed into a non-default location, you must create a symbolic link from /opt/nfast/ to the directory where the software is actually installed. For more information about creating symbolic links, see your operating system's documentation.

1.4.3. 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.

1.5. Setting the PATH for nShield utilities

It is recommended that the PATH environment variable be changed to include `opt/nfast/bin`.

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

This will allow all the nShield command-line utilities to be run without the need to type the full path, for example running `enquiry` instead of `opt/nfast/bin/enquiry>`.

`opt/nfast/bin` 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 a separate directory, `opt/nfast/python/bin`. 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.

1.6. 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.

1.7. 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://nshieldsupport.entrust.com, where any announcement of nShield Security Advisories will be made.

1.8. Recycling and disposal information

For recycling and disposal guidance, see the nShield product's Warnings and Cautions documentation.

2. Security Worlds

This chapter describes the *Security World* infrastructure we have developed for the secure life-cycle management of cryptographic keys. The Security World infrastructure gives you control over the procedures and protocols you need to create, manage, distribute and, in the event of disaster, recover keys.

A Security World provides you with the following features:

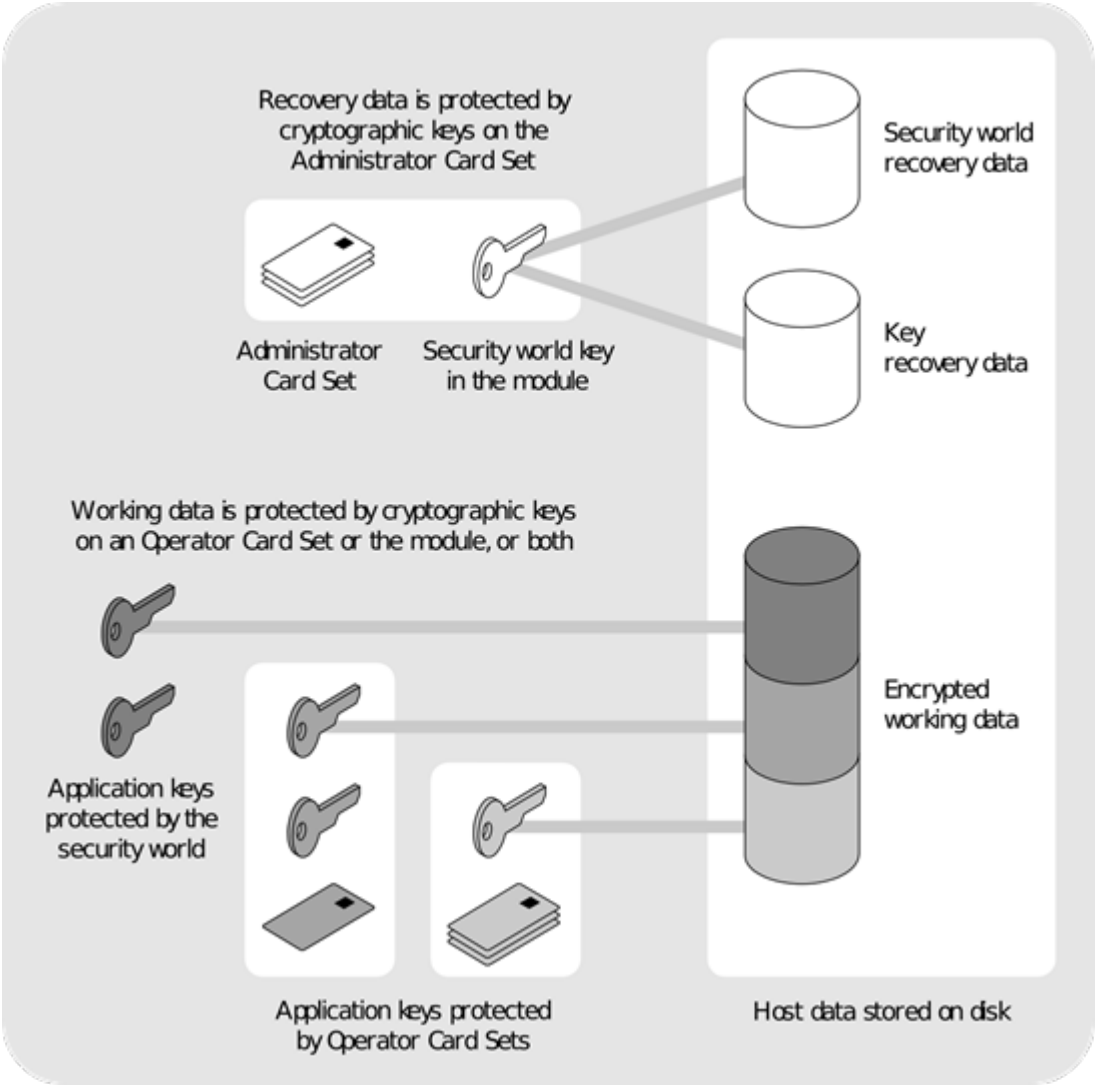
- Security
- Application independence
- Platform independence
- Flexibility
- Scalability
- Robustness
- Audit logging

A Security World comprises:

- One or more Entrust nShield HSMs
- An *Administrator Card Set* (ACS)
  A set of Administrator smart cards used to control access to the Security World config uration, as well as in recovery and replacement operations.
- Optionally, one or more *Operator Card Sets* (OCSs)
  A set or sets of Operator smart cards used to control access to application keys.
- Some cryptographic key and certificate data that is encrypted using the Security World key and stored on a host computer or computers

You can add or remove cards, keys, and even hardware security modules at any time. These components are linked by the Security World key, which is unique to each world. To see how these components are related to one another, see the image below.

Distributing the keys used for different tasks within the Security World over different stor- age media means that the Security World can recover from the loss of any one component. It also increases the difficulties faced by an attacker, who needs to obtain all the compo- nents before gaining any information.



2.1. Security

We have designed the Security World technology to ensure that keys remain secure throughout their life cycle. Every key in the Security World is always protected by another key, even during recovery and replacement operations.

Because the Security World is built around nShield key-management modules, keys are only ever available in plain text on secure hardware.

All Security Worlds rely on you using the security features of your operating system to control the users who can access the Security World and, for example, write data to the host.

2.1.1. Smart cards

The Security World uses:

- An *Administrator Card Set* (ACS) to control access to recovery and replacement functionality
- Zero or more *Operator Card Sets* (OCSs) to control access to application keys

In FIPS 140 Level 3 Security Worlds, you require a card from either the ACS or an OCS to authorize most operations, including the creation of keys and OCSs.

Each card set consists of a number of smart cards, *N*, of which a smaller number, *K*, is required to authorize an action. The required number *K* is known as the *quorum*.

The value for *K* should be less than *N*. We do not recommend creating card sets in which *K* is equal to *N* because an error on one card would render the whole card set unusable. If your ACS became unusable through such an error, you would have to replace the Security World and generate new keys. In Common Criteria CMTS Security Worlds the minimum value of *K* for the ACS is 2.

An ACS is used to authorize several different actions, each of which can require a different value for *K*. All the card sets are distinct: a smart card can only belong to the ACS or to one OCS.

Each user can access the keys protected by the Security World and the keys protected by their OCS. They cannot access keys that are protected by another OCS.

Operator Cards employ the Security World key to perform a challenge-response protocol with the hardware security module. This means that Operator Cards are only useable by an HSM that belongs to the same Security World.

2.1.2. Remote Operator

The Remote Operator feature is used to load a key protected by an OCS onto a machine to which you do not have physical access (for example, because it is in a secure area).

The Remote Operator feature is not available in Common Criteria CMTS Security Worlds.

The Remote Operator feature enables the secure transmission of the contents of a smart card inserted into the slot of one module (the *attended module*) to another module (the *unattended module*). To transmit to a remote module, you must ensure that:

- The smart card is from a persistent OCS
  See Using persistent Operator Card Sets for more about persistent cards.

- The attended and unattended modules are in the same Security World

To achieve secure communication channels between the attended and unattended modules, the hardserver uses an *impath* (an abbreviation of *intermodule path*), a secure protocol for communication over IP networks. The communication channels between the modules:

- Are secure against both eavesdroppers and active adversaries
- Can carry arbitrary user data as well as module-protected secrets, such as share data, that pass directly between modules.

2.1.3. Remote Administration

Remote Administration is a collection of features that allow you to configure and operate an HSM or set of HSMs without being physically present at the HSM. This includes creating ACS when creating a Security World and presenting ACS to authorize loading of a Security World. It also includes creating OCS to protect application keys and presenting OCS to authorize the loading of application keys. The OCS may be persistent or non-persistent.

The ACS and/or OCS cards must be nShield Remote Administration smart cards. When presenting a card, a secure channel is formed directly between the Remote Administration smart card and the target HSM before any token shares are read from or written to the smart card. The secure channel is secure against both eavesdroppers and active adversaries.

For more information see Security World Remote Administration

2.1.4. Client cooperation feature

The client cooperation feature allows nShield HSM host computers to automatically update the Security World and key data stored on a remote file system (RFS). For more information, see Setting up client cooperation.

2.1.5. NIST SP800-131A

When a new Security World is created it will be SP800-131A compliant.

2.1.6. FIPS 140 compliance

All Security Worlds are compliant with the Federal Information Processing Standards (FIPS)

140 specification. The default setting for Security Worlds complies with Level 2 of FIPS 140.

A Security World that complies with the roles and services section of FIPS 140 Level 2 does not require any authorization to create an OCS or an application key.

2.1.6.1. FIPS 140 Level 3 compliance

When you create a Security World, you can choose whether the Security World is compliant with the roles and services section of either:

- FIPS 140 at Level 2
- FIPS 140 at Level 3

The FIPS 140 Level 3 option is included for those customers who have a regulatory requirement for compliance with FIPS 140 at Level 3.

If you choose to create a Security World that complies with FIPS 140 Level 3, the nShield HSM initializes in that mode, conforming with the roles and services, key management, and self-test sections of the FIPS validation certificate.

Before you can create or erase an OCS in a Security World that complies with FIPS 140 Level 3, you must authorize the action with a card from the ACS or an OCS from that Security World.

For more details about FIPS 140, see http://csrc.nist.gov/publications/fips/fips140-2/fips1402.pdf.

2.1.7. Common Criteria compliance

The nShield XC range of HSMs are Common Criteria certified to Common Criteria v3.1 EAL4+ AVA_VAN.5 and to eIDAS.

To configure and operate the module in its evaluated configuration, the separate Common Criteria guides should be followed. Please contact Entrust nShield Support, https://nshield-support.entrust.com.

2.2. Platform independence

The Security World is completely platform independent. All key information is stored in a proprietary format that any computer supported by Security World Software can read, regardless of the native format used by that computer. This enables you to:

- Safely move a Security World between platforms with differing native formats. For example, you can move a Security World between Windows and Linux operating environments.
- Include hosts running different operating systems in the same Security World.

When copying host data between computers using different operating systems or disk formats, use a mechanism that preserves the original data format and line endings (such as `.tar` file archives).

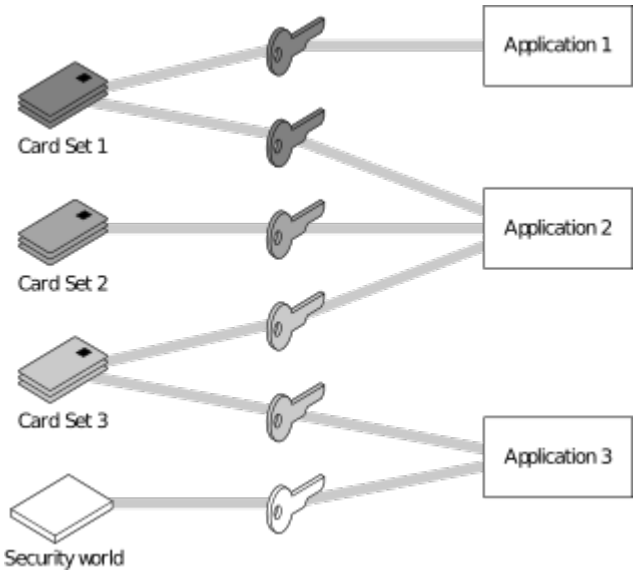
2.3. Application independence

A Security World can protect keys for any applications correctly integrated with the Security World Software. Each key belongs to a specific application and is only ever used by that application. Keys are stored along with any additional data that is required by the application.

You do not need to specify:

- Which applications you intend to use. You can add a key for any supported application at any time.
- How the key is used by an application. A Security World controls the protection for the key; the application determines how it is used.

Although keys belong to a specific application, OCSs do not. You can protect keys for different applications using the same OCS.



In the image above:

- **Card Set 1** protects multiple keys for use with **Application 1** and **Application 2**
- **Card Set 2** protects a single key for use with **Application 2**
- **Card Set 3** protects multiple keys for use with **Application 2** and **Application 3**
- The *Security World key* protects a single key for use with **Application 3**.

2.4. Flexibility

Within a Security World, you can choose the level of protection for each application key that you create.

When you create a Security World, a cryptographic key is generated that protects the appli cation keys and the OCSs in the Security World.

2.4.1. Using the Security World key: module-protected keys

You can use the Security World key to protect an application key that you must make available to all your users at all times. This key is called a *module-protected key*. Module-protected keys:

- Have no passphrase
- Are usable by any instance of the application for which they were created, provided that this application is running on a server fitted with a hardware security module belonging to the correct Security World.

This level of protection is suitable for high-availability Web servers that you want to recover immediately if the computer resets.

2.4.2. Using Operator Card Sets: OCS-protected keys

An OCS belongs to a specific Security World. Only a hardware security module within the Security World to which the OCS belongs can read or erase the OCS. There is no limit to the number of OCSs that you can create within a Security World.

An OCS stores a number of symmetric keys that are used to protect the application keys. These keys are of the same type as the Security World key.

Each card in an OCS stores only a fragment of the OCS keys. You can only re-create these keys if you have access to enough of their fragments. Because cards sometimes fail or are lost, the number of fragments required to re-create the key (K) are usually less than the total number of fragments (N).

To make your OCS more secure, we recommend that you make the value of *K* relatively large and the value of *N* less than twice that of *K* (for example, the values for *K*/*N* being 3/5 or 5/9). This practice ensures that if you have a set of *K* cards that you can use to recreate the key, then you can be certain that there is no other such card set in existence.

Some applications restrict *K* to 1.

2.4.2.1. Using Operator Card Sets to share keys securely

You can use OCSs to enable the same keys for use in a number of different HSMs at the same time.

If you have a non-persistent OCS, you must leave one of the cards in an appropriate card slot of each HSM. This should only be done if it is in accordance with the security policies of your organization.

To use OCS-protected keys across multiple HSMs, set:

- *K* to 1
- *N* at least equal to the number of the HSMs you want to use.

You can then insert single cards from the OCS into the appropriate card slot of each HSM to authorize the use of that key.

To issue the same OCS-protected key to a set of users, set:

- *K* to 1
- *N* equal to the number of users.

You can then give each user a single card from the OCS, enabling those users to authorize the use of that key.

If you have created an OCS for extra security (in which *K* is more than half of *N*), you can still share the keys it protects simultaneously amongst multiple modules as long you have enough unused cards to form a *K*/*N* quorum for the additional hardware security modules. For example, with a 3/5 OCS, you can load keys onto 3 hardware security modules because, after loading the key on the first device, you still have 4 cards left. After loading the key on a second device, you still have 3 cards left. After loading the key onto a third device, you have only 2 cards left, which is not enough to create the quorum required to load the key onto a fourth device.

If a card becomes damaged, you can replace the whole OCS if you have authorization from

the ACS belonging to that Security World.

You can only replace OCSs that were created by Security Worlds that have the OCS/softcard replacement option enabled. For more information, see OCS and softcard replacement.

2.4.2.2. Using Operator Card Sets for high availability

If you cannot risk the failure of a smart card, but some keys must remain accessible at all times, you can create a 1/2 OCS.

Use the first card as the working card and store the second card in a completely secure environment. If the working card fails, retrieve the spare second card from storage, and use it until you re-create a new set of 2 cards (see Replacing an Operator Card Set or recovering keys to softcards).

You can only replace OCSs that were created by Security Worlds that have the OCS/softcard replacement option enabled. For more information, see OCS and softcard replacement.

2.4.2.3. Using persistent Operator Card Sets

If you create a standard (non-persistent) OCS, you can only use the keys protected by that OCS while the last required card of the quorum remains loaded in the card reader. The keys protected by this card are removed from the memory of the hardware security module as soon as the card is removed from the card reader, which provides added security.

If you create a *persistent* OCS, the keys protected by a card from that OCS persist after the card is removed from the smart card reader.

This enables:

- The use of the same smart card in several hardware security modules at the same time
- Several users to load keys onto the same hardware security module at the same time.

The Security World Software maintains strict separation between the keys loaded by each user, and each user only has access to the keys protected by their OCS.

Keys protected by a persistent card are automatically removed from the hardware security module:

- When the application that loaded the OCS closes the connection to the hardware security module

- After a time limit that is specified when the card set is created
- When an application chooses to remove a key
- When the HSM is cleared. See Manually removing keys from an HSM for more information
- If there is a power loss to the module, for example, due to power outage.

Some applications automatically remove a key after each use, reloading it only when required. Such applications do not benefit from persistent OCSs. The only way of sharing keys between hardware security modules for such applications is by having multiple smart cards in an OCS.

Although the hardware security module stores the key, the key is only available to the application that loaded it. To use keys protected by this card in another application, you must reinsert the card, and enter its passphrase if it has one. Certain applications only permit one user at a time to log in, in which case any previously loaded persistent OCS used in that application is removed before the user is allowed to log in with a new OCS.

2.4.2.4. Manually removing keys from an HSM

You can manually remove all keys protected by persistent cards by clearing the hardware security module. For example, you could:

- Run the command `nopclearfail --clear --all`

Any of these processes removes all keys protected by OCSs from the hardware security module. In such cases, all users of any applications using the hardware security module must log in again.

Persistence is a permanent property of the OCS. You can choose whether or not to make an OCS persistent at the time of its creation, but you cannot change a persistent OCS into a non-persistent OCS, or a non-persistent OCS into a persistent OCS.

A Security World can contain a mix of persistent and non-persistent card sets.

2.4.3. Using passphrases for extra security

You can set individual passphrases for some or all the cards in an OCS.

You can change the passphrase for a card at any time provided that you have access to the card, the existing passphrase, and a hardware security module that belongs to the Security World to which the card belongs. For more information, see Changing card and softcard passphrase.

Some applications do not support the use of passphrases.

2.4.3.1. Maximum passphrase length

The maximum passphrase length limitation is not applicable to software versions before Security World Software v11.72.

passphrases are limited to a maximum length of 254 characters, when using the following commands:

- `new-world`
- `createocs`
- `cardpp`
- `ppmk`
- `racs`

Other commands are unaffected.

You can still use and edit existing passphrases that are longer than 254 characters.

Prior to Security World Software v11.72, we set no absolute limit on the length of passphrases, although individual applications may not accept passphrases longer than a spe cific number of characters. Likewise, the Security World does not impose restrictions on which characters you can use in a passphrase, although some applications may not accept certain characters.

Entrust recommends that your password only contains 7-bit ASCII characters:
`A-Z, a-z, 0-9, ! @ # $ % ^ & * - _ + = [ ] { } | \ : ' , . ? / ` ~ " < > ( ) ;`

2.4.3.2. passphrase penalty timer

The HSM maintains a penalty time, measured in seconds and based on the number of failed PINs. Each failed attempt to enter a passphrase adds 4 seconds to the penalty time.

The penalty timer has a 14s penalty threshold, the first 3 failed passphrase verifications do not incur a penalty delay. Before verifying a passphrase, the HSM waits for the current penalty timer to be below 14s. The penalty time decays over time.

A HSM only has a small number of command processing threads, related to the kind of hardware in use (for example, 9 threads on an nShield Solo). Once all of these are waiting for a penalty to expire, any other submitted commands will be forced to wait. This can mean that

> even if penalty time isn't large, the total delay experienced by clients may be substantial.

2.4.4. Using softcard-protected keys

If you want to use passphrases to restrict key access but avoid using physical tokens (as required by smart-card protection), you can create a *softcard-protected key*.

A *softcard* is a file containing a logical token that you cannot load without a passphrase. You must load the logical token to authorize the loading of any key that is protected by the softcard. Softcard files:

- Are stored in the `/opt/nfast/kmdata/local directory`
- Have names of the form `softcard_hash` (where `hash` is the hash of the logical token share).

Softcard-protected keys offer better security than module-protected keys and better availability than OCS-protected keys. However, because softcard-protected keys do not require physical tokens to authorize key-loading, OCS-protected keys offer better security than softcard-protected keys.

The passphrase of a softcard is set when you generate it, and you can use a single softcard to protect multiple keys. Softcards function as persistent 1/1 logical tokens, and after a soft card is loaded, it remains valid for loading its keys until its `KeyID` is destroyed.

2.5. Scalability

A Security World is scalable. You can add multiple hardware security modules to a server and share a Security World across multiple servers. You can also add OCSs and application keys at any time. You do not need to make any decisions about the size of the Security World when you create it.

To share a Security World across multiple servers:

- Ensure each server has at least one hardware security module fitted
- Copy the host data to each server, or make it available on a shared disk
- Use the recovery and replacement data with the ACS to load the required cryptographic keys securely onto every hardware security module.

If you create cards or keys in a Security World from a client rather than on the hardware security module (using the command line or KeySafe), you must transfer the files from the client to the remote file system, unless the client is already on the same computer as a

remote file system.

To provide access to the same keys on every server, you must ensure that all changes to the data are propagated to the remaining servers. If your servers are part of a cluster, then the tools provided by the cluster should synchronize the data. If the servers are connected by a network, then they could all access the same copy of the data.

There is no risk of an attacker obtaining information by snooping on the network, as the data is only ever decrypted inside a hardware security module. Alternatively, you can maintain copies of the data on different servers.

You can configure the host computer of an nShield HSM to:

- Access a Remote File System (RFS) as used by an nShield HSM.
- Share Security World and key data stored in the `/opt/nfast/kmdata/local` directory.

Client hardware security modules that access data in this way are described as *cooperating clients*. For more information, see Setting up client cooperation.

We provide the `rfs-sync` command-line utility to synchronize the `opt/nfast/kmdata/local` directory between a cooperating client and the remote file system it is configured to access. Run `rfs-sync` whenever a cooperating client is initialized, to retrieve data from the remote file system, and also whenever a client needs to update its local copy of the data (or, if the client has write access, to commit changes to the data).

2.5.1. Load-sharing

If you have more than one hardware security module on your system, your applications (that have been integrated with the Security World Software) can make use of the load-sharing features in the Security World Software to share the cryptography between them. Two approaches are supported:

- API specific load-sharing modes
- HSM Pool mode: a more generic load-sharing approach for module protected keys introduced with module firmware version 2.65.2.

Some applications may not be able to make use of these features.

HSM Pool mode is supported on all major APIs except Java (i.e. nCipherKM JCA/JCE CSP). When HSM Pool mode is enabled for an API, the application sees the HSMs in the Security World as a single resource pool. A significant benefit is that when a failed HSM is restored

to the Security World or a new HSM is added to the Security World, it is automatically added to the resource pool making it available for cryptographic operations without restart ing the application (i.e. failback support). The pool of HSMs can be viewed as a single resource using the command `enquiry --pool`.

`Module #1: Not Present` indicates that there are no HSMs in the pool.

2.6. Robustness

Cryptography must work 24 hours a day, 7 days a week, in a production environment. If something does go wrong, you must be able to recover without compromising your security. A Security World offers all of these features.

2.6.1. Backup and recovery

The Security World data stored on the host is encrypted using the Security World key.

You should regularly back up the data stored in the Key Management Data directory with your normal backup procedures. It would not matter if an attacker obtained this data because it is worthless without the Security World key, stored in your hardware security module, and the Administrator cards for that Security World.

When you create a Security World, it automatically creates recovery data for the Security World key. As with all host data, this is encrypted with the same type of key as the Security World key. The cryptographic keys that protect this data are stored in the ACS. The keys are split among the cards in the ACS using the same *K*/*N* mechanism as for an OCS. The ACS protects several keys that are used for different operations.

The cards in the ACS are only used for recovery and replacement operations and for adding extra hardware security modules to a Security World. At all other times, you must store these cards in a secure environment.

In FIPS 140 Level 3 Security Worlds, the ACS or an OCS is needed to control many operations, including the creation of keys and OCSs.

2.6.2. Replacing a hardware security module

If you have a problem with a hardware security module, you can replace it with a new hardware security module by using the ACS and the recovery data to load the Security World key securely. Use the same mechanism to reload the Security World key if you need to upgrade the firmware in the hardware security module or if you need to add extra hardware

security modules to the Security World.

If you have more than one hardware security module on your system and you use one of the load-sharing modes identified above, then your system is resilient to the failure of individual hardware security modules.

For information about replacing a hardware security module, see Adding or restoring an HSM to the Security World.

2.6.3. Replacing the Administrator Card Set

If you lose one of the smart cards from the ACS, or if the card fails, you must immediately create a replacement set using either:

- The KeySafe **Replace Administrator Card Set** option
- `racs` utility (see Replacing the Administrator Card Set).

You should also use `racs` or the KeySafe **Replace Administrator Card Set** option to migrate the ACS from standard nShield cards to nShield Remote Administration Cards. Authorization needs to take place using the local slot of an HSM.

When using the `racs` utility, you cannot redefine the quantities in a K of N relationship for an ACS. The K of N relationship defined in the original ACS persists in the new ACS.

A hardware security module does not store recovery data for the ACS. Provided that *K* is less than *N* for the ACS, and you have at least *K* cards available, a hardware security module can re-create all the keys stored on the device even if the information from other cards is missing.

The loss or failure of one of the smart cards in the ACS means that you must replace the ACS. However, you cannot replace the ACS unless you have:

- The required number of current cards
- Access to their passphrases.

Although replacing the ACS deletes the copy of the recovery data on your host, you can still use the old ACS with the old host data, which you may have stored on backup tapes and other hosts. To eliminate any risk this may pose, we recommend erasing the old ACS as soon as you create a new ACS.

2.6.4. Replacing an Operator Card Set or recovering keys to softcards

If you lose an Operator Card, you lose all the keys that are protected by that card. To prevent this, you have the option to store a second copy of the working key that the recovery key protects in a Security World. Similarly, you can recover keys protected by one softcard to another softcard.

The ability to replace an OCS is an option that is enabled by default during Security World creation (see OCS and softcard replacement). You can only disable the OCS replacement option during the Security World creation process. You cannot restore the OCS replacement option, or disable this option, after the creation of the Security World.

You can only recover keys protected by an OCS to another OCS, and not to a softcard. Likewise, you can only recover softcard-protected keys to another softcard, and not to an OCS.

To create new copies of the keys protected by the recovery key on a given card set, and to recover keys protected by one softcard to another softcard, use the `rocs` command-line utility.

2.6.4.1. The security of recovery and replacement data

Replacing OCSs and softcards requires authorization. To prevent the duplication of an OCS or a softcard without your knowledge, the recovery keys are protected by the ACS.

However, there is always some extra risk attached to the storage of any key-recovery or OCS and softcard replacement data. An attacker with the ACS and a copy of the recovery and replacement data could re-create your Security World. If you have some keys that are especially important to protect, you may decide:

- To issue a new key if you lose the OCS that protects the existing key
- Turn off the recovery and replacement functions for the Security World or the recovery feature for a specific key.

You can only generate recovery and replacement data when you create the Security World or key. If you choose not to create recovery and replacement data at this point, you cannot add this data later. Similarly, if you choose to create recovery and replacement data when you generate the Security World or key, you cannot remove it securely later.

If you have not allowed recovery and replacement functionality for the Security World, then you cannot recover any key in the Security World (regardless of whether the key itself was created as recoverable).

The recovery data for application keys is kept separate from the recovery data for the Security World key. The Security World always creates recovery data for the Security World key. It is only the recovery of application keys that is optional.

2.7. Audit Logging

Use of nShield HSMs in regulated environments where there is a requirement to provably log events in the HSMs can make use of the Audit Logging facility. This facility provides the following features:

- Tamper evident logging of relevant nCore command execution on the HSM
- Tied to Security World
- Traceability of cryptographic key lifetime
  - Authorization for key usage
  - Key loading onto HSM
  - Optional logging of key usage
  - Key destruction
- Compatibility with syslog and SIEM infrastructures
  - Logs produced in Common Event Format (CEF)
- Public key log verification without need for generating HSM.

For further information, see Audit Logging.

2.8. KeySafe and Security Worlds

KeySafe provides an intuitive and easy-to-use graphical interface for managing Security Worlds. KeySafe manages the Security World and the keys protected by it. For more information about using KeySafe, see Using KeySafe.

Most applications store only their long-term keys in the Security World. Session keys are short term keys generated by the application which are not normally loaded into the Security World.

Although you may use KeySafe to generate keys, it is your chosen application that actually uses them. You do not need KeySafe to make use of the keys that are protected by the Security World. For example, if you share a Security World across several host computers, you do not need to install KeySafe on every computer. To manage the Security World from a single computer, you can install KeySafe on just that one computer even though you are using the Security World data on other computers.

KeySafe enables you to:

- Create OCSs
- List the OCSs in the current Security World
- Change the passphrase on an Operator Card
- Remove a lost OCS from a Security World
- Replace OCSs
- Erase an Operator Card
- Add a new key to a Security World
- Import a key into a Security World
- List the keys in the current Security World
- Delete a key from a Security World.

KeySafe does not provide tools to back up and restore the host data or update hardware security module firmware, nor does KeySafe provide tools to synchronize host data between servers. These functions can be performed with your standard system utilities.

In addition to KeySafe, we also supply command-line utilities to manage the Security World; for more information about the supplied utilities, see Supplied utilities. Current versions of these tools can be used interchangeably with the current version of KeySafe.

2.9. Applications and Security Worlds

A Security World can protect keys for a range of industry standard applications. For details of the applications that are currently supported, visit https://nshieldsupport.entrust.com.

We have produced Integration Guides for many supported applications. The Integration Guides describe how to install and configure an application so that it works with Entrust hardware security modules and Security Worlds.

For more information about the Entrust range of Integration Guides:

- Visit https://nshieldsupport.entrust.com.
- Contact Support.

2.10. The nShield PKCS #11 library and Security Worlds

Many applications use a PKCS (Public Key Cryptography Standard) #11 library to generate and manage cryptographic keys. We have produced an nShield version of the PKCS #11 library that uses the Security World to protect keys.

Enabling a PKCS #11 based application to use nShield hardware key protection involves con figuring the application to use the nShield PKCS #11 library.

The nShield PKCS #11 library treats a smart card from an OCS in the current Security World as a PKCS #11 token. The current PKCS #11 standard only supports tokens that are part of a 1-of-*N* card set, however the nShield PKCS #11 library has vendor specific extensions that support K/*N* card sets, see nShield PKCS #11 library with the preload utility.

A Security World does not make any distinction between different applications that use the nShield PKCS #11 library. Therefore, you can create a key in one PKCS #11 compliant applica tion and make use of it in a different PKCS #11 compliant application.

2.11. Risks

Even the best-designed tools cannot offer security against every risk. Although a Security World can control which user has access to which keys, it cannot prevent a user from using a key fraudulently. For example, although a Security World can determine if a user is autho- rized to use a particular key, it cannot determine whether the message that is sent with that key is accurate.

A Security World can only manage keys that were created inside the Security World. Keys created outside a Security World, even if they are imported into the Security World, may remain exposed to a security risk.

Most failures of security systems are not the result of inherent flaws in the system, but result from user error. The following basic rules apply to any security system:

- Keep your smart cards safe.
- Always obtain smart cards from a trusted source: from Entrust or directly from the smart card manufacturer.

  nShield Remote Administration Cards can only be supplied by Entrust.

- Never insert a smart card used with key management products into a smart card reader you do not trust.
- Never insert a smart card reader you do not trust into your hardware security module.
- Never tell anyone your passphrase.
- Never write down your passphrase.
- Never use a passphrase that is easy to guess.

If you have any doubts about the security of a key and/or Security World, replace that key and/or Security World with a newly generated

one.

3. Platform services and ncoreapi

The nShield HSM firmware provides multiple services. These are divided into platform services and the `ncoreapi` service.

3.1. ncoreapi service

The `ncoreapi` service provides cryptographic services to the end user. This can either be via custom applications created by the end user accessing services using the `ncoreapi` service, as described in *nCore API Documentation* and *Cryptographic API*, or by using the utilities provided on the installation media.

3.1.1. Multi-tenancy ready

The system has been prepared for use in multi-tenant systems. In the current firmware version, only one instance of the `ncoreapi` service is allowed to run at any one time. Future versions of firmware will allow multiple instances of the `ncoreapi` service to run concurrently.

3.2. Platform services

Several platform services are provided which perform the tasks associated with the installation, commissioning, and maintenance of the HSM firmware and hardware. These run independently of the `ncoreapi` service.

The platform services are

Service name	Function
`updater`	This services provides functions to upgrade the HSM firmware
`setup`	This service provides functions to view the HSM 'lifetime' data installed in the factory and to return the HSM to factory settings
`monitor`	This service provides functions to retrieve and clear logs stored within the HSM
`sshadmin`	This service provides functions to manage the SSH keys used by the plat form services and the `ncoreapi` service

The administration of platform services is described in Administration of platform services

An interlock mechanism prevents the platform services from being accessed when the `ncoreapi` service is in operational mode. To access platform services the `ncoreapi` service

must be put into maintenance mode using `nopclearfail -M -m <MODULEID> -w`.

For example:

```
>nopclearfail -M -m 1
Module 1, command ClearUnitEx: OK
```

3.3. Separation of services

Each of the platform services and the `ncoreapi` service has its own communication channel with the host PC that is protected by use of SSH encryption. The procedure for installing the necessary SSH keys is described in Set up communication between host and module

4. Recovery mode

4.1. Recovery mode

nShield HSMs are loaded with two different firmware images:

- The primary image.
- A recovery image.

During normal operation, the HSM is running firmware that is loaded from the primary image.

If required, the HSM can be forced into recovery mode to run firmware loaded from the recovery image. Entry into recovery mode performs the same actions as `hsmadmin factorystate`

Recovery mode is useful in the following cases:

- To return the HSM to a known good state for disaster recovery.
- If the SSH keys used to communicate with the HSM have been lost and no backup is available. See Set up communication between host and module.

4.1.1. Restrictions in recovery mode

The main purpose of recovery mode is to allow essential maintenance activities that are not possible in primary mode.

When in recovery mode, the `ncoreapi` service does not run. Only the platform services are available, meaning that only the commands described in Administration of platform services are available.

Commands that make use of the `ncoreapi` service do not run and may show error messages.

4.2. Entry into recovery mode

Boot the HSM into recovery mode by holding down the recovery mode button on the back panel of the HSM whilst rebooting. See the appropriate *Installation Guide* for your nShield HSM for the location of the recovery mode button. This button is non-latching and must be held down for at least 60s after the reboot has been initiated. The reboot may be triggered either by `hsmadmin reset` or by power cycling the host machine containing the HSM.

Booting into recovery mode performs the same actions as `hsmadmin factorystate`. You must run `hsmadmin enroll` after the boot has completed before any further actions can be performed.

Run `hsmadmin status` to verify that the HSM is in recovery mode.

4.3. Exit from recovery mode

Exit recovery mode by booting the HSM without the recovery mode button held down. If the firmware is changed whilst in recovery mode using `hsmadmin upgrade`, the unit automati cally reboots.

When the unit next boots into primary mode it will be in factory state. You must run `hsmadmin enroll` again before any further actions can be performed.

Run `hsmadmin status` to verify that the HSM is in the correct mode.

5. Software installation

See the appropriate *Installation Guide* for your nShield module for more about installing the Security World software.

After you have installed the software, you must complete further Security World creation, configuration and setup tasks before you can use your nShield environment to protect and manage your keys.

5.1. After software installation

After you have successfully installed the Security World Software, as described in the *Instal lation Guide*), complete the following steps to finish preparing your HSM for use:

1. Ensure that your public firewall is set up correctly. See the *Installation Guide* for your HSM for more information about firewall settings.
2. If the SSH keys have not been set up, create the communication path between the host machine and the HSM, as described in Set up communication between host and module.

   If you followed the steps in the *Installation Guide* when installing the software, this should already be set up.

3. If necessary, perform additional software and HSM configuration tasks, as described in Client Software and module configuration:
   - Set up client configuration, as described in Setting up client cooperation.
   - Set nShield specific environment variables, as described in Setting environment variables.
   - Configure logging and debugging parameters, as described in Logging and debug-ging
   - Configure Audit Logging, as described in *Audit Logging*.
   - Configure Java support for KeySafe, as described in Configuring Java support for KeySafe
   - Configure the hardserver, as described in Configuring the hardserver.
4. Create and configure a Security World, as described in Creating a Security World.
5. Create an OCS, as described in Creating Operator Card Sets (OCSs).

6. Set up communication between host and module

6.1. Overview of SSH keys

Communications between the host and the HSM are protected by use of SSH secure channels. To allow mutual authentication of the endpoints, the SSH protocol uses separate key pairs in the host and the HSM. The functionality within the HSM is divided into different services that use separate SSH channels. See Platform services and ncoreapi. You need to install the SSH keys for each service before you can use those services.

Entrust recommends that you back up the `sshadmin` key as described in Making a backup whenever SSH keys are installed or changed, if your security policy allows.

6.2. Installation of SSH keys as part of software installation

The `hsmadmin enroll` command automates the installation of SSH keys.

This command is run automatically as part of the software installation script

6.3. Installation of SSH keys independently of a software installation

If the HSM has been returned to factory state, either with the `hsmadmin factorystate` command or by booting the HSM in recovery mode, as described in Recovery mode, you must install the SSH keys with the `hsmadmin enroll` command before any other actions can be performed.

The `hsmadmin enroll` command can be run on a module in which SSH keys have already been installed. In such a system, the command detects that valid keys already exist and takes no action.

If you are installing SSH keys due to their accidental loss or erasure, and you have previously made a backup of the `sshadmin` key using `hsmadmin keys backup`, then you can install them without returning your HSM to factory state by passing the path to the backed-up `sshadmin` key to `hsmadmin keys restore`.

6.4. Viewing installed SSH keys

The SSH keys installed on the host and each connected HSM can be viewed using the com mand `hsmadmin keys show`. All the keys shown are public keys. Private keys are not viewable with this command.

The command also shows the date and time at which the client (host) keys were installed.

6.5. Changing installed SSH keys

If your security policy requires you to change the client (host) SSH keys, you can achieve this with the following method.

1. Print the currently installed keys with the command `hsmadmin keys show`
2. Generate and install new client keys with the command `hsmadmin keys roll`. See SSH Client Key Protection for information about protection options that can be set on keys during generation.

The hardserver must be restarted in order to be able to use the new `ncoreapi` SSH client key after performing this operation, e.g. with `/opt/nfast/sbin/init.d-ncipher restart`. . Verify that the new keys have been installed with the command `hsmadmin keys show`

It is not possible to change the server (HSM) keys with this method. Should you be required to change the server keys, this can only be achieved by returning the unit to factory state with the `hsmadmin factorystate` command or by booting the HSM in recovery mode, see Recovery mode.

6.6. Making a backup of installed SSH keys

If your security policy allows it, make a backup of your private client key for the `sshadmin` service so that communication with the HSM can be re-established if the installed keys are erased or otherwise lost.

Do this with `hsmadmin keys backup` for verbatim copy of the `sshadmin` key with its existing protections (by default, it is tied to the host machine). Use `hsmadmin keys backup --passphrase` to backup the `sshadmin` key with a user-supplied passphrase so that it can be restored on another machine or after a re-installlation of the OS if necessary.

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

6.7. Restoring SSH keys from backup

If you erase or lose your SSH keys, communication with the HSM will be lost. If you have previously made a backup of those keys using `hsmadmin keys backup` you can restore that backup with `hsmadmin keys restore`. This command will restore the private client key for the `sshadmin` service and then create keys for all other services.

6.8. Preparing an HSM for use in another host

The client (host) SSH keys must be the same for every HSM connected to the same host. This will happen automatically if the HSMs are all installed together and are all in factory state. The `hsmadmin enroll` command installs the same client keys in each HSM.

Additional HSMs can be installed in a host at any time and, provided that the new modules are in factory state, the `hsmadmin enroll` command installs the same client keys in the new modules as are currently installed in any existing modules.

If it is necessary to be able to transfer a module from one host to another without returning it to factory state this can be achieved with the following method.

In the method below, the term 'source' refers to the host from which the module will be transferred and the term 'destination' refers to the host to which the module will be transferred.

1. Backup the private `sshadmin` client key from the destination host to a location that can be accessed by the source host, such as a shared drive or a USB stick, with the following command:

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

   Where `<FILE>` specifies the location of the shared drive or USB stick. You will be prompted to enter and confirm a passphrase to use to protect the key.

2. Install the destination host private sshadmin key on the source host with the following command:

   ```
   hsmadmin keys migrate --privkeyfile <FILE>
   ```

   Where `<FILE>` specifies the location of the file written in the previous step. You will be prompted to enter the passphrase of the key.

3. Remove the module from the source host and install in the destination host.

If the keys are not changed on the destination host, this step may be left indefinitely or until needed. For example, the module could be kept in storage as a cold standby unit.

4. After removing the module, run `hsmadmin enroll` on the source host to refresh the list of installed nShield 5s HSMs.
5. After installing the module, run `hsmadmin enroll` on the destination host.

7. Administration of platform services

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 n5s-ug-nix:hsmadmin:::ssh-key-prot.pdf and n5s-ug-nix:hsmadmin:::ssh-key-prot.pdf.

All of the platform services are administered by a unified utility called `hsmadmin`

7.1. hsmadmin

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

```
hsmadmin <subcommand>
```

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

7.1.1. 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.
`--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.
`--verbose`	Prints verbose logs.

7.1.2. 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 metadata in JSON format.
`--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.

7.1.3. hsmadmin npkginfo

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

It inspects an `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.

7.1.4. hsmadmin upgrade

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

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

This command takes the following parameters:

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

7.1.5. 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.

7.1.6. 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.

The install script calls this command automatically as `hsmadmin enroll --sshadmin-key /root/.ssh/id_nshield5_sshadmin`. This will generate SSH client keys and register them with the units if they have not been previously set up. If the `sshadmin` key is not found in its usual location under `/opt/nfast` then `/root/.ssh/id_nshield5_sshadmin` will be tried instead, so it is convenient to use `hsmadmin keys backup` to backup the key to this location.

If `hsmadmin enroll` is called after install to change the installed units, the hardserver will need to be restarted in order to pick up the configuration changes e.g. by running `/opt/nfast/sbin/init.d-ncipher restart`.

```
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.

7.1.7. 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

7.1.7.1. 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.

7.1.7.2. 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.

Parameter	Description
`--verbose`	Prints verbose logs.
`--privkeyfile`	Specifies the file containing the private key to be migrated to.

7.1.7.3. hsmadmin keys roll

This subcommand changes the client keys for all services.

See SSH Client Key Protection 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.

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

7.1.7.4. 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.

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.

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

7.1.7.5. 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`

7.1.8. hsmadmin logs

This command retrieves or clears logs from connected HSMs.

```
hsmadmin logs <subcommand>
```

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

- get
- clear

7.1.8.1. hsmadmin logs get

This subcommand retrieves logs from connected HSMs. The information in these logs is pri

marily intended for use by nShield Support.

If primary mode fails to boot, boot into recovery mode and retrieve the init log, then send the information it contains to nShield Support.

```
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.
`--esn`	Specifies the HSMs from which to retrieve logs.
`--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`

7.1.8.2. hsmadmin logs clear

This subcommand clears logs from connected HSMs. The HSM must be in maintenance mode for clearing logs.

```
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.

Parameter	Description
`--json`	Prints output in JSON format.
`--verbose`	Prints verbose logs.
`--log`	Selects log to be cleared. Options are `system`, `init`

7.1.9. 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.
`--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.

7.1.10. Manage the system clock of an nShield 5s

Ensuring that an HSM's system clock is closely synchronized with an external time source, such as the HSM's host clock, is critical for maintaining robust security and audit capabilities. The HSM's date and time are used to validate security certificate expiry dates and to provide accurate timestamps for security and audit logs, which are essential for official traceability. However, due to environmental differences and differences between the hard-

ware devices responsible for keeping time on the HSM and host sides, their time can drift apart over time if at least one of them is not periodically adjusted. `hsmadmin` has commands for managing system clock settings and synchronization. The first step in managing the HSM's system clock is to set the initial HSM time correctly. See settime.

Make sure that the date and time on the host machine are set correctly according to the documentation for the operating system on the host machine.

To prevent malicious actors from tampering with the HSM's system clock by moving it backward, the HSM is designed to prevent the setting of a time and date that is earlier than a previously set time and date. Only `hsmadmin factorystate` can reset the HSM's clock to an earlier time and date. This ensures that the HSM's system clock remains secure and accurate and helps prevent unauthorized access that could occur if the system clock were tampered with.

The second critical step in maintaining minimal drift between an HSM's host clock and its own clock is to periodically use the settime command with the adjust parameter. When the nShield 5s receives this command, it will work to gradually reduce any difference in time between the host and the HSM's clocks, preventing large jumps or discontinuities in time. The frequency at which the HSM's administrator should issue `hsmadmin settime --adjust` depends on multiple factors, for example the precision of the internal clock of the host of the HSM and the extent of drift between the host's clock and the HSM's clock. The recommended starting point for most systems would be to issue the command once per day. Experimentation is required to find the optimal frequency.

The settime command uses UTC date and time format.

7.1.11. hsmadmin settime

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

Without the `adjust` parameter, `hsmadmin settime` obtains the host local clock time in UTC format and sets the system date and time of the HSM.

To set system date and time, the HSM must be in maintenance mode.

Setting the system date and time automatically resets the HSM.

If the `adjust` parameter is used with `hsmadmin settime`, the HSM system clock drift will be

calibrated.

You can calibrate the clock drift when the HSM is in operational mode, but calibration will gradually converge the clocks at a few seconds per day. This means that it may take several days to correct the clock. When you execute `hsmadmin settime adjust`, the command immediately returns `HSM system time calibration in progress` to acknowledge that the calibration process has started. There is no notification when the calibration is complete.

```
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.
`--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.

7.1.12. 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.

7.1.13. 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.

Parameter	Description
`--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.

7.1.14. 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	Weal 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 excesive 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.

8. Client Software and module configuration

This chapter describes software and module configuration tasks that you can choose to perform after the initial installation of Security World Software and hardware. See the Installation Guide for more information about hardware and software installation.

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 Creating a Security World) and an OCS (as described in "Creating Operator Card Sets (OCSs).

8.1. About user privileges

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

8.2. Setting up client cooperation

You can allow an nShield HSM to automatically access the remote file system (RFS) belonging 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 hardware security modules that are not nShield HSMs:

1. Configure the RFS used by your nShield HSM to accept access by cooperating clients.
    - For every authenticated client (with write access and K_{NETI} authorization) that needs to be a client of this remote file system, run the command:

      ```
      rfs-setup --gang-client <client_IP_address> <EEEE-SSSS-NNNN> <keyhash>
      ```

      In this command:

        - `<client_IP_address>` is the IP address of the client.
        - `<EEEE-SSSS-NNNN>` is the ESN of the nToken used by the client when using a nToken K_{NETI} key to authenticate itself. When using software-based authentication, it must be empty (i.e. "") or can be omitted altogether.
        - `<keyhash>` is the hash of the software or module K_{NETI} key used by the client.
    - For every unauthenticated client (with write access but without K_{NETI} authoriza-

tion), run the command:

```
rfs-setup --gang-client --write-noauth client_IP_address
```

The `--write-noauth` option should be used only if you believe your network is secure. This option allows the client you are configuring to access the RFS without K_{NETI} authorization.

To limit a gang-client to read-only, use the `--readonly` flag.

2. On each client that is to be a cooperating client, you must run the `rfs-sync` command-line utility with appropriate options:
   - for clients using a software K_{NETI} key to authenticate themselves to the RFS, run the command with the default options:

     ```
     rfs-sync --setup <RFS_IP_ADDRESS>
     ```

   - for clients using a module K_{NETI} key to authenticate themselves to the RFS, run the command:

     ```
     rfs-sync --setup --authenticate --module=<MODULE> <RFS_IP_ADDRESS>
     ```

     In this command:

     - `<RFS_IP_ADDRESS>` is the IP address of the RFS.
     - `<MODULE>` is the local module to use for authentication.
   - for clients to authenticate the RFS using software-based authentication, use the `--rfs-hkneti=HKNETI` option to specify the hash of the software K_{NETI} key of the RFS.
   - for clients to authenticate the RFS using nToken authentication, use the `--rfs-esn=ESN` and `--rfs-hkneti=HKNETI` options to specify the `ESN` and hash of the K_{NETI} key of the nToken installed in the RFS.

The `rfs-sync` utility uses lock files to ensure that updates are made in a consistent fashion. If an `rfs-sync --commit` operation (the operation that writes data to the remote file system) fails due to a crash or other problem, it is possible for a lock file to be left behind. This would cause all subsequent operations to fail with a lock time-out error.

The `rfs-sync` utility has options for querying the current state of the lock file, and for deleting the lock file; however, we recommend that you do not use these options unless they are necessary to resolve this problem. Clients without write access cannot delete the lock file.

For more information about the `rfs-sync` utility, see rfs-sync.

To remove a cooperating client so the RFS no longer recognizes it, you must:

- Know the IP address of the cooperating client that you want to remove
- Manually update the `remote_file_system` section of the hardserver configuration file by removing the following entries for that particular client:

```
remote_ip=<client_IP_address>
remote_esn=keyhash=0000000000000000000000000000000000000000
native_path=/opt/nfast/kmdata/local
volume=kmdata-local
allow_read=yes
allow_write=yes
allow_list=yes
is_directory=yes
is_text=no
```

and

```
remote_ip=<client_IP_address>
remote_esn=keyhash=0000000000000000000000000000000000000000
native_path=/opt/nfast/kmdata/local/sync-store/
volume=kmdata-backup
allow_read=yes
allow_write=yes
allow_list=yes
is_directory=yes
is_text=no
```

8.2.1. Useful utilities

8.2.1.1. anonkneti

To find out the ESN and the hash of the K_{NETI} key for a given IP address, use the `anonkneti` command-line utility. A manual double-check is recommended for security.

The IP address could be one of the following:

- An IPv4 address, for example `123.456.789.123`.
- An IPv6 address, for example `fc00::1`.
- A link-local IPv6 address, for example `fe80::1%eth0`.
- A hostname.

8.2.1.2. rfs-sync

This utility synchronises the `opt/nfast/kmdata/local` folder between a cooperating client

and the remote file system it is configured to access. It should be run when a cooperating client is initialised in order to retrieve data from the remote file system and also whenever a client needs to update its local copy of the data or, if the client has write access, to commit changes to the data.

8.2.1.2.1. Usage

```
rfs-sync [-U|--update] [-c|--commit] [-s|--show] [--remove] [--setup [setup_options] ip_address]
```

8.2.1.2.2. Options

`-U|--update`

These options update local key-management data from the remote file system.

If a cooperating client has keys in its `kmdata/local` directory that are also on the remote file system, if these keys are deleted from the remote file system and then `rfs-sync --update` is run on the client, these keys remain on the client until manually removed.

`-c|--commit`

These options commit local key-management data changes to the remote file system, and update the client from the remote file system.

`-s|--show`

These options display the current synchronisation configuration.

`--setup`

This option sets up a new synchronisation configuration. Specifics of the configuration can be altered using `setup_options` as follows:

`-a|--authenticate`

This set-up option specifies the use of a module KNETI key to authenticate this client to the RFS. By default the software KNETI key is used instead.

`-m|--module=module`

This option selects the local module to use for authentication. The default is 1. This option can only be used with the `--authenticate` option.

`-p|--port=port`

These options specify the port on which to connect to the remote file system. The default is 9004.

`--rfs-hkneti=HNETI`

This option specifies the hash of the K_{NETI} key to use for nToken or software-based authentication of the RFS.

`--rfs-esn=ESN`

This options specifes the `ESN` of the nToken to use for authentication of the RFS.

`ip_address`

This option specifies the IP address of the remote file system, which could be one of the following:

- An IPv4 address, for example `123.456.789.123`.
- An IPv6 address, for example `fc00::1`.
- A link-local IPv6 address, for example `fe80::1%eth0`.
- A hostname.

`--remove`

This option removes the synchronisation configuration.

A client can use `rfs-sync --show` to display the current configuration, or `rfs-sync --remove` to revert to a standalone configuration. Reverting to a standalone configuration leaves the current contents of the Key Management Data directory in place.

The `rfs-sync` command also has additional administrative options for examining and removing lock files that have been left behind by failed `rfs-sync --commit` operations. Using the `--who-has-lock` option displays the task ID of the lock owner. As a last resort, you can use the `rfs-sync` command-line utility to remove lock files. In such a case, the `--kill-lock` option forcibly removes the lock file.

The lock file can also be removed via menu item 3-3-2, **Remove RFS Lock**: this executes the `rfs-sync --kill-lock` command.

8.2.2. 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. Environment variables

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 your self. `/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.

8.2.3. Logging and debugging

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

8.2.4. Configuring Java support for KeySafe

To use KeySafe, follow the instructions in Using KeySafe.

8.3. Configuring the hardserver

The hardserver handles secure transactions between the HSMs connected to the host com puter 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 the [`/opt/nfast/kmdata/config/` directory, 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 "Hardserver 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/` so that the configuration can be restored if necessary.
2. Edit the configuration file `/opt/nfast/kmdata/config/` to contain the required configuration. (See "Hardserver configuration files" for descriptions of the options in the configuration file.)
3. Run the `cfg-reread` command-line utility 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 hard-

> server.

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` com mand-line utility.

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

8.3.1. Overview of hardserver configuration file sections

8.3.1.1. Configuring remote HSM connections

A *remote HSM* is an HSM that is not connected directly to the host computer but with which the hardserver can communicate. It can be one of the following:

- A network-connected nShield HSM that is configured to use the host computer as a client computer
- An HSM to which an attended Remote Operator slot is imported for the hardserver's unattended local HSM

  (Remote Operator feature only).

You configure the hardserver's communications with remote HSMs in the `server_re-motecomms` 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.

8.3.1.2. 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.

8.3.1.3. 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.

8.3.1.4. 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.

8.3.1.5. 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.

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

8.3.1.6. 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 Enabling 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.

8.3.1.7. 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 Enabling optional features.

8.3.1.8. 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 Setting up 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.

8.3.1.9. 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.

8.3.2. 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.

8.3.2.1. 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.

8.3.2.2. 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.

8.3.2.2.1. 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.

8.3.2.3. 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. Refer to the *Installation Guide* for information on installing nShield hardware.
2. 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.

8.3.2.4. 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.

8.4. 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:

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

Similarly, you can start the hardserver on the client, and where applicable the Remote Administration Service, by running the following command

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

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

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

9. Enabling optional features

nShield HSMs support a range of optional features. Optional features must be enabled with a certificate that is supplied by Entrust. You can order features when you purchase a unit, or you can obtain them at a later date (from your Entrust account manager). Feature certificates are supplied as a file made available for download or requested as a smart (Activator) card, to be delivered by post. Features are enabled using the Feature Enable Tool.

Features provide additional functionality that must be enabled using the certificate file before the HSM can perform certain actions and use particular mechanisms. Features are either static or dynamic. Static features are persistent and remain enabled even if the HSM is factory stated or upgraded, most features are static. Conversely, dynamic features are non persistent and, if the HSM is factory stated, must be enabled again using the features file or activator card.

Dynamic features are identified in Available optional features. If a feature is not identified as dynamic it is a static feature.

Features that have been set in the factory will persist if the module is returned to factory state as described in Returning a module to factory state. Features that have been set with the Feature Enable tool will be lost if the unit is returned to factory state and must be enabled again when the HSM is returned to service.

For more information about:

- Ordering optional features, see Ordering additional features
- Feature-enabling procedures, see Enabling features.

The HSM checks to confirm whether any features that it attempts to use are enabled. It normally does this when it authorizes the commands or command options that relate to a specific feature.

After you have enabled features on a module, you must clear the module to make them available. Clear the module by running the command `nopclearfail --clear --all`

If you are enabling the Remote Operator feature, you must enable it on the HSM that is to be used as the unattended HSM.

For information about Remote Operator, see Remote Operator.

9.1. Available optional features

This section lists the features that can be added to the HSM. For details of all available features, contact Sales.

9.1.1. Elliptic Curve

Cryptography based on elliptic curves relies on the mathematics of random elliptic curve elements. It offers better performance for an equivalent key length than either RSA or Diffie-Hellman public key systems. Using RSA or Diffie-Hellman to protect 128-bit AES keys requires a key of at least 3072 bits. The equivalent key size for elliptic curves is only 256 bits. Using a smaller key reduces storage and transmission requirements.

Elliptic curve cryptography is endorsed by the US National Security Agency and NIST (the National Institute of Standards and Technology), and by standardization bodies including ANSI, IEEE and ISO.

nShield modules incorporate hardware that supports elliptic curve operations for ECDH (Elliptic curve Diffie-Hellman) and ECDSA (Elliptic Curve Digital Signature Algorithm) keys.

9.1.2. Elliptic Curve activation

All nShield HSMs require specific activation to utilize the elliptic curve features. HSMs use an activator smart card to enable this feature. Refer to Enabling features with a smart card for instructions on how to enable the EC feature. Additionally it is possible to activate the elliptic curve feature without a physical smart card. In this case the certificate details can be provided by email and entered locally. Refer to Enabling features without a smart card

Contact Sales if you require an EC activation.

nShield modules with elliptic curve activation support *MQV* (*Menezes-Qu-Vanstone*) modes.

9.1.3. Elliptic Curve support on the nShield product line

The following table details the range of nShield HSMs and the level of elliptic curve support that they offer.

HSM module type	**Elliptic Curve support**		**Elliptic Curve offload acceleration**[3]	
	Named curves[2]	Custom curves[1], [5]	Named curves[2]	Custom curves[1], [5]
nShield Edge (Windows only)	Yes	Yes	No	No

HSM module type	Elliptic Curve support		Elliptic Curve offload acceleration[3]	
nShield Solo 500 and 6000 nShield 500, 1500, and 6000	Yes	Yes	No	No
nShield Solo 500+, 6000+ nShield 6000+	Yes	Yes	Yes, Prime curves and twisted Brain pool curves are accelerated[4].	Yes
nShield Solo XC	Yes	Yes	Yes, Prime curves and both twisted and non-twisted Brainpool curves are accelerated[4].	Yes
nShield 5s	Yes	Yes	Yes, Prime curves and both twisted and non-twisted Brainpool curves are accelerated.	Yes

[1]Accessed via nCore, PKCS#11 and JCE APIs.

[2]Both Prime and Binary named curves are supported. Refer to Named Curves, below, which lists the most commonly supported elliptic curves.

[3]Offload acceleration refers to offloading the elliptic curve operation from the main CPU for dedicated EC hardware acceleration.

[4]Binary curves are supported, but are not hardware offload accelerated.

[5]Brainpool curves are supported as named curves via nCore, PKCS#11 and JCE only.

9.1.4. nShield software / API support required to use elliptic curve functions

	Security World Software for nShield	CodeSafe
Elliptic curve supported / API	Microsoft CNG, PKCS#11, Java Cryptographic Engine (JCE)[1].	Microsoft CNG, PKCS#11, Java Cryptographic Engine (JCE)[1].

[1]Java elliptic curve functionality is fully supported by the nShield security provider, nCipherKM. There is also the option to use the Sun/IBM PKCS #11 Provider with nCipherKM configured to use the nShield PKCS#11 library.

To demonstrate the accelerated performance of elliptic signing and verify operations, run the perfcheck utility. See *perfcheck: performance measurement checking tool*.

9.1.5. Named Curves

This table lists the supported named curves that are pre-coded in nShield module firmware.

Supported named curves			
ANSIB163v1	BrainpoolP160r1	NISTP192	SECP160r1
ANSIB191v1	BrainpoolP160t1	NISTP224	SECP256k1
	BrainpoolP192r1	NISTP256	
	BrainpoolP192t1	NISTP384	
	BrainpoolP224r1	NISTP521	
	BrainpoolP224t1	NISTB163	
	BrainpoolP256r1	NISTB233	
	BrainpoolP256t1	NISTB283	
	BrainpoolP320r1	NISTB409	
	BrainpoolP320t1	NISTB571	
	BrainpoolP384r1	NISTK163	
	BrainpoolP384t1	NISTK233	
	BrainpoolP512r1	NISTK283	
	BrainpoolP512t1	NISTK409	
		NISTK571	

9.1.6. Custom curves

nShield modules also allow the entry of custom elliptic curves which are not pre-coded in firmware. If the curve is Prime, it may benefit from hardware acceleration if supported by the nShield HSM (see nShield software / API support required to use elliptic curve functions, above).

Custom curves are supported by nCore and PKCS #11 APIs.

9.1.7. Further information on using elliptic curves

For more information on how to use elliptic curves, see the following sections:

- PKCS #11:

    - Mechanisms supported by PKCS #11: Mechanisms
- Symmetric and asymmetric algorithms: Cryptographic algorithms
- Using `generatekey` options and parameters to generate ECDH and ECDSA keys: Key generation options and parameters

Java elliptic curve functionality is fully supported by the nShield security provider, nCipherKM. There is also the option to use the Sun/IBM PKCS #11 Provider with nCipherKM configured to use the PKCS #11 library.

9.1.8. Secure Execution Engine (SEE)

The SEE is a unique secure execution environment. The SEE features available to you are:

SEE Activation (EU+10)	This SEE feature is provided with the CodeSafe developer product to enable you to develop and run SEE applications. The CodeSafe developer product is only available to customers in the Community General Export Area (CGEA, also known as EU+10). Contact Entrust to find out whether your country is currently within the CGEA.
SEE Activation (Restricted)	This SEE feature is provided with specific products that include an SEE application. This feature enables you to run your specific SEE application and is available to customers in any part of the world. This feature is a dynamic feature.

For more information about the SEE, see the *CodeSafe Developer Guide*.

9.1.9. Remote Operator support

Many Entrust customers keep critical servers in a physically secure and remote location. The Security World infrastructure, however, often requires the physical presence of an operator to perform tasks such as inserting cards. Remote Operator enables these customers to remotely manage servers running Security World Software using a secure nShield communications protocol over IP networks.

The Remote Operator feature must be enabled on the module installed in the remote server. Remote Operator cannot be enabled remotely on an unattended module.

For more information about using Remote Operator, see Remote Operator.

For v12 and later, Entrust recommends that you use Remote Administration, which is more

flexible than the Remote Operator functionality.

9.1.10. ISO smart card Support (ISS)

ISS, also called Foreign Token Open (FTO) allows data to be read to and written from ISO 7816 compliant smart cards in a manner prescribed by ISO7816-4. ISS allows you to develop and deploy a security system that can make full use of ISO 7816 compliant smart cards from any manufacturer.

9.1.11. Korean algorithms

This feature enables the following mechanisms:

- Korean Certificate-based Digital Signature Algorithm (KCDSA), which is a signature mechanism.

  KCDSA is used extensively in Korea as part of compliance with local regulations specified by the Korean government. For more information about the KCDSA, see the *nCore API Documentation*.

- SEED, which is a block cipher.
- ARIA, which is a block cipher.
- HAS160, which is a hash function.

9.1.12. Fast RNG for ECDSA

Utilise a faster alternative for Random Number Generation (RNG) for Elliptic Curve Digital Signature Algorithm (ECDSA). This feature is applicable only for nShield Solo XC, nShield Connect XC, nShield 5s, and nShield 5c.

The faster performance, comparable with v12.40 performance, is achieved by the RNG part of ECDSA being done on the NXP C291 Crypto Coprocessor.

This implementation of ECDSA uses an RNG that is not within scope for the nShield HSM certifications and for this reason it will not be used when the HSM is in a fips-140-level-3 or common-criteria-cmts Security World (regardless of the feature bit setting).

9.2. Ordering additional features

When you have decided that you require a new feature, you can order it from Sales. Before

you call Sales, collect information about your HSM as follows:

- If possible, make a note of the serial number. This can be found on the circuit board of the nShield module.

You can also get the serial number of the nShield HSM with `enquiry`:

```
physical serial   12-A30045
hardware part no  PCA10005-02 revision 02
hardware status   OK
```

- Run the `enquiry` command and note the Electronic Serial Number of the module.

You must provide the ESN number to order a new feature.

If you prefer, you can include this information in an e-mail to Sales. You can use the Feature Enable Tool to save the ESN details to a file. For more information about using the Feature Enable Tool, see Enabling features.

When your order has been processed, you receive a Feature Enabling Certificate in one of the following ways:

- Entrust e-mails you the Feature Enabling Certificate.
- Entrust sends you a smart card that contains the Feature Enabling Certificate.

The Feature Enabling Certificate contains the information that you need to enable the features you have ordered.

For more information, including pricing of features, telephone or email your nearest Sales representative using the contact details from this guide, or contact Entrust nShield Support, https://nshieldsupport.entrust.com.

9.3. Enabling features

9.3.1. Viewing enabled features

The **Feature Enable Tool** can be used to view the status of modules connected to the host or to confirm that a feature has been successfully enabled on all modules connected to the host. To view the status of features, run the tool without a smart card.

Some features do not appear in the default output from the **Feature Enable Tool** because they are no longer sold. To see the status of all features, run `fet --show-all`.

9.3.2. Enabling features with a smart card

When it is launched, the **Feature Enable Tool** automatically scans the smart card readers of all modules attached to a host computer for any Feature Enabling smart cards present in the smart card readers, including imported Remote Operator slots and Dynamic Slots. However, feature enable smart cards do not work in Dynamic Slots.

To enable a new feature with a Feature Enabling smart card from Entrust:

1. Insert the Feature Enabling card from Entrust into a slot available to the module to be updated, excluding any Dynamic Slots.
2. Run the `fet` command-line utility to start the **Feature Enable Tool**.

A message is displayed if the features are enabled successfully. If you do not see this message confirming a successful upgrade, see Enabling features without a smart card.

9.3.3. Enabling features without a smart card

The **Feature Enable Tool** can also obtain the Feature Enabling Certificate information supplied by Entrust from a file or from the keyboard.

When you run the **Feature Enable Tool** without a Feature Enabling smart card in an HSM slot, a message similar to the following is displayed. There is a line for the features on each module, and a list of options.

In this example, only one module (`ESN 15C8-4387-C748`) is attached to the host.

```
                      Feature Enable Tool
                      ===================

                    ISO Smart Card Support
                    |  Remote Operator
                    |  |  Korean Algorithms
                    |  |  |  SEE Activation (EU+10)
                    |  |  |  |  SEE Activation (Restricted)
                    |  |  |  |  |  Elliptic Curve algorithms
                    |  |  |  |  |  |  Elliptic Curve MQV
                    |  |  |  |  |  |  |  Fast RNG for ECDSA
                    |  |  |  |  |  |  |  |  HSM Speed Rating
Mod   Electronic    |  |  |  |  |  |  |  |  |
No.  Serial Number
 1  15C8-4387-C748 -- Y  Y  Y  N  N  Y  Y  N  Mid Speed

Reading card in slot 0 of module 1.

0. Exit Feature Enable Tool.
1. Read FEM certificate(s) from a smart card or cards.
2. Read FEM certificate from a file.
3. Read FEM certificate from keyboard.
4. Write table to file.

Enter option :
```

10. Security World Remote Administration

Gathering a quorum of card holders to carry out card holder duties in a remote datacenter can be expensive and inconvenient. Remote Administration enables Administrators and Operators to present their cards remotely to authorize HSM operations without being phys ically present at the HSM.

When presenting a card, a secure channel is formed directly between the Remote Administration smart card and the target HSM before any token shares are read from or written to the smart card.

Remote Administration enables Administrators to use their remote access solution to perform administration operations and extends the operations that can be performed in this way.

Remote Administration enables:

- Card holders to present smart cards to an HSM without being physically present at the HSM (e.g. the card holder may be in an office, while the HSM is in a datacenter)
- All Administrator and Operator card operations to be carried out remotely
- Security World programs and utilities to be run remotely when used in combination with a standard remote access solution
- Full remote administration of Security Worlds and their HSMs including:
  - Remote mode change
  - Create/load/unload Security World
  - Firmware upgrade
  - Module status (SOS) reporting

Once the software has been installed and the hardware security modules have been configured, Remote Administration enables full remote administration of Security Worlds and their HSMs.

10.1. Remote Administration components

Remote Administration consists of a number of components:

10.1.1. Remote Administration software

The following software is needed to allow remote card readers to be associated with an HSM:

- nShield Remote Administration Client software
  Must be installed on the computer that has the card reader attached. See Remote Administration Client for more information.
- nShield Remote Administration Service software
  Must be installed where it can access the appropriate HSM to provide communications between the card in the card reader and the HSM. See the *Installation Guide* for your nShield HSM for more about where to install Remote Administration Service software.

  The Remote Administration Service should be installed on the host machine of your HSMs and this machine must be accessible to Remote Administration Clients.

  See Remote Administration Service for more information.

When a card is inserted in a reader that is associated with an HSM, the nShield Remote Administration Client and the Remote Administration Service convey messages between the card and the HSM, allowing a secure channel of communications to be established.

10.1.2. Security World programs and utilities

The Security World programs and utilities are typically installed on a computer within your datacentre. In such cases the Remote Administration feature assumes you will use your pre ferred remote access solution, e.g. SSH or Remote Desktop, to run the Security World programs and utilities remotely. This means you can run a utility like `createocs` from a remote location and present the OCS to be created using a Remote Administration Client. The Remote Administration feature includes the ability to change the mode of HSMs remotely using the `nopclearfail` utility. This means it is possible to create a Security World remotely and perform firmware upgrades.

10.1.3. nShield Remote Administration smart cards

You must use nShield Remote Administration Cards with Remote Administration. These are smart cards that are capable of negotiating cryptographically secure connections with an HSM, using warrants as the root of trust. nShield Remote Administration Cards can also be used in the local slot of an HSM if required.

The nShield Remote Administration smart cards provide:

- Storage and retrieval of logical token fragments, similar to the smart cards used with previous releases
- Security mechanisms to ensure authentication and confidentiality of data transferred between itself and the HSM

The nShield Remote Administration smart cards are FIPS 140 Level 3 certified devices, supporting execution of a custom Java Applet developed by Entrust. The smart cards used with previous versions of Security World software and nShield HSMs are still useable but, as previously, only in an HSM's local slot. Remote Administration smart cards can be used both remotely and in an HSM's local slot.

The use of nShield Remote Administration Cards is controlled by an Authorized Card List. If a card does not appear in the list, it cannot be used. See Authorized Card List for more information.

Existing Administrator smart cards can be migrated to new Remote Administration smart cards using the `racs` (replace administrator card set) utility.

When using the `racs` utility, you cannot redefine the quantities in a K of N relationship for an ACS. The K of N relationship defined in the original ACS persists in the new ACS.

Similarly, existing OCS can be migrated using the `rocs` (replace operator card set) utility, provided the Security World has recovery enabled and the keys protected by that OCS are recovery enabled.

10.2. Authorized Card List

The use of nShield Remote Administration smart cards, both remotely and in an HSM's local slot, is controlled by an Authorized Card List. If the serial number of a card does not appear in the Authorized Card List, it cannot be used by the system. The list only applies to Remote Administration smart cards.

By default, the Authorized Card List is empty following software installation. The serial num bers of Remote Administration smart cards must be added to the list using a text editor before they can be used.

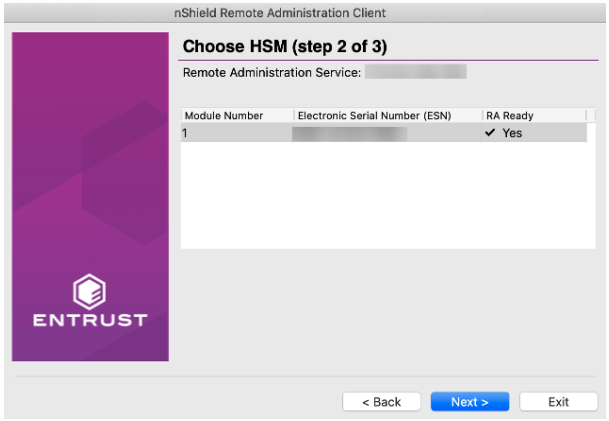
For more information on the Authorized Card List, see Authorized Card List.

10.3. Remote Administration Client

The Remote Administration Client (RAC) is a utility that enables you to select an HSM located elsewhere from a list provided by the Remote Administration Service (RAS), and as-

sociate an nShield Trusted Verification Device attached to your computer with the HSM.

The RAC GUI (usually running on a laptop or workstation) communicates with the RAS (in a datacenter) over a standard TCP/IP connection. If the RAC computer is not on the same local network as the RAS computer, Entrust recommend that the connection is made over a VPN.



In the example screen shown, an HSM will not be **Remote Administration (RA) Ready** until it has the appropriate firmware, and has one or more dynamic Slots configured.

For users who want to script the card presentation process, there is also a command line utility, `raccmd`.

See the nShield Remote Administration Client *User Guide* for more information on deploying and using the Remote Administration GUI or command line utility.

Windows 8.1 + only

If you disconnect the TVD while you are on the **Use Card Reader** screen the Windows Smart Card service `SCardSvr` displays an error and terminates.

10.4. Remote Administration Service

The Remote Administration Service (RAS) provides a bridge between the RAC and the back end HSMs (via the hardserver). Its functionality is to:

- Manage connections from multiple RACs
- Supply a list of available HSMs to the connected RACs
- Negotiate a connection to an HSM via the hardserver and route messages between the RAC and destination HSM.

To use Remote Administration with an nShield HSM, the Remote Administration Service must be installed on the host where the hardserver or nShield HSM reside. If you have multi

ple HSM hosts in a Security World, the Remote Administration Service must be installed on each one. See the *Installation Guide* for the HSMs for further details.

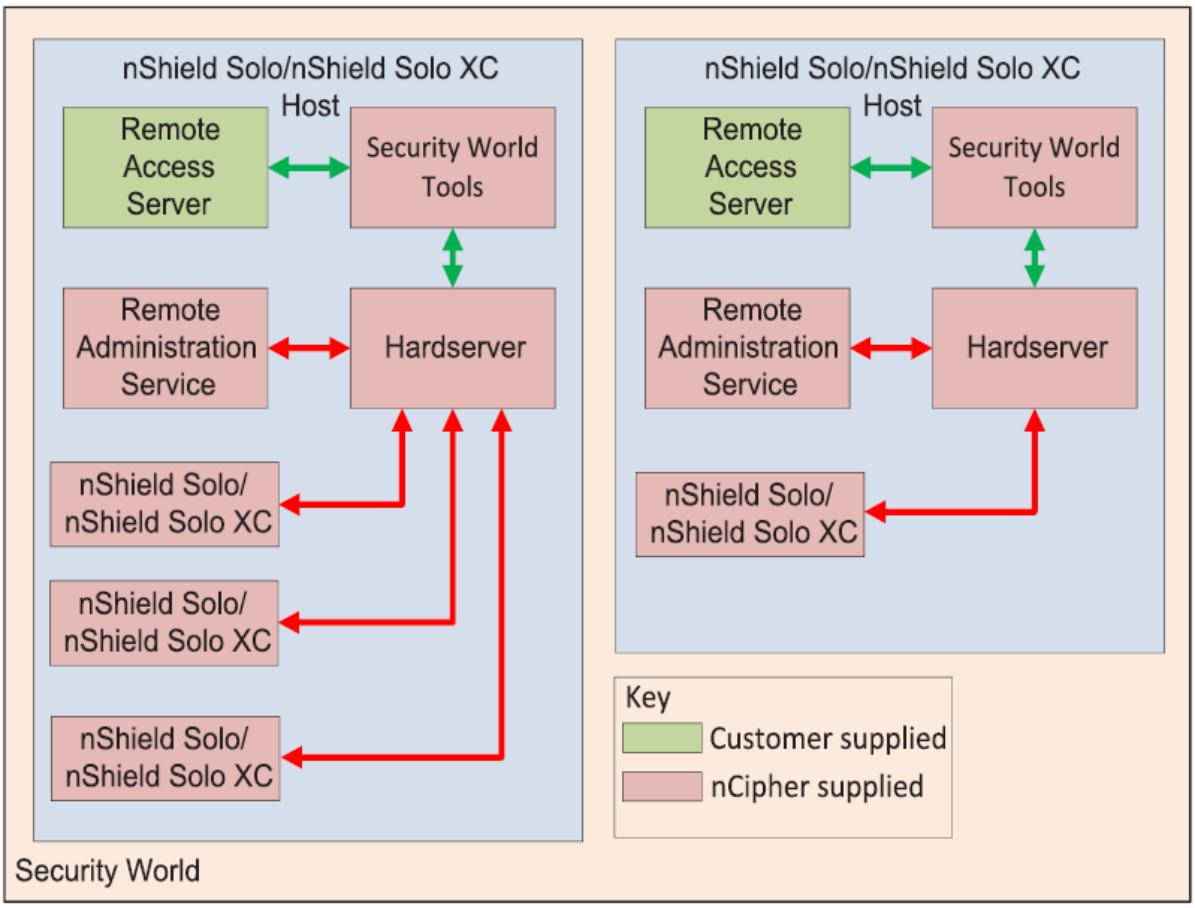
10.5. nShield Trusted Verification Device

Entrust supply and recommend the use of the nShield Trusted Verification Device (TVD). This is an intelligent smart card reader that blocks any malware on the client machine from spoofing the HSM identity passed to the nShield Remote Administration smart card. The TVD allows the card holder to securely confirm the Electronic Serial Number (ESN) of the HSM to which they want to connect, using the Trusted Verification Device display.

For more information, see the *Trusted Verification Device (TVD) User Guide*.

10.6. Software installation

10.6.1. The Remote Administration Service with the nShield HSM



The Remote Administration Service must be installed on the host where the hardserver and the HSM reside. If you have multiple HSM hosts in a Security World, see the *User Guide* for the HSMs for further details.

nShield Remote Administration Cards cannot be used until their serial numbers have been added to the Authorized Card List. See the *nShield Remote Administration User Guide* for further details.

10.6.2. Remote Administration Service bundle

The Remote Administration Service (RAS) is provided through the Remote Administration Service bundle and needs to be installed in the default directory.

For information on installing the Remote Administration Service bundle, see the *installation guide* for your HSM.

10.6.3. Remote Administration Client

The Remote Administration Client is normally deployed on its own using the instructions in the *nShield Remote Administration Client* user guide but it can be deployed on a client at

the same time as rest of nShield software

For more information on the Remote Administration Client, see the *nShield Remote Administration Client* user guide.

10.6.4. TVD

A nShield Remote Administration Client can connect to one nShield TVD during a session.

For information on installing the TVD driver and confirming the HSM Electronic Serial Number (ESN) using the nShield TVD, see the *nShield Remote Administration Client* user guide.

10.7. System configuration

10.7.1. Remote Administration Service port

The port used by Remote Administration Clients to access the Remote Administration Service can be changed by setting the `port` field in the *remote_administration_service_slot_server_startup* section of the hardserver configuration file, see [remote_administration_service_startup].

10.7.2. Stopping and restarting the Remote Administration Service

The Remote Administration Service can be stopped and started using a command window.

```
$ systemctl stop nc_raserv
$ systemctl start nc_raserv
```

10.7.3. Firewall settings

Assuming there is a firewall to protect your Remote Administration Service, open the port given in the *Firewall settings* section of the Installation Guide for the HSM.

To support Remote Administration, HSMs have to be configured to support between 1 and 16 Dynamic Slots. These Dynamic Slots are virtual card slots that can be associated with a card reader connected to a remote computer. Dynamic Slots are in addition to the local slot of an HSM and any soft token slot that may be available.

The default number of slots is 0. This disables Remote Administration on the relevant HSM.

1. Do the following:
   a. Use the `dynamic_slots` section in the hardserver configuration file to define the number of Dynamic Slots for each relevant HSM.
2. Clear the HSM for the changes to take effect.

   For example, run the `nopclearfail` command:

   ```
   nopclearfail --clear --all
   ```

You can check that the HSM has Dynamic Slots by:

- Running the command:

  ```
  slotinfo -m 1
  ```

  For example, if four Dynamic Slots have been configured, the output from this command includes the lines:

  ```
  Slot    Type            Token   IC  Flags   Details
  #0      Smartcard       -       1   A
  #1      Software Tkn    -       0
  #2      Smartcard       -       0   AD
  #3      Smartcard       -       0   AD
  #4      Smartcard       -       0   AD
  #5      Smartcard       -       0   AD
  ```

- The `D` in the Flags column indicates that slots `#2` to `#5` are Dynamic Slots.

Depending upon your system configuration, it can take up to 30 seconds for the Dynamic Slots to appear.

10.8. Adjusting card removal detection timers to account for network characteristics

Depending upon the characteristics of the network between nShield Remote Administration Clients and HSMs, you may need to adjust the timers that determine how long the system waits for a response, before it regards a card as having been removed. This enables you to balance the assured card removal detection time and network traffic.

Do the following:

- Use The `dynamic_slot_timeouts` section in the module configuration file to define the round trip (HSM to smartcard and back) time limit (the default is 10 seconds), and the card removal detection timeout (the default is 30 seconds).

- Push the updated configuration file to the nShield HSM.

10.9. Using Remote Administration with applications requiring cards in slot 0

If you want to use Remote Administration, but have an application that expects cards to be presented in slot 0, you must configure a slot mapping for each affected HSM.

1. Do the following:
   a. Use the `dynamic_slots` section in the hardserver configuration file to define the number of Dynamic Slots for each relevant HSM.

      See *dynamic_slots* for more about the `dynamic_slots` section.

You can check the mapping by:

- Running the command:

  ```
  slotinfo -m 1
  ```

  For example, if dynamic slot #2 has been mapped to slot #0, the output from this command includes the lines:

  ```
  Slot Type Token IC Flags Details
  #0 Smartcard - 1 AD
  #1 Software Tkn - 0
  #2 smartcard - 0 A
  ```

- The `D` in the `Flags` column indicates that slot `#0` is now a Dynamic Slot

10.10. Authorized Card List

The use of nShield Remote Administration smart cards is controlled by an Authorized Card List. If the serial number of a card does not appear in the Authorized Card List, it is not recognized by the system and cannot be used. The list only applies to Remote Administration cards and is used when a card is inserted:

- In the local slot of an HSM
- In a card reader that is associated with a dynamic slot of the HSM, through the nShield Remote Administration Client

By default, the Authorized Card List is empty following software installation. The serial num

bers of Remote Administration Cards must be added to the list before they can be used.

The Authorized Card List is a text file `/opt/nfast/kmdata/config/cardlist` on the RFS and each client computer. The file is read from the RFS by associated nShield HSMs as and when required for front panel operations. The list applies to all nShield network-attached HSMs associated with the RFS, regardless of the Security World to which an HSM may belong, including when creating a Security World from the front panel. For client initiated card operations the Authorized Card List file on that client computer is used. The RFS and client copies of the Authorized Card List have to be kept in step manually.

10.10.1. Adding cards to the Authorized Card List

Add the serial numbers (16 digits with no separators) of all Remote Administrator Cards you intend to use to the Authorized Card List, with a standard text editor. The serial numbers are printed on the smart cards and are reported by using `slotinfo -m1 -s0` when the card is in a slot, where **1** is the number of the HSM and **0** is the number of the slot.

There is an option to allow any Remote Administration Card to be used, by including a wildcard (*) in the Authorized Card List. Entrust recommends that you do not use this option, except under controlled circumstances, as it effectively disables the Using Remote Administration control.

10.11. Using Remote Administration

A privileged client can run the Command Line Tools remotely to:

- Change the mode of the HSM using `nopclearfail –M/-O/-I` to set the mode of the MOI switch, see *Remote mode switch*

10.11.1. Presenting nShield Remote Administration smart cards using the Remote Administration Client

With Remote Administration, you present a smartcard in a remote work station or laptop rather than locally at the nShield HSM. Remote Administration creates a separate secure connection from the Remote Administration smart card to the nShield HSM enabling remote card presentation.

For information on presenting nShield Remote Administration smart cards, see the *nShield® Remote Administration Client* user guide.

10.11.2. Remote Administration Configuration file sections

The following sections relevant to Remote Administration are included in the hardserver configuration file:

10.11.2.1. [dynamic_slot_timeouts]

```
# Start of the dynamic_slot_timeouts section
# Timeout values used to specify expected smartcard responsiveness for all
# modules on the network.
# Each entry has the following fields:
#
# Round trip time limit, in seconds, is how long to wait before giving up due
# to network delays. (default=10)
# round_trip_time_limit=INT
#
# Maximum time, in seconds, that can pass without a response from the
# smartcard before considering it removed and unloading all associated secrets
# (default=30)
# card_remove_detect_time_limit=INT
```

The dynamic_slot_timeout section is in the hardserver configuration file for the HSM.

10.11.2.2. [dynamic_slots]

```
# Start of the dynamic_slots section
# The dynamic smartcard slots that the modules should provide for the use of
# administrators who do not have physical access to the module hardware
# Each entry has the following fields:
#
# ESN of the module to be configured with dynamic slots.
# esn=ESN
#
# Number of dynamic slots the module will support. (default=0)
# slotcount=INT
```

The dynamic_slots section is in the hardserver configuration file for the HSM.

10.11.2.3. [slot_mapping]

```
# Start of the slot_mapping section
# Slot remapping configuration.
# Each entry has the following fields:
#
# ESN of the module on which slot 0 will be remapped with another.
# esn=ESN
#
# Slot to exchange with slot 0. Setting this value to 0 means do
# nothing.(default=0)
```

```
# slot=INT
```

The slot_mapping section is in the hardserver configuration file for the HSM.

Mapping a Dynamic Slot to slot 0 is needed if you want to use Remote Administration with applications that are not aware of slot numbers greater than zero. This applies to KeySafe and CNG Wizard but may also apply to your own applications.

10.11.2.4. [remote_administration_service_startup]

```
# Start of the remote_administration_service_startup section
# Remote Administration Service communication settings, these are only read at
# Remote Administration Service startup time
# Each entry has the following fields:
#
# The port for the Remote Administration Service to listen on for incoming TCP
# connections from remote administration clients (default=9005)
# port=PORT
```

11. Creating and managing a Security World

This chapter describes how to create and manage a Security World. You must create a Security World before using the HSM to manage keys.

You normally create a Security World after installing and configuring the module and its software. For more information, see:

- The Installation Guide for more about installing the module and software.
- Client Software and module configuration

You create a Security World with a single HSM. If you have more than one module, select one module with which to create the Security World, then add additional modules to the Security World after its creation. For more information, see Adding or restoring an HSM to the Security World. If you create a Security World with the audit logging feature enabled, all additional HSMs added to this Security World will also have audit logging enabled.

To use the module to protect a different set of keys, you can replace an existing Security World with a new Security World.

For more information about the type of user that is required for different operations, see About user privileges.

All Security Worlds rely on you using the security features of your operating system to control the users who can access the Security World and, for example, write data to the host.

11.1. Creating a Security World

You can use the following to create Security Worlds:

- The `new-world` command line utility
  See Creating a Security World using new-world.

11.1.1. The creation process

When you create a Security World:

- The HSM is erased
- A new HSM key for this Security World is generated
- A new ACS to protect this HSM key is created

- The Security World information is stored on hard disk of the host computer
  - The information is encrypted using the secrets stored on the ACS
- The HSM and Security World are configured for Audit Logging if selected

If you want to re-use the physical cards created in a previous Security World, you must erase all Operator Cards, except for nShield Remote Administration Cards, while the previous Security World still exists. See Erasing cards and softcards.

We recommend that you regularly back up the entire contents of the RFS. Either the `%NFAST_KMDATA%` directory on Windows, or the `kmdata` directory on Linux, is required to restore an nShield HSM or its replacement, to the current state in case of failure.

Due to the additional primality checking required by SP800-131A, Security World generation will take longer when using the new default Ciphersuite (from v12.40 onwards) - on nShield USB-attached HSMs, this could be up to 45 minutes.

11.1.2. Security World Files

The Security World infrastructure stores encrypted key material and related data in files on the host. For multiple hosts to use the same Security World, the system administrator must ensure that these files are copied to all the hosts and updated when required.

11.1.2.1. Location of Security World files

The logic for finding the security world data directory is:

1. If `NFAST_KMLOCAL` is set, use that.
2. Otherwise, if `NFAST_KMDATA` is set, use `${NFAST_KMDATA}/local` on Linux, `%NFAST_KMDATA%\local` on Windows.
3. Otherwise, if `NFAST_HOME` is set, use `${NFAST_HOME}/kmdata/local` on Linux, `%NFAST_HOME%\kmdata\local` on Windows.
4. Otherwise, use `/opt/nfast/kmdata/local` on Linux, `C:\nfast\kmdata\local` on Windows.

By default, the Key Management Data directory, and sub-directories, inherit permissions from the user that creates them. Installation of the Security World Software must be performed by a user with Administra-

> tor rights that allow read and write operations, and the starting and stop ping of applications.

Security World operations create or modify Security World files as follows:

Operation	creates/modifies	file(s)
Create a Security World	creates	`world` (for each module in the Security World) `module_ESN`
Load a Security World	creates or modifies	(for each module in the Security World) `module_ESN`
Replace an ACS	modifies	`world`
Create an OCS	creates	`card_HASH` `cards_HASH_NUMBER`
Create a softcard	creates	`softcard_HASH`
Generate a key	creates	`key_APPNAME__IDENT`
Recover a key	modifies	`key_APPNAME` (for each key that has been recovered)

- <ESN> - Electronic serial number of the module on which the Security World is created.
- <IDENT> - Identifier given to the card set or key when it is created.
- <NUMBER> - Number of the card in the card set.
- <APPNAME> - Name of the application by which the key was created. It's a 40-character string that represents the hash of the card set's logical token. It's either user supplied or a hash of the key's logical token, depending on the application that created the key.

11.1.2.2. Required files

The following files must be present and up to date in the `/opt/nfast/kmdata/local` directory, or the directory specified by the `NFAST_KMLOCAL` environment variable, for a host to use a Security World:

- `world`
- A `module_ESN` file for each module that this host uses
- A `cards_<IDENT>` file for each card set that is to be loaded from this host

- A `card_<IDENT>_NUMBER` file for each card in each card set that is to be loaded from this host
- A `key_<APPNAME>_<IDENT>` file for each key that is to be loaded from this host.

These files are not updated automatically. You must ensure that they are synchronized whenever the Security World is updated on the module.

11.1.3. Security World options

Decide what kind of Security World you need before you create it. Depending on the kind of Security World you need, you can choose different options at the time of creation. For convenience, Security World options can be divided into the following groups:

- Basic options, which must be configured for all Security Worlds
  - Optionally enable Audit Logging for the Security World
- Recovery and replacement options, which must be configured if the Security World, keys, or passphrases are to be recoverable or replaceable
- SEE options, which only need be configured if you are using CodeSafe
- Options relating to the replacement of an existing Security World with a new Security World.

Security World options are highly configurable at the time of creation but, so that they will remain secure, not afterwards. For this reason, we recommend that you familiarize yourself with Security World options, especially those required by your particular situation, before you begin to create a Security World.

11.1.3.1. Security World basic options

When you create a Security World, you must always configure the basic options described in this section.

11.1.3.1.1. Cipher suite

Only one Cipher suite is supported and this is SP800-131 compliant.

11.1.3.1.2. ACS quorum

You must decide the total number of cards (*N*) in a Security World's ACS and must have that many blank cards available before you start to create the Security World. You must also decide how many cards from the ACS must be present (*K*) when performing administrative

functions on the Security World.

We recommend that you do not create ACSs for which *K* is equal to *N*, because you cannot replace such an ACS if even 1 card is lost or damaged.

In Common Criteria CMTS Security Worlds the minimum value of K for the ACS is 2.

In many cases, it is desirable to make *K* greater than half the value of *N* (for example, if *N* is 7, to make *K* 4 or more). Such a policy makes it harder for a potential attacker to obtain enough cards to access the Security World. Choose values of *K* and *N* that are appropriate to your situation.

The total number of cards used in the ACS must be a value in the range 1 – 64.

11.1.3.1.3. FIPS 140 Level 3 compliance

By default, Security Worlds are created to comply with the roles and services, key management, and self-test sections of the FIPS 140 standard at Level 2. However, you can choose to enable compliance with the FIPS 140 standard at Level 3.

This option provides compliance with the roles and services of the FIPS 140- 2 Level 3 standard. It is included for those customers who have a regulatory requirement for compliance.

If you enable compliance with FIPS 140 Level 3 roles and services, authorization is required for the following actions:

- Generating a new OCS
- Generating or importing a key, including session keys
- Erasing or formatting smart cards (although you can obtain authorization from a card you are about to erase).

In addition, you cannot import or export private or symmetric keys in plain text.

11.1.3.1.4. UseStrongPrimes Security World setting

From firmware version 12.70, the nShield HSM always targets FIPS 186-4 compliance when generating RSA keys of 1024 bits or more. It typically does this using a "strong primes" strategy, however Entrust only guarantees this strategy if the `UseStrongPrimes` setting is enabled.

If your firmware is version 12.70 or higher, you do not need this setting enabled for FIPS 186-4 compliance.

If you are using an older version of firmware, meaning it has a version number *lower than* 12.70, then you need the `UseStrongPrimes` setting enabled to grant FIPS 186-2 compliance.

If your Security World is FIPS 140 Level 3, then this setting is on by default. If your Security World is not FIPS 140 Level 3, then you can disable the `UseStrongPrimes` setting for faster RSA key generation, however this removes FIPS 186-2 compliance.

11.1.3.1.5. Remote Operator

To use a module without needing physical access to present Operator Cards, you must enable the Remote Operator feature on the module. For more information, see Enabling optional features.

By default, modules are initialized into Security Worlds with remote card set reading enabled. If you add a module for which remote card reading is disabled to a Security World for which remote card reading is enabled, the module remains disabled.

11.1.3.2. OCS and softcard replacement

By default, Security Worlds are created with the ability to replace one OCS or softcard with another. This feature enables you to transfer keys from the protection of the old OCS of softcard to a new OCS or softcard.

You can replace an OCS with another OCS, or a softcard with another softcard, but you cannot replace an OCS with a softcard or a softcard with an OCS. Likewise, you can transfer keys from an OCS to another OCS, or from a softcard to another softcard, but you cannot transfer keys from an OCS to a softcard or from a softcard to an OCS.

You can choose to disable OCS and softcard replacement for a Security World when you create it. However, in a Security World without this feature, you can never replace lost or damaged OCSs; therefore, you could never recover the keys protected by lost or damaged OCSs, even if the keys themselves were generated as recoverable (which is the default for key generation).

OCS and softcard replacement cannot be enabled after Security World creation without reinitializing the Security World and discarding all the existing keys within it.

For an overview of Security World robustness and OCS or softcard replacement, see Replacing an Operator Card Set or recovering keys to softcards. For details about performing OCS and softcard replacement operations, see Replacing Operator Card Sets and Replacing the Administrator Card Set.

11.1.3.3. passphrase replacement

By default, Security Worlds are created so that you cannot replace the passphrase of a card or softcard without knowing the existing passphrase.

However, you can choose to enable passphrase replacement at the time you create a Security World. This option makes it possible to replace the passphrase of a card or softcard even if you do not know the existing passphrase. Performing such an operation requires authorization from the Security World's ACS.

For details about performing passphrase replacement operations, see Changing unknown or lost passphrase.

11.1.3.4. Nonvolatile memory (NVRAM) options

Enabling nonvolatile memory (NVRAM) options allows keys to be stored in the module's NVRAM instead of in the Key Management Data directory of the host computer. Files stored in the module's non-volatile memory have Access Control Lists (ACLs) that control who can access the file and what changes can be made to the file. NVRAM options are relevant only if your module's firmware supports them, and you can store keys in your module's NVRAM only if there is sufficient space.

When the amount of information to be stored in the NVRAM exceeds the available capacity, you can instead store this data in a blob encrypted with a much smaller key that is itself then stored in the NVRAM. This functionality allows the amount of secure storage to be limited only by the capacity of the host computer.

11.1.3.5. Security World SEE options

You must configure `SEE options` if you are using the nShield Secure Execution Engine (SEE). If you do not have SEE installed, the SEE options are irrelevant.

11.1.3.5.1. SEE debugging

SEE debugging is disabled by default, but you can choose whether to enable it for all users

or whether to make it available only through use of an ACS. In many circumstances, it is use ful to enable SEE debugging for all users in a development Security World but to disable SEE debugging in a production Security World. Choose the SEE debugging options that best suit your situation.

11.1.3.5.2. Real-time clock (RTC) options

Real-time clock (RTC) options are relevant only if you have purchased and installed the CodeSafe Developer kit. If so, by default, Security Worlds are created with access to RTC operations enabled. However, you can choose to control access to RTC operations by means of an ACS.

11.1.3.6. Security World replacement options

Options relating to Security World replacement are relevant only if you are replacing a Secu rity World.

If you replace an existing Security World, its `/opt/nfast/kmdata/local` directory is not over written but renamed `/opt/nfast/kmdata/local_N` (where *N* is an integer assigned depend- ing on how many Security Worlds have been previously saved during overwrites). A new Key Management Data directory is created for the new Security World. If you do not wish to retain the `/opt/nfast/kmdata/local_N` directory from the old Security World, you must delete it manually.

11.1.4. Creating a Security World using new-world

11.1.4.1. Before you start

Before you start to create a Security World:

- The HSM must be in pre-initialization mode. See Checking and changing the mode on an nShield 5s module for more about changing the mode.
- You must be logged in to the host computer as `root` or as a user in the group `nfast`. For more information, see server_settings.
- If you have installed the Security World Software in a directory other than `/opt/nfast/`, you must have created a symbolic link from `/opt/nfast/` to the directory in which the software is actually installed.
- Before configuring the Security World, you should know:

- The security policy for the HSM
- The number and quorum of Administrator Cards and Operator Cards to be used

To help you decide on the Security World you require, see Security World options.

- You must have enough smart cards to form the Security World's card set.

When you have finished creating a Security World, you must change the mode to "Operational" using `nopclearfail -I m1` or `nopclearfail -O -m1`.

Follow the directions in this section to create a Security World from the command line with the `new-world` utility.

11.1.4.2. Running the new-world command-line utility

Open a command prompt window and type the command `new-world` using the options in the table.

The example below will create a Security World supporting FIPS140 Level 3 with a ACS quorum of 3/5 and with audit logging enabled.

```
new-world --mode=fips-140-level-3 --acs-quorum=3/5 --audit-logging
```

In this command:

Option	Description
`--initialize`	This option creates a new Security World, replacing any existing `/opt/nfast/kmdata/local/` directory.  Replacing an existing Security World in this way does not delete the Security World's host data and recovery and replacement data, but renames the existing `/opt/nfast/kmdata/local/` directory in which these reside as `/opt/nfast/kmdata/localN` (where *N* is an integer assigned depending on how many Security Worlds have been previously saved during overwrites).
`--factory`	This option erases an HSM, restoring it to factory state.
`--no-remoteshare-cert`	This option prevents making the HSM from becoming a target for remote shares.
`--no-strict-rsa-keygen`	If you have not specified a mode parameter you can use the `-no-strict-rsa-keygen` flag to disable the UseStrongPrimes setting. Otherwise it will be enabled by default. See UseStrongPrimes Security World setting.

Option	Description
`--mode=MODE`	`FIPS-140-level-3` creates a Security World compliant with FIPS 140 Level 3. `common-criteria-cmts` creates a Security World supporting *Common Criteria PP 419 221-5*. Omitting this option will create a default Security World compliant with FIPS 140 Level 2.
`--no-recovery`	This option disables the ability to recovery or replace OCSs and softcard (which is otherwise enabled by default). This is equivalent to setting `!r`, where the `!` operator instructs the system to turn off the specified feature (`r`). By default, `new-world` creates key recovery and replacement data that is protected by the cryptographic keys on the ACS. This option does not give Entrust or any other third party access to your keys. Keys can only be recovered if authorization from the ACS is available. We recommend that you leave OCS and softcard recovery and replacement functionality enabled.  We recommend that you do not disable the recovery and replacement option.  If you set the `--no-recovery` option, you can never replace lost or damaged OCSs generated for that Security World. Therefore, you could never recover any keys protected by lost or damaged OCSs, even if the keys themselves were generated as recoverable (which is the default for key generation).  OCS and softcard replacement cannot be enabled after Security World creation without reinitializing the Security World and discarding all the existing keys within it.
`--cipher-suite=<CIPHER-SUITE>`	This option specifies the Cipher suite and type of key that is used to protect the new Security World. `<CIPHER-SUITE>` should be set to `DLf3072s256mAESc-SP800131Ar1`.
`--nso-timeout=<TIMEOUT>`	This option allows you to specify the time-out (`<TIMEOUT>`) for new Security Worlds. By default, an integer given for *TIMEOUT* is interpreted in seconds, but you can supply values for *TIMEOUT* in the form *N* `s`, *N* `h`, or *N* `d` where *N* is an integer and `s` specifies second, `h` specifies hours, and `d` specifies days.
`--module=<MODULE>`	This option specifies the module to use (by its `ModuleID`). If you have multiple modules, `new-world` initializes them all together.

Option	Description
`--acs-quorum=<K>/<N>`	In this option, `<K>` specifies the minimum number of smart cards needed from the ACS to authorize a feature. You can specify lower *K* values for a particular feature. All the *K* values must be less than or equal to the total number of cards in the set. If a value for *K* is not specified, `new-world` creates an ACS that requires a single card for authorization. When the Security World is created in `common-criteria-cmts` mode, `new-world` requires a minimum K of 2. Some applications do not have mechanisms for requesting that cards be inserted. Therefore any OCSs that you create for use with these applications must have *K*=1. `<N>` specifies the total number of smart cards to be used in the ACS. This must be a value in the range 1 – 64. If a value for this option is not specified, `new-world` creates an ACS that contains a single card. We recommend that you do not create an ACS for which the required number of cards is equal to the total number of cards because you will not be able to replace the ACS if even a single card is lost or damaged. This option only takes effect if you are creating a new Security World.
`--reduced-features`	This option instructs `new-world` to use a reduced default feature set when creating a Security World. A Security World created with the `--reduced-features` option has no passphrase recovery; no NVRAM, RTC, or FTO; and no NSO dele gate keys. However, such a reduced-features Security World can perform many operations faster than more fully featured Security Worlds.
`--disablepkcs1pad`	This option disables the use of PKCS#1 v1.5 padding. All attempts to use PKCS#1 v1.5 padding for encryption or decryption operations will be rejected. PKCS#1 v1.5 signature operations are not affected. PSS and OAEP are not affected.

Option	Description
`--pp-min=LENGTH`	This option enables a minimum passphrase length check for the Administrator Card Set (ACS) the Operator Card Set (OCS) and any associated softcards when you create a Security World. The minimum passphrase length check is then applied after the Security World is created. When enabled and you attempt to create a card passphrase with fewer characters than the specified minimum length, the following warning message displays:  Warning: short passphrase. However, the passphrase can still be used. Example: `new-world --initialize --acs-quorum=K/N  --pp-min=14` If `--pp-min=<length>` is not used, the minimum passphrase length is set to the default value (0).
`--pp-strength`	This option enables passphrases to have at least one uppercase, lowercase, number, and symbol. If the `--pp-strength` argument is omitted, the complexity requirements are not enforced.
`--audit-logging`	This option configures the Security World and the HSM on which it is being created for audit logging, creating a log signing key for each HSM.  The log destination must have already been set in the hardserver configuration file. See Audit Logging. Audit logging is automatically enabled when the Security World is created in `common-criteria-cmts` mode.
`--max-keyusage`	This option allows the administrator to specify a maximum reauthorization condition in terms of number of key usages since authorization for Assigned keys in common-criteria-cmts mode. A use limit compatible with the specified maximum will be applied at key creation time and can be verified for Assigned keys. If this is not set then no `--max-keyusage` limit is applied to Assigned keys on creation.
`--max-keytimeout`	This option allows the administrator to specify a maximum reauthorization condition in terms of a TIMEOUT since authorization for Assigned keys in common-criteria-cmts mode. By default, an integer given for TIMEOUT is interpreted in seconds, but you can supply values for TIMEOUT in the form *Ns*, *Nh*, or *Nd* where *N* is an integer and *s* specifies second, *h* specifies hours, and *d* specifies days. A use limit compatible with the specified maximum will be applied at key creation time and can be verified for Assigned keys. If this is not set then no limit is applied to Assigned keys on creation.

The `--max-keyusage` and `--max-keytimeout` options are only available in common-criteria-cmts mode. They provide support for the Protection Profile requirement that reauthorization conditions are set by an adminis trator on creating an Assigned Key.

11.1.4.3. new-world command-line utility features

Features for the Security World can be specified using the command line.

Security world features are selected by *feature expressions*. A feature expression is a comma-separated list of *feature terms*. Each term consists of a feature name, optionally preceded by either a double dash `--`, an exclamation point `!`, or `no-` to turn off the feature, and optionally followed by an equals sign `=` and the quorum of cards from the ACS required to use the feature. The default quorum is taken from the `K` argument of the `--acs-quorum` option.

The `!` character is interpreted by some shells as history expansion and must be escaped with a backslash, `\!`. The dash may be interpreted as being the start of an command-line option unless you have used the `-f` option or specified an HSM without including the `-m` flag.

If you set the `!fto` flag, that is, turn off FTO, you will not be able to use smart cards to import keys.

To use extended debugging for the HSM, you must set the `dseeall` flag.

The following feature names are available:

Feature name	Description
`m`	This feature makes it possible to add new HSMs into the Security World. This feature cannot be disabled.
`r`	This feature enables OCS and softcard replacement; see Replacing Operator Card Sets.
`p`	This feature enables passphrase replacement; see passphrase replacement and Changing card and softcard passphrase.
`nv`	This feature specifies that ACS authorization is needed to enable nonvolatile memory (NVRAM) allocation.
`rtc`	This feature specifies that ACS authorization is needed to set the real-time clock (RTC), see Real-time clock (RTC) options.

Feature name	Description
dsee	This feature specifies that that ACS authorization is needed to enable SEE World debugging.
dseeall	This feature enables SEE World debugging for all users.
fto	This feature specifies that ACS authorization is needed to enable foreign token operations (FTO).

The following features remain available for use on presentation of the standard ACS quorum, even if turned off using the ! operator:

- nvram
- rtc
- fto

Setting the quorum of one these features to 0 has the same effect as turning it off using the ! operator.

The passphrase replacement (p) and dseeall features are turned off by default; the other options are turned on by default.

The nonvolatile memory and SEE world debugging options are relevant only if you are using the Secure Execution Engine. If you have bought the CodeSafe Developer Kit, refer to the *CodeSafe Developer Guide* for more information.

To use extended debugging for the HSM, you must set the dseeall flag.

The dseeall option is designed for testing purposes only. Do not enable this feature on production Security Worlds as it may enable SEE applications to leak security information.

For example, the following features:

```
m=1, r, !p, nv=2, rtc=1
```

Create a Security World for which:

- A single card from the ACS is required to add a new HSM
- The default number is required to replace an OCS
- passphrase replacement is not enabled
- Two cards are required to allocate nonvolatile memory

- One card is required to set the real-time clock (applies to SEE only).

11.1.4.4. new-world command-line utility output

If `new-world` cannot interpret the command line, it displays its usage message and exits.

If you attempt to set a quorum for a feature that you have disabled or if you attempt to set a quorum too high, `new-world` displays an error and exits.

If the HSM is not in the pre-initialization mode, `new-world` advises you that you must put the HSM in this mode and waits until you have changed the HSM mode before continuing.

The HSM must be in pre-initialization mode. See Checking and changing the mode on an nShield 5s module for more about changing the mode.

If the HSM is in the pre-initialization mode, `new-world` prompts you for smart cards and passphrases as required.

11.1.5. After you have created a Security World

Store the ACS in a safe place.

If you lose more than *N* minus *K* of these Administrator Cards you cannot restore the Security World or lost Operator Cards. For example, if you have a 2/3 ACS and you lose more than one card, you cannot restore the Security World. If you have created an Administrator card set where *K* = *N*, then the loss of one card stops you from being able to restore the Security World.

To prevent this situation from occurring, replace lost or damaged cards from the ACS as soon as you discover the loss or damage. For more information, see Replacing the Administrator Card Set.

The security of the keys that you create within this Security World is wholly dependent on the security of these smart cards.

The Security World host data is stored in the directory to which the `NFAST_KMLOCAL` environment variable points (see Security World Files). The data in this directory is encrypted. You should:

- Ensure that this directory is backed up regularly.
- Check the file permissions for this directory.

- Ensure that the nFast Administrator role, and any user that you want to be able to create Operator Cards or keys, have write permission for this directory.
- All other valid users must have read permission.

Installation of Security World Software must be performed by a user with Administrator rights that allow read and write operations, and applications to be started and stopped.

The HSM can now be used to create Operator Cards and keys for the new Security World.

11.2. Displaying information about your Security World

To display information about the status of your Security World:

- Run the `nfkminfo` command-line utility. See Displaying information about a Security World with nfkminfo.
- Run the `kmfile-dump` command-line utility. See Displaying information about a Security World with kmfile-dump.

You can also use KeySafe to view a summarized description of the Security World.

11.2.1. Displaying information about a Security World with nfkminfo

To display information about a Security World from the command line, run the command:

```
nfkminfo -w|--world-info [-r|--repeat] [-p|--preload-client-id]
```

In this command, the `-w|--world-info` option specifies that you want to display general information about the Security World. This option is set by default, so you do not need to include it explicitly.

Optionally, the command can also include the following:

Option	Description
`-r|--repeat`	This option repeats the information displayed. There is a pause at the end of each set of information. The information is displayed again when you press **Enter**.
`-p|--preload-client-id`	This option displays the preloaded client ID value, if any.

To output a detailed list of Security World information, run `nfkminfo` with the `-w|--world-info` option (with or without the other options). For a description of the fields in this list,

and more information about using `nfkminfo`, see *nfkminfo: information utility*.

The following table maps there flags visible on the front panel when you select **3 Security World mgmt > 3-1 Display World Info** to the flags in the output of `nfkminfo`.

Front panel	nfkminfo
admin	k-out-of-n
nCore flags	slotlistflags
NFKM flags	flags
Module slots	nflags
Initialized	Initialised
ForeignTokenOpen	FTO

11.2.2. Displaying information about a Security World with kmfile-dump

To display information about a World from the command line, run the command:

```
kmfile-dump [<worldfile>]
```

where `<worldfile>` is the file storing the World data, usually `/opt/nfast/kmdata/local/world`

If no **WorldVersion** is received as a result of the command then the World is either version 1 or version 2.

If a **WorldVersion** of either '2' or '3' is received then the World is version 3.

11.3. Adding or restoring an HSM to the Security World

When you have created a Security World, you can add additional HSMs to it. These additional HSMs can be on the same host computer as the original HSM or on any other host. The HSMs may have previously been removed from the same Security World, that is, the Security World can be restored on an HSM by adding the HSM to the Security World again.

You can also restore an HSM to a Security World to continue using existing keys and Operator Cards:

- After you upgrade the firmware
- If you replace the HSM.

The additional HSMs can be any nShield HSMs.

To add an HSM to a Security World, you must:

- Have installed the additional HSM hardware, as described in the *Installation Guide*.
- After installing additional HSM hardware and restarting host machine, you must stop and then restart the hardserver as described in Stopping and restarting the client hardserver. This ensures that the added HSM is recognized and accessible.
- Have a copy of the Security World data on this host. This is the host data written by Keysafe or `new-world` when you created the Security World. This data is stored in the `local` directory within the Key Management Data directory.

  If the Key Management Data directory is not in the default location, ensure that the `NFAST_KMDATA` environment variable is set with the correct location for your installation.

- Be logged in to the host computer as root; see Hardserver start-up settings and server_startup.
- Have started the HSM in pre-initialization mode.
- The HSM must be in pre-initialization mode. See Checking and changing the mode on an nShield 5s module for more about changing the mode.
- Possess a sufficient number of cards from the ACS and the appropriate passphrases.

Adding or restoring an HSM to a Security World:

- Erases the HSM
- Reads the required number of cards (*K*) from the ACS so that it can re-create the secret
- Reads the Security World data from the computer's hard disk
- Uses the secret from the ACS to decrypt the Security World key
- Stores the Security World key in the HSM's nonvolatile memory
- Configures the HSM for audit logging if the Security World was created with audit logging selected.

After adding an HSM to a Security World:

- You cannot access any keys that were protected by a previous Security World that contained that HSM.
- You have to sync the module file to the clients by one of the following methods:
  - Copy the files manually to the clients.

- Run `rfs-sync -update`.

See Hardserver configuration files.

It is not possible to program an HSM into two separate Security Worlds simultaneously.

Initialization removes any data stored in an HSM's nonvolatile memory (for example, data for an SEE program or NVRAM-stored keys). To preserve this data, you must back it up before initializing the HSM and restore it after the HSM has been reprogrammed. We provide the `nvram-backup` utility to enable data stored in nonvolatile memory to be backed up and restored.

In order to continue using existing keys and Operator Cards, you must reprogram the HSM:

- After you upgrade the firmware
- If you replace the HSM
- If you need to add an HSM to an existing Security World.

11.3.1. Adding an HSM to a Security World with new-world

1. Open a command window and type the command:

```
new-world [-l|--program] [-S|--no-remoteshare-cert] [-m|--module=<MODULE>]
```

In this command:

- `-l|--program`

  This option adds an HSM to an existing Security World (in the Key Management Data directory). If you have multiple HSMs available, you can use the `-m|--module=`MODULE option to specify an HSM. If you do not specify an HSM `new-world` adds all available HSMs to the Security World.

- `-S|--no-remoteshare-cert`

  These options prevent the HSM from becoming a target for remote shares.

- `-m|--module=<MODULE>`

  This option specifies the HSM to use (by its `ModuleID`). If you have multiple HSMs and do not specify an HSM, `new-world` adds all available HSMs to the existing Security World.

If `new-world` cannot find the key-management data, it displays the message:

```
new-world: no existing world to load.
```

If you intend to initialize the HSM into a new Security World, run `new-world` with the `-i` option.

If the HSM is not in the pre-initialization state, `new-world` displays an error and exits.

The HSM must be in pre-initialization mode. See Checking and changing the mode on an nShield 5s module for more about changing the mode.

If the HSM is in the pre-initialization state, `new-world` prompts you for cards from the Security World's ACS and to enter their passphrases as required.

2. After `new-world` has reprogrammed the HSM, restart the HSM in the operational state.

   The HSM must be in pre-initialization mode. See Checking and changing the mode on an nShield 5s module for more about changing the mode.

3. Store the ACS in a safe place.

If any error occurs (for example, if you do not enter the correct passphrases), the HSM is reset to the factory state. The HSM does not form part of the Security World unless you run `new-world` again.

11.4. Security World migration

The current version of Security World software enables you to create a Security World that fully complies with the NIST Recommendations for the Transitioning of Cryptographic Algorithms and Key Sizes (SP800-131Ar1) or alternatively Common Criteria PP 419 221-5 (common-criteria-cmts) depending on the options selected at World creation. This is called World version 3.

We recommend that where compliance with the specifications above is required, you create a new World and create new keys within that World. However, the software also includes a `migrate-world` command-line utility that you can use for migrating existing keys into the new World. This is provided as a convenience for customers who require compliance with the specifications, and who need to continue using existing keys.

In the case of a Common Criteria World the specification prohibits the importing of assigned keys. Only general keys can be imported into a common-criteria-cmts World.

Throughout the following sections, the terms `Source World` refers to the World from which you want to migrate keys, and `Destination World`

refers to the World to which you want to migrate keys.

The utility requires the use of two modules. One module is referred to as the source module. The other module is referred to as the destination module.

11.4.1. Pre-requisites for migrating keys

In order to use the `migrate-world` utility the following will be needed:

- Two HSMs. These can be any of the currently supported HSM types and the two HSMs do not need to be of the same type.
- A quorum of ACS cards for the source World.
- A quorum of ACS cards for the destination World.
- Sufficient blank cards to create new OCS cards for any keys that are OCS protected.
- Remote mode switching must be enabled on both HSMs used for the migration.

11.4.2. Restrictions on migrating keys

The following restrictions apply to the use of `migrate-world`:

- The source module must be running firmware version 12.50 or later.
- The destination module must be running firmware version 12.50 or later.
- Only recoverable keys can be migrated. If your source keys are non-recoverable, you cannot use the migration utility to migrate keys.
- **Security world software version 13.2 and earlier**: The key protection or quorum cannot be changed during migration.
- **Security world software version 13.3 and later**: The key protection or quorum can be changed during migration. The new protections must be created in the destination world before migration begins.
- Replacement cards should be of the same or newer generation than the cards that they replace.
- The source and destination modules must both have KLF2 warrants.

  nShield Connect and nShield 5c HSMs have a pre-installed KLF2 warrant file in:

  Linux `NFAST_KMDATA/hsm-<esn>/warrants/<esn>`.

  Windows: `NFAST_KMDATA\hsm-<esn>\warrants\<esn>`

If one or both of the modules are Solo XC and have a KLF warrant you should request an upgrade to a KLF2 warrant before starting migration, see Warrant Management for nShield 5s.

- The operator running the migrate-world utility must have the access rights to create a privileged connection to the hardserver.
- The migration tool must have exclusive use of the modules during migration. Do not use them for any other purpose during migration and if either module is an nShield network-attached HSM, do not enter anything via the front panel during migration.

If the destination World is `fips-140-level-3`, then some keys that were usable in the source World may not be usable in the destination World due to those algorithms or key lengths being restricted. The migration tool might not be able to successfully migrate these keys so they should be removed from the source World before attempting the migration. Any keys of this type that do migrate successfully will be restricted at the point of use.

If the destination World is `fips-140-level-3` or `common-criteria-cmts` the migration tool will automatically remove ExportAsPlain from the ACL of any migrated key during the migration process.

If the destination world does not support audit logging the migration tool will automatically remove LogKeyUsage from the ACL of any migrated key during the migration process.

11.4.3. About the migration utility

You can run the migration utility in the following modes:

- **Plan mode**: Returns a list of steps for migration and the required card sets and passphrases but does not migrate any keys.
- **Perform mode**: Runs the plan mode prior to presenting the option to proceed and migrate keys according to the plan.

11.4.3.1. Usage and options

```
migrate-world [OPTIONS] --src-module=<source_module> --dst-module=<dest_module> --source=<source-kmdata-path>
--debug --dst-warrant=<dst-warrant-path> --src-warrant=<src-warrant-path [--plan | --perform] --key-logging
```

Option	Enables you to...
-k <KEYS>|--keys-at-once=<KEYS>	Migrate no more than this number of keys per ACS loading. This is useful to prevent ACS time-outs if you have a large number of keys to migrate. (0=unlimited, default=0). It is recommended to limit the number of keys to be migrated at any one time to no more than 100.
-h|--help	Obtain information about the options you can use with the utility.
-c <CARDSETS>|--cardsets-at-once=<CARDSETS>	Migrate keys protected by this number of card sets or softcards per ACS loading. This is useful to prevent ACS time-outs if you have a large number of different card sets or softcards to migrate. (0=unlimited, default=0).
--version	View the version number of the utility.
--src-warrant=<src-warrantfile>	Specify the location of the warrant file of the source module.
--src-module=<MODULE>	Specify which module ID to use as the source module.
--source=<SOURCE>	Specify the path to the folder that contains the source World data.
--plan	View the list of steps that will be carried out.
--perform	Migrate keys interactively.
--dst-warrant=<dst-warrantfile>	Specify the location of the warrant file of the destination module.
--dst-module=<ModuleId>	Specify which module ID to use as the destination module.
--debug	Outputs debug messages and stack traces in case of errors. It is recommended to use this only for testing as it will slow down operation and make card timeouts more likely to occur. A large volume of output is produced for each key that is migrated, so it is recommended to migrate a single key at a time when using this option.
--key-logging	This option will enable key usage logging on all migrated keys. If the destination World does not support audit logging the keys will still be migrated but LogKeyUsage logging will not be set in the ACL of the migrated keys.
--src-prots=<list of source protections>	Specify a comma-separated list of OCS or softcard names in the source security world. The keys will be migrated to the corresponding protections specified with --dst-prots.
--dst-prots=<list of destination protections>	Specify a comma-separated list of OCS or softcard names in the destination security world. These will be the target protections for the keys that are protected with methods specified with --src-prots in the source security world.
--prots-config=<PATH>	Specify a configuration file that lists the source and destination protection pairs for migration. The file must contain pairs of tab-separated protection names src_prot dst_prot, one pair per line.

Do not terminate path names in the command parameters with a backslash character. If this is not possible then either terminate with a double backslash or insert a blank space between the backslash and the terminating quotation mark.

11.4.4. Migrating keys

11.4.4.1. Preparing for migration

Before you begin:

- Install the latest version of the Security World Software from the installation media. See the *Installation Guide* for more information.
- Ensure that the warrant files for the source and destination modules are stored in their default locations. If the warrant files are not at the default location, the `--src-warrant` and `--dst-warrant` parameters need to be specified in the `migrate-world` command.
  - For Solo +, or Solo XC, the default location is `NFAST_KMDATA/warrants/`.
  - For Connect +, Connect XC modules, the default location is `NFAST_KMDATA/hsm-<ESN>/warrants/`.
  - For nShield 5s and nShield 5c, you do not need to specify warrant locations because they store their warrants within the module.
- Copy the source World data to a location defined by the `--source=<SOURCE>` parameter of the migration tool.
- If the destination World does not exist already, create a new destination World. For instructions, see Creating a Security World

You must enable all your features on the destination module before migration. Otherwise, the migration will fail.

11.4.5. Migrating keys process

To ensure the security of your keys, we recommend that the migration process is overseen by ACS-holding personnel and the end-to-end migration process is completed continuously, without any breaks in the process. This will also reduce the possibility of your ACS experiencing a time-out.

If the destination World supports audit logging you can choose whether the migrated keys will have key usage logging enabled or not by use of

> the `--key-logging` command line switch. If you only wish key usage logging to be enabled on a subset of the keys then you must separate the source keys into two groups and run the migrate-world command separately for each group.

To migrate keys to the destination World:

1. Run the migration utility in the perform mode with the required options. For information about the usage and options you can use, see About the migration utility.
2. Ensure that the data for the destination World is in the standard location for World data, derived from one of the following:
   - Either the environment variable `NFAST_KMLOCAL` or `NFAST_KMDATA`.
   - The default directory: `/opt/nfast/kmdata/local/`.
3. If the module is not configured to use the destination World, the utility prompts you to program the module and supply the ACS of the destination World.
4. The utility guides you through specific steps, prompting you to supply the required card sets and passphrases.
5. At the end of the migration both the source and destination modules are cleared. If you wish to use the modules then you must reload them with an appropriate Security World.

> The utility will attempt to automatically change the module mode when needed. Should the automatic change of mode fail for any reason, then the utility will prompt you to change the module state to either initialization or operational at various points during the pro cedure. * The HSM must be in pre-initialization mode. See Checking and changing the mode on an nShield 5s module for more about changing the mode.

11.4.6. Verifying the integrity of the migrated keys

To verify the integrity of the migrated keys, run the `nfkmverify` utility with the following options, as appropriate:

- If the keys are module-protected, run the utility with one of the following options:
  - `-L` option, which checks the ACL of a loaded key instead of the generation certificate.
  - `-R` option, which checks the ACL of a key loaded from a recovery blob.
- If the keys are protected by cardsets or softcards, run the `nfkmverify` utility with the `-R` option in combination with the preload utility.

Example:

```
preload --admin=RE nfkmverify -R -m1 <application> <key-ident>
```

Do not use the `nfkmverify` utility with the default `-C` option. If you use this option, the utility returns errors because the ACL in the cer tificate will reflect the old world.

Note that if the destination World is `fips-140-level-3` then some keys that were usable in the source World may not be usable in the destination World due to those algorithms or key lengths being restricted. The migration tool will successfully migrate the keys but they will be restricted at the point of use.

11.4.7. Migrating keys using custom protection pairs

Regular security world migration will create new card sets and softcards in the destination world with the same names as the source protections or it will use existing destination pro tections if they share a name and type (card set or softcard) with the source protection.

You can specify custom protection pairs if you want to change the name, the quorum, or the properties of the protection. You can also combine multiple source protections of the same type into one destination protection. You cannot diffuse keys from one source protec tion to multiple destination protections.

The source-destination protection pairs can be selected either as:

- Two comma-separated lists `--src-prots <source protections>` and `--dst-prots <des tination protections>`.
- Tab-separated pairs "source destination", one per line, in a configuration file `--prots -config <file path>`.

The protections can be referred to by their name, 40-character hash, or "c:name" and "s:name" when a source card set and softcard share a name. The source and destination pro tection types must match.

The following example shows the two ways of specifying a set of protection pairs and the different ways each protection can be referred to. The example hashes are shortened for readability.

Protection type	Source protection to be migrated	Target destination protection
card set	`ocs 1`	`ocstarget1`
softcard	`softcard 1`	`softcardtarget`
card set	`name1` (duplicate name)	`ocstarget1`
softcard	`name1` (duplicate name)	`softcardtarget`
card set	`name2` (duplicate name and type) `hash: XXXXXXX1`	`ocstarget1`
card set	`name2` (duplicate name and type) `hash: XXXXXXX2`	`ocstarget2`

By specifying the lists using the `--src-prots` and `--dst-prots` options:

```
migrate-world [OPTIONS] \
--src-prots "ocs 1,softcard 1,c:name1,s:name1,XXXXXXX1,XXXXXXX2" \
--dst-prots "ocstarget1,softcardtarget,ocstarget1,softcardtarget,ocstarget1,ocstarget2"
```

By using a configuration file specified with the `--prots-config` option:

```
migrate-world [OPTIONS] --prots-config=migration.cfg

--- migration.cfg ---
ocs 1    ocstarget1
softcard 1    softcardtarget
c:name1    ocstarget1
s:name1    softcardtarget
XXXXXXX1    ocstarget1
XXXXXXX2    ocstarget2
--------------------
```

11.4.8. Troubleshooting

If you encounter any errors that are not listed in the following table, con tact Support.

Error	Explanation	Action
There are no keys requiring migration.	Any migrate-able keys found in the source World already exist in the destination World. The migration utility returns this error if: • The keys have already been migrated • All keys are non-recoverable and therefore cannot be migrated.	None.
Source module must be specified. Destination module must be specified. Source and Destination modules must be different. Module is not usable.	This utility requires you to specify both a source and destination module which must be different modules and both must be usable.	Specify the correct modules.
Source World has indistinguishable cardsets or softcards. Destination World has indistinguishable keys.	There are irregularities in one of the Worlds, but these irregularities do not affect the migration process.	None.
Destination World has indistinguishable cardsets or softcards. Source World has indistinguishable keys. Cannot determine protection of keys.	There are problems with one of the Worlds.	Contact Support.
Source World not recoverable.	The source World is not recoverable and the keys therefore cannot be migrated.	If the source World is not recoverable, you cannot use the migration utility to migrate keys. Contact Support.
Missing security World at PATH. Source world must be specified.	The path for the source World is wrong. There is no World data at the location that was specified when running the migration utility.	Supply the correct path to the source World. If you have supplied the correct path to the directory that contains the source World data, the migration utility has not found a destination World.

Error	Explanation	Action
Source World is the same as the destination World.	An incorrect path was supplied for the source World data when running the utility. The destination World data does not exist in the default location defined by the environment variable `NFAST_ KMLOCAL` or `NFAST_KMDATA`.	Run the utility with the correct path to the source World data. Move the source World data to a different location and then copy the destination World data to the default location. If the default location is defined by an environment variable, configure the variable to point to the location of the destination World, which then becomes the new default location.
Cannot find <NAME> utility, needed by this utility. <NAME> utility is too old, need at least version <VERSION NUMBER>.	The software installation is partially completed. The path (in the environment variable for the operating system) might be pointing to an old version of the software.	Reinstall the software. Ensure that the path points to the latest version of the software.
nFast error: TimeLimitExceeded; in response to SetKM...	The ACS time-out limit has expired.	Restart the key migration process; see Security World migration.
Destination world does not support audit logging.	You have specified the `--key-logging` option but the destination world does not support audit logging.	None. The keys will be migrated but LogKeyUsage will not be set in the ACL of migrated keys.
Failed to load warrant file <FILE>.	There is a problem reading the warrant file.	Check that your warrant files are in the correct location and have not been edited in any way.

11.5. Erasing a module from a Security World

Erasing a module from a Security World deletes from the module all of the secret information that is used to protect your Security World. This returns the module to the factory state. Provided that you still have the ACS and the host data, you can restore the secrets by adding the module to the Security World.

Erasing a module removes any data stored in its nonvolatile memory (for example, data for an SEE program or NVRAM-stored keys). To preserve this data, you must back it up before erasing the module. We provide the `nvram-backup` utility to enable data stored in nonvolatile memory to be backed up and restored.

In order to erase a module, you must:

- Be logged into your computer as `root`.
- Have started the module in the pre-initialization mode.
- The HSM must be in pre-initialization mode. See Checking and changing the mode on an nShield 5s module for more about changing the mode.

You do not need the ACS to erase a module. However, unless you have a valid ACS and the host data for this Security World, you cannot restore the Security World after you have erased it.

After you have erased a module, it is in the same state as when it left Entrust (that is, it has a random module key and a known K_{NSO}).

If you are physically removing the module from the machine where it is installed, run `hsmadmin enroll` after removing the module. This refreshes the list of nShield 5s modules currently installed on the machine.

11.5.1. Erasing a module with new-world

The `new-world` command-line utility can erase any modules that are in the pre-initialization mode.

To erase modules with the `new-world` utility, run the command:

```
new-world [-e|--factory] [-m|--module=<MODULE>]
```

In the `new-world` command:

Option	Description
`-e|--factory`	These options restore a module to its factory state.
`-m|--module=<MODULE>`	These options specify the `ModuleID` to use. `new-world` erases only one module at a time. To erase multiple modules, you must run `new-world` once for every module that you want to erase.

11.5.1.1. Output

If `new-world` successfully erased a module, it displays a message that it restored the module to factory state. Otherwise, `new-world` returns an error message.

11.5.2. Erasing a module with KeySafe

You can erase a module on a server with KeySafe by following these steps:

1. Start KeySafe. (For an introduction to KeySafe and information on starting the software, see Using KeySafe.)
2. Click the **World** menu button, or select **World** from the **Manage** menu. KeySafe takes you to the **World Operations** panel.
3. Click the **Erase Module** button. KeySafe takes you to the **Erase Module** panel.
4. Select the module that you want to erase by clicking its listing on the **Security world status** tree, then click the **Commit** command button.

   KeySafe erases all secrets from the module, returning it to its factory state.

If you have any keys that were protected by an erased module, you cannot access them unless you restore these secrets. You cannot restore these secrets unless you have the appropriate ACS.

11.5.3. Erasing a module with initunit

The `initunit` command-line utility erases any modules that are in the pre-initialization state.

To erase modules with the `initunit` utility, run the command:

```
initunit [-m|--module=<MODULE>] [-s|--strong-kml]
```

In the `initunit` command, `--module=<MODULE>` specifies the ID of the module you want to erase. If you do not specify this option, all modules in the pre-initialization state are erased. `--strong-kml` specifies that the module generates an AES (SP800-131A) module signing key, rather than the default key.

The `--disablepkcs1pad` option will only work on SP800-131A Security Worlds.

11.5.3.1. Output

If `initunit` is successful, for each module that is in the pre-initialization state, it returns a message similar to this:

```
 Initialising Unit
>#  Setting dummy HKNSO Module Key Info:
 HKNSO
>################### HKM
>###################
```

Otherwise, `initunit` returns an error message.

11.6. Deleting a Security World

You can remove an existing Security World and replace it with a new one if, for example, you believe that your existing Security World has been compromised. However:

- You are not able to access any keys that you previously used in a deleted Security World
- It is recommended that you reformat any nShield Remote Administration Cards that were used as Operator Cards within this Security World *before* you delete it. For more information about reformatting (or erasing) Operator Cards, see Erasing cards and soft cards.

Except for nShield Remote Administration Cards, if you do not reformat the smart cards used as Operator Cards before you delete your Security World, you must throw them away because they cannot be used, erased, or reformatted without the old Security World key.

You can, and should, reuse the smart cards from a deleted Security World's ACS. If you do not reuse or destroy these cards, then an attacker with these smart cards, a copy of your data (for example, a weekly backup) and access to any nShield key management HSM can access your old keys.

To delete an existing Security World:

1. Remove all the HSMs from the Security World.
2. Delete the Security World data files, see Location of Security World files.

There may be copies of the Security World data archive saved on your backup media. If you have not reused or destroyed the old ACS, an attacker in possession of these cards could access your old keys using this backup media.

If audit logging was enabled for the Security World then audit logs can still be verified provided that the audit log data is maintained as this contains all the information needed to verify the logs. For further information see *Audit Logging*.

12. Managing card sets and softcards

This chapter describes how to create and manage card sets and softcards, using a Security World.

When you create a Security World, an Administrator Card Set (ACS) is created at the same time. You use the ACS to:

- Control access to Security World configuration
- Authorize recovery and replacement operations.

The Security World is used to create and manage keys, and the Operator Card Sets (OCSs) and softcards you create with the Security World are used to protect those keys.

A Security World offers three levels of key protection:

Level of protection	Description
Direct protection	Keys that are directly protected by the Security World are usable at any time without further authorization.
Softcard	Keys that are protected by a softcard can only be used by the operator who possesses the relevant passphrases.
OCS	Keys that are protected by an OCS can only be used by the operator who possesses the OCS and any relevant passphrases (if set).

For more information about creating a Security World, see Creating and managing a Security World.

For more information about key management, see Working with keys.

After a Security World has been created, you can use it to create and manage OCSs and softcards (as described in this chapter), as well as to create and manage the keys it protects (see Working with keys).

If you want to use the Remote Operator feature to configure smart cards for use with a remote module, see Remote Operator.

12.1. Creating Operator Card Sets (OCSs)

You can use an Operator Card Set (OCS) to control access to application keys. OCSs are optional, but if you require one, create it before you start to use the hardware security mod ule with applications. You must create an OCS before you create the keys that it is to protect.

You can create OCSs that have:

- Names for individual cards, as well as a name for the whole card set
- Specific *K*/*N* policies
- Optional passphrases for any card within a given set
- Formal FIPS 140 Level 3 compliance.

Some third-party applications impose restrictions on the OCS smart card quorums (*K*/*N*) or the use of smart card passphrases. For more information, see the appropriate integration guide for the application. Integration guides for third-party applications are available from https://nshieldsupport.entrust.com/.

OCSs belong to the Security World in which they are created. When you create an OCS, the smart cards in that set can only be read by hardware security modules belonging to the same Security World.

You can also use the following tools to create an OCS:

- The `createocs` command-line utility, as described in Creating an Operator Card Set using the command line
- KeySafe, as described in Creating an Operator Card Set with KeySafe

12.1.1. Persistent Operator Card Sets

If you create a standard (non-persistent) OCS, the keys it protects can only be used while the last required card of the quorum remains loaded in the local slot of the HSM, or one of its Dynamic Slots. The keys protected by this card are removed from the memory of the device as soon as the card is removed from the smart card reader. If you want to be able to use the keys after you have removed the last card, you must make that OCS persistent.

Keys protected by a persistent card set can be used for as long as the application that loaded the OCS remains connected to the hardware security module (unless that application removes the keys).

For more information about persistent OCSs, see Using persistent Operator Card Sets.

12.1.2. Time-outs

OCSs can be created with a time-out, so that they can only be used for limited time after the OCS is loaded. An OCS is loaded by most applications at start up or when the user sup-

plies the final required passphrase. After an OCS has timed out, it is not loadable by another application unless it is removed and reinserted. Time-outs operate independently of OCS persistence.

12.1.3. FIPS 140 Level 3-compliant Security Worlds

When you attempt to create an OCS for a Security World that complies with FIPS 140 Level 3, you are prompted to insert an Administrator Card or Operator Card from an existing set. You may need to specify to the application the slot you are going to use to insert the card. You need to insert the card only once in a session.

12.1.4. Creating an Operator Card Set using the command line

To create an OCS from the command line:

1. Run the command:

```
createocs -m <MODULE>|--module=<MODULE> -Q|--ocs-quorum=<K>/<N> +
-N|--name=<NAME> [-M|--name-cards] +
[[-p|--persist]|[-P|--no-persist]] [[-R|--no-pp-recovery]|--pp-recovery] +
[-q|--remotely-readable] [-T|--timeout=<TIME>] [-e|--erase]
```

   This command uses the following options:

Option	Description
-m <MODULE>|--module=<MODULE>	This option specifies the number of the hardware security module to be used to create the token. If you only have one hardware security module, <MODULE> is 1.
-Q|--ocs-quorum=<K>/<N>	In this option, <K> is the minimum required number of cards. If you do not specify the value <K>, the default is 1. Some applications do not have mechanisms for requesting that cards be inserted. Therefore any OCSs that you create for use with these applications must have <K>=1. <N> is the total number of cards. If you do not specify the value <N>, the default is 1.
-N|--name=<NAME>	This option specifies a name for the card set. The card set must be named with this option before individual cards can be named using the -M/--name-cards=<NAME> options.

Option	Description
-M|--name-cards	Specifying this option allows you to name individual cards within the card set. You can only use this option after the card set has been named by using the --name=`NAME option. `createocs prompts for the names of the cards as they are created. Not all applications can display individual card names.
-p|--persist	This option creates a persistent card set.
-P|--no-persist	This option creates a non-persistent card set.
-R|--no-pp-recovery	This option specifies that passphrase replacement for this OCS is disabled. Setting this option overrides the default setting, which is that the card passphrases are replaceable. You can specify the enablement of passphrase replacement explicitly by setting the --pp-recovery option.
-q|--remotely-readable	This option allows this card set to be read remotely. For information on configuring Remote OCSs, see Remote Operator. Not required for Remote Administration.
-T|--timeout=<TIME>	This option sets the time-out for the card set. Use the suffix s to specify seconds, m for minutes, h for hours, and d for days. If the time-out is set to 0, the OCS never times out. Otherwise, the hardware security module automatically unloads the OCS when the amount of time specified by TIME has passed since the OCS was loaded.
-e|--erase	Specifying this option erases a card (instead of creating a card set). You can specify this option twice in the form -ee to repeatedly erase cards.

With Security World Software v11.72 and later, passphrases are limited to a maximum length of 254 characters, when using createocs. See Maximum passphrase length.

If you have created a FIPS 140 Level 3 compliant Security World, you must provide authorization to create new Operator Cards; createocs prompts you to insert a card that contains this authorization. Insert any card from the Administrator Card Set or any Operator Card from the current Security World.

When createocs has obtained the authorization from a valid card, or if no authorization is required, it prompts you to insert a card.

2. Insert the smart card to use.

   If you insert an Administrator Card from another Security World or an Operator Card that you have just created, createocs displays the following message:

```
Module x slot n: unknown card +
```

```
Module x slot n: Overwrite card ? (press Return)
```

where `x` is the hardware security module number and `n` is the slot number. If you insert an Operator Card from another Security World, `createocs` displays the following message:

```
Module x slot n: inappropriate Operator Card (TokenAuthFailed).
```

When you insert a valid card, `createocs` prompts you to type a passphrase.

The nShield PKCS #11 library requires Operator Cards with passphrases.

Some applications do not have mechanisms for entering passphrases. Do not give passphrases to Operator Cards that are to be used with these applications.

3. Type a passphrase and press **Enter**. Alternatively, press **Enter** if you do not want this card to have a passphrase.

   A passphrase can be of any length and can contain any character that you can type.

   If you entered a passphrase, `createocs` prompts you to confirm it.

4. Type the passphrase again and press **Enter**.

   If the passphrases do not match, `createocs` prompts you to input and confirm the passphrase again.

5. When the new card has been created, if you are creating a card set with more than one card in it, `createocs` prompts you to insert another card.
6. For each additional card in the OCS, follow the instructions from step 2 through 4.

12.1.5. Creating an Operator Card Set with KeySafe

KeySafe enables you to create OCSs with:

- Their own names
- *K*/*N* policies
- Optional passphrases for any card within the OCS
- Formal FIPS 140 Level 3 compliance.

To create an OCS with KeySafe:

1. Start KeySafe. (For an introduction to KeySafe and information on starting the software, see Using KeySafe.)
2. Click the **Card sets** menu button, or select **Card sets** from the menu.

   The **List Operator Card Sets** panel is displayed.
3. Select an HSM within the Security World from the Security World status pane.
4. Click the **Create new card set** button to open the **Create Operator Card Set** panel. You can specify the following options:
   a. A name for the card set.
   b. Whether passphrase recovery will be enabled for the OCS. (Only available if the Security World has passphrase recovery enabled.)
   c. Whether the card set can be used remotely. (Only available if the Security World has remote sharing available.) For more information, see Remote Operator.
   d. Whether this OCS will be persistent.
   e. Whether this OCS will have a time-out (a period after which the card set must be inserted again).
   f. The value for the time-out, in seconds.
   g. The total number of Operator Cards (*N*) that you want this OCS to have. This must be a value in the range 1 – 64.
   h. The number of Operator Cards needed to re-create a key (*K*). *K* must be less than or equal to *N*.
5. When you have entered all the details, click **Commit**. KeySafe takes you to a new **Create Operator Card Set** panel.

   If *K* is equal to *N*, a message is displayed:

   The total number of cards is equal to the required number of cards. – If the total and required number of cards are equal, losing one card will render any nonrecoverable keys unusable. Is this what you want?

   Click **Yes** to confirm the values for *K* and *N*, or **No** to change them.

   If you are creating the card set in a FIPS 140 Level 3 Security World, insert an Administrator Card or an existing Operator Card when prompted.
6. Insert a blank, unformatted card into the reader.

   A message is displayed, confirming that the card is blank. Click **OK** to open the **Set**

Card Protection passphrase panel.

If you insert a card from another OCS, KeySafe asks whether you want to erase it. If you insert an Administrator Card from the current Security World, KeySafe prevents you from accidentally erasing it. If you insert an OCS card from another Security World you will get the message:

Error. Unreadable card - may be incorrectly inserted or be from another Security World's operator card set. Please check.

To overcome this you must replace the card you have inserted with another card that is readable (or blank).

When creating a card set, KeySafe recognizes cards that already belong to the set before the card set is complete. If you accidentally insert a card to be written again after it has already been written, you receive a warning.

7. Select whether or not you want to set a passphrase for the currently inserted card. Each card in a set can have an individual passphrase, and you can also create a set in which some cards have passphrases and others do not.
8. If setting a passphrase for the currently inserted card, enter the same passphrase in both text fields. A passphrase can contain any characters you can type except for tabs or carriage returns (because these keys are used to move between data fields).

You can change a passphrase at any time. If you do not set a passphrase now, you can use the KeySafe `Change passphrase` option (on the **Examine/Change Card** panel) to add one later. Likewise, if you later decide that you do not need a passphrase on a card, you can use this option to remove it.

9. After entering your desired passphrase (if any) in both text fields, click the **OK** button. Unless you have entered details for the last card in the set, KeySafe returns you to the **Create Operator Card Set** panel and prompts you to enter the next card in the set to be written.
10. After KeySafe has written the details of the last smart card in the set, it displays a dialog indicating that the OCS has been successfully created. Click the **OK** button, and KeySafe returns you to the Create Operator Card Set panel, where you can create another OCS or choose a different operation by clicking one of the menu buttons.

12.2. Creating softcards

You must create a softcard before you create the keys that it is to protect.

A softcard is a file containing a logical token that cannot be loaded without a passphrase; its logical token must be loaded in order to authorize the loading of any key that is protected by the softcard. Softcard files are stored in the Key Management Data directory and have names of the form `softcard_<hash>` (where `<hash>` is the hash of the logical token share). Softcards belong to the Security World in which they are created.

A softcard's passphrase is set when you generate it, and you can use a single softcard to protect multiple keys. Softcards are persistent; after a softcard is loaded, it remains valid for loading the keys it protects until its `KeyID` is destroyed.

It is possible to generate multiple softcards with the same name or passphrase. For this reason, the hash of each softcard is made unique (unrelated to the hash of its passphrase).

Softcards are not supported for use with the nCipherKM JCA/JCE CSP in Security Worlds that are compliant with FIPS 140 Level 3.

To use softcards with PKCS #11, you must have `CKNFAST_LOADSHARING` set to a nonzero value. When using pre-loaded softcards or other objects, the PKCS #11 library automatically sets `CKNFAST_LOADSHARING=1` (load-sharing mode on) unless it has been explicitly set to `0` (load-sharing mode off).

As with OCSs, if debugging is enabled, a softcard's passphrase hash is available in the debug output (as a parameter to a `ReadShare` command).

You can create softcards from either:

- The command-line (see Creating a softcard with ppmk)
- KeySafe (see Creating softcards with KeySafe)

12.2.1. Creating a softcard with ppmk

To create a new softcard using the `ppmk` command-line utility:

1. Decide whether you want the new softcard's passphrase to be replaceable or non-replaceable. To create a softcard with a replaceable passphrase, run the command:

```
ppmk --new --recoverable <NAME>
```

To create a softcard with a non-replaceable passphrase, run the command:

```
ppmk --new --non-recoverable <NAME>
```

In these commands, <NAME> specifies the name of the new softcard to be created.

2. If you are working within a FIPS 140 Level 3 compliant Security World, you must provide authorization to create new softcards. The ppmk utility prompts you to insert a card that contains this authorization. Insert any card from the ACS. If you insert an Administrator Card from another Security World, ppmk displays an error message and prompts you to insert a card with valid authorization.

   When ppmk has obtained the authorization from a valid card, or if no authorization is required, it prompts you to type a passphrase.

3. When prompted, type a passphrase for the new softcard, and press **Enter**.

   A passphrase can be of any length and contain any characters that you can type except for tabs or carriage returns (because these keys are used to move between data fields).

4. When prompted, type the passphrase again to confirm it, and press **Enter**.

   If the passphrases do not match, ppmk prompts you to input and confirm the passphrase again.

After you have confirmed the passphrase, ppmk completes creation of the new softcard.

12.2.2. Creating softcards with KeySafe

To create a softcard with KeySafe:

1. Start KeySafe. (For an introduction to KeySafe and information on starting the software, see Using KeySafe.)
2. Click the **Softcards** menu button, or select **Softcards** from the **Manage** menu. KeySafe takes you to the **List Softcards** panel.
3. Click **Create New Softcard** to open the **Create Softcard** panel.
4. Choose parameters for the softcard:
   a. Enter a name for the softcard. You must provide a valid name for each softcard.
   b. Choose whether you want passphrase replacement to be enabled for the softcard.

In a Security World with passphrase recovery enabled the **Yes** radio button is selected as default and the selection can be changed between **Yes** and **No**. In a Security World with passphrase recovery disabled the **No** button is selected, and cannot be changed to **Yes**.

5. Click **Commit**.

If you are creating the softcard in a FIPS 140 Level 3 Security World, insert an Administrator Card or an existing Operator Card when prompted.

The **Set Softcard Protection passphrase** pane is displayed.

6. Set a passphrase for the softcard by entering the same passphrase in both text fields.

   A passphrase can contain any characters you can type except for tabs or carriage returns (because these keys are used to move between data fields) and can be up to 1024 characters long. You can change a passphrase at any time. You must provide a passphrase for each card.

7. After entering your desired passphrase in both text fields, click the **OK** button.

   KeySafe displays a dialog indicating that the softcard has been successfully created.

8. Click the **OK** button.

   KeySafe returns you to the **Create Softcard** panel, where you can create another softcard or choose a different operation by clicking one of the menu buttons.

12.3. Erasing cards and softcards

Erasing a card or softcard removes all the secret information from the card or softcard and deletes information about the card or softcard from the host.

In the case of an OCS that uses nShield Remote Administration Cards, it is possible to reformat the cards at any time using `slotinfo --ignore-auth`. In the case of an OCS that uses standard nShield cards, it is only possible to erase or format the cards within the Security World in which they were created.

You can erase Operator Cards using KeySafe or the `createocs` utility. You can also use these methods to erase Administrator Cards other than those in the current Security

World's ACS (for example, you could use these methods to erase the remaining Administrator Cards from an incomplete set that has been replaced or Administrator Cards from another Security World).

None of these tools erases cards from the current Security World's ACS.

If you erase an Operator Card that is the only card in an OCS, information about the card set is deleted. However, if you erase one card from an OCS of multiple cards, you must remove the card information from the `opt/nfast/kmdata/local/` directory after you have erased the last card.

You can erase an entire card set at one time with the KeySafe **Remove OCS!** feature. For more information, see List an Operator Card Set.

12.3.1. FIPS 140 Level 3-compliant Security Worlds

When you attempt to erase cards for a Security World that complies with FIPS 140 Level 3, you are prompted to insert an Administrator Card or Operator Card from an existing set. You may need to specify to the application the slot you are going to use to insert the card. You need to insert the card only once in a session. You can therefore use one of the cards that you are about to erase.

12.3.2. Erasing cards with KeySafe

To erase a card using KeySafe use the following procedure:

1. Start KeySafe. (For an introduction to KeySafe and information on starting the software, see Using KeySafe.)
2. Click the **Card Sets** menu button. KeySafe takes you to the Card Operations panel.
3. Click the **Examine/Change Card** navigation button. KeySafe takes you to the **Examine/Change Card** panel.
4. Insert the card that you want to erase into the reader.
5. Click the **Erase Card** button. You do not need to supply the passphrase (if there is one) to erase an Operator Card.
6. KeySafe asks you to confirm that you want to erase this card. If you are sure that you want to erase it, click the `Yes` button.

Erasing a card does not erase the keys protected by that card. The keys are still listed on the keys panel but are unusable.

If you erase an Operator Card that is the only card in an OCS, KeySafe deletes information about that card set. However, if you erase one card from an OCS of multiple cards, you must remove the card information from the `opt/nfast/kmdata/local` after you have erased the last card.

7. After erasing a card, KeySafe displays a dialog to confirm that the card has been erased. Click **OK** to continue using KeySafe.

You can erase an entire card set at one time with the KeySafe **Discard Card Set(s)** feature.

12.3.3. Erasing cards using the command line

To erase a card from the command line, run the command:

```
createocs -m|--module=<MODULE> -e|--erase
```

This command uses the following options:

Option	Description
`-m|--module=<MODULE>`	These options specify the module number of the module. If you only have one module, *MODULE* is 1.
`-e|--erase`	These options specify that you want to erase a card (rather than create an OCS).

If you have more than one card reader and there is more than one card available, `createocs` prompts you to confirm which card you wish to erase. Use **[Ctrl][X]** to switch between cards.

If you have created a FIPS 140 Level 3 compliant Security World, you must provide authorization in order to erase or create Operator Cards. You can obtain this authorization from any card in the ACS or from any Operator Card in the current Security World, including cards that are to be erased. After you insert a card containing this authorization, `createocs` prompts you to insert the card to be erased.

As an alternative, you can reformat using `slotinfo --format`.

12.3.4. Erasing softcards

Erasing a softcard deletes all information about the softcard from the host. You can erase softcards using KeySafe or with the `ppmk` command-line utility.

12.3.4.1. Erasing softcards with KeySafe

To erase softcards with KeySafe:

1. Start KeySafe.
2. Click the **Softcards** menu button. KeySafe takes you to the Softcard Operations panel.
3. Select the softcard you want to erase from the list.
4. Click the **Discard Softcard** button.
5. KeySafe asks you to confirm that you want to erase this card. Click **Yes** to confirm.
6. After erasing a softcard, KeySafe displays a dialog box to confirm that the card has been erased. Click **OK** to continue using KeySafe.

12.3.4.2. Erasing softcards with ppmk

To erase a softcard with `ppmk`, open a command window, and give the command:

```
ppmk --delete <NAME>|<IDENT>
```

In this command, you can identify the softcard to be erased either by its name (`NAME`) or by its logical token hash as listed by `nfkminfo` (`<IDENT>`).

If you are working within a FIPS 140 Level 3 compliant Security World, you must provide authorization to erase softcards; `ppmk` prompts you to insert a card that contains this authorization. Insert any card from the ACS or any Operator Card from the current Security World.

If you insert an Administrator Card from another Security World or an Operator Card that you have just created, `ppmk` displays an error message and prompts you to insert a card with valid authorization. When `ppmk` has obtained the authorization from a valid card or if no authorization is required, it completes the process of erasing the softcard.

12.4. Viewing cards and softcards

It is often necessary to obtain information from card sets, usually because for security reasons they are left without any identifying markings.

To view details of all the Operator Cards in a Security World or details of an individual Operator Card, you can use KeySafe or the `nfkminfo` command-line utility. To check which passphrase is associated with a card, you can use the `cardpp` command-line utility.

To list all softcards in a Security World or to show details of an individual softcard, you can

use the `ppmk` or `nfkminfo` command-line utilities. To check which passphrase is associated with a softcard, you can use the `ppmk` command-line utility.

12.4.1. Viewing card sets with KeySafe

You can use KeySafe to view details of all the Operator Cards in a Security World, details of individual OCSs or details of an individual Operator Card.

12.4.1.1. Examining a Card

In order to view information about individual cards with KeySafe, follow these steps:

1. Start KeySafe. (For an introduction to KeySafe and information on starting the software, see Using KeySafe.)
2. Click the **Card Sets** menu button, or select the **Card sets** menu item from the **Manage** menu. KeySafe takes you to the **List Operator Card Sets** panel.
3. Click **Examine/Change Card** to open the **Examine/Change Card** panel.
4. Insert a card into the appropriate smart card slot. KeySafe displays information about the smart card currently in the slot. If there is no smart card in the slot, KeySafe displays a message **Card slot empty - please insert the card that you want to examine.**

From the **Examine/Change Card** panel, you can also:

- Change a card's passphrase (if it has one)
- Give a passphrase to a card that does not already have one
- Remove a passphrase from a card that currently has one
- Erase the card.

12.4.1.2. List an Operator Card Set

In order to view information about whole OCSs with KeySafe, follow these steps:

1. Start KeySafe. (For an introduction to KeySafe and information on starting the software, see Using KeySafe.)
2. Click the `Card Sets` menu button, or select the **Card sets** menu item from the **Manage** menu. KeySafe takes you to the `List Operator Card Sets` panel, which displays information about all OCSs in the current Security World.

From the `List Operator Card Sets` panel, you can also:

- Examine / change a card (see Examining a Card)

- Create a new card set (see Creating an Operator Card Set with KeySafe)
- Replace an Operator Card Set (see Replacing OCSs with KeySafe)
- Replace an Administrator Card Set (see Replacing an ACS with KeySafe)
- Discard a card set (see Erasing cards with KeySafe).

12.4.2. Viewing card sets using the command line

You can use the `nfkminfo` command-line utility to view details of either all the Operator Cards in a Security World or of an individual Operator Card.

To list the OCSs in the current Security World from the command line, open a command window, and give the command:

```
nfkminfo --cardset-list
```

In this command, `--cardset-list` specifies that you want to list the operator card sets in the current Security World.

`nfkminfo` displays output information similar to the following:

```
Cardset summary - 1 cardsets:              (in timeout, P=persistent, N=not)
 Operator logical token hash                    k/n timeout name
hash                                           1/1 none-N name
```

To list information for a specific card, use the command:

```
nfkminfo <TOKENHASH>
```

In this command, `<TOKENHASH>` is the `Operator logical token hash` of the card (as listed when the command `nfkminfo --cardset-list` is run).

This command displays output information similar to the following:

```
name            "name"
k-out-of-n      1/1
flags           NotPersistent
timeout         none
card names      ""
hkltu           794ada39038fa8c4e9ea46a24136bbb2b8b337f2
```

Not all software can give names to individual cards.

12.4.3. Viewing softcards

To view softcards, use KeySafe or the command line. The command line provides several options for viewing softcard information.

12.4.3.1. Viewing softcards with KeySafe

To view a softcard with KeySafe, follow these steps:

1. Start KeySafe.
2. Click the **Softcards** menu button. KeySafe takes you to the **Softcard Operations** panel.
3. Click the **List Softcards** navigation button. KeySafe takes you to the **List Softcards** panel, which displays information about all softcards in the current Security World.

   From the **List Softcards** panel, you can also choose to remove a softcard from the Security World. For more information about this procedure, see Erasing cards and softcards.

12.4.3.2. Viewing softcards with nfkminfo

To list the softcards in the current Security World using the `nfkminfo` command-line utility, give the command:

```
nfkminfo --softcard-list
```

In this command `--softcard-list` specifies that you want to list the softcards in the current Security World.

To show information for a specific softcard using the `nfkminfo` command-line utility, give the command:

```
nfkminfo --softcard-list <IDENT>
```

In this command `<IDENT>` is the softcard's logical token hash (as given by running the command `nfkminfo --softcard-list`). This command displays output information similar to the following:

```
SoftCard
 name       "mysoftcard"
 hkltu      7fb95888ea2850d4e3ffcc8f0c22100937344308
Keys protected by softcard 7fb95888ea2850d4e3ffcc8f0c22100937344308:
 AppName simple              Ident mykey
 AppName simple              Ident myotherkey
```

12.4.3.3. Viewing softcards with ppmk

To list the softcards in the current Security World using the `ppmk` command-line utility, use the command:

```
ppmk --list
```

In this command `--list` specifies that you want to list the softcards in the current Security World.

In order to view the details of a particular softcard using the `ppmk` command-line utility, give the command:

```
ppmk --info <NAME>|<IDENT>
```

In this command, you can identify the softcard whose details you want to view either by its name (`<NAME>`) or by its logical token hash (as given by running the command `nfkminfo --softcard-list`).

12.4.4. Verifying the passphrase of a card or softcard

12.4.4.1. Verifying the passphrase of a card with cardpp

To verify the passphrase associated with a card using the `cardpp` command-line utility, use the command:

```
cardpp --check [-m|--module=<MODULE>]
```

This command uses the following options:

Option	Description
`--check`	This option checks the passphrase.
`--module=<MODULE>`	This option specifies the number of the module to use. If you only have one module, `<MODULE>` is 1. If you do not specify a module number, `cardpp` uses all modules by default.

The `cardpp` utility polls all available slots; if there is no card inserted, it prompts you to insert one. If the card belongs to this Security World, `cardpp` either tells you if no passphrase is set or prompts you to enter the passphrase and checks to see if it is correct.

12.4.4.2. Verifying the passphrase of a softcard with ppmk

In order to verify the passphrase of a particular softcard, open a command window, and give the command:

```
ppmk --check <NAME>|<IDENT>
```

In this command, you can identify the softcard whose passphrase you want to verify either by its name (`<NAME>`) or by its logical token hash (as given by running the command `nfkminfo --softcard-list`).

`ppmk` prompts you to enter the passphrase and then tells you whether the passphrase you entered is correct for the specified softcard.

12.5. Changing card and softcard passphrase

Each softcard or card of a card set can have its own individual passphrase: you can even have a card set in which some cards have a passphrase and others do not, and you can have distinct softcards that nevertheless use the same passphrase. A passphrase can be of any length and can contain any characters that you can type.

Normally, in order to change the passphrase of a card or softcard, you need the card or soft card and the existing passphrase. Known card passphrase can be changed using KeySafe or the `cardpp` command-line utility; softcard passphrase can be changed using KeySafe or the `ppmk` command-line utility. You can also add a passphrase to a card or softcard that currently does not have one or remove a passphrase from a card that does currently have one.

If you generated your Security World with the passphrase replacement option, you can also replace the passphrase of a card or softcard even if you do not know the existing passphrase. Such a passphrase replacement operation requires authorization from the ACS.

12.5.1. Changing known passphrase

To change a card passphrase, you need the card and the old passphrase.

Each card in a set can have its own individual passphrase. You can even have a set in which some cards have a passphrase and others do not.

Prior to Security World Software v11.72, we set no absolute limit on the length of a passphrase. However, some applications may not accept a passphrase longer than 255 characters. Likewise, the Security World does not impose restrictions on which characters you can use, although

> some applications may not accept certain characters. Entrust recommends that your password only contains 7-bit ASCII characters:
>
> ```
> A-Z, a-z, 0-9, ! @ # $ % ^ & * - _ + = [ ] { } | \ : ' , . ? / ` ~ " < > ( ) ;
> ```
>
> See Maximum passphrase length for more about passphrase length when using Security World Software v11.72.

12.5.1.1. Changing known passphrase with KeySafe

To change a known passphrase for an Operator Card using KeySafe:

1. Start KeySafe. (For an introduction to KeySafe and information on starting the software, see Using KeySafe.)
2. Click **Card sets**, or select **Card sets** from the **Manage** menu. The **List Operator Card Sets** panel is displayed.
3. Click **Examine / change card** to open the **Examine / Change Card** panel.
4. Click **Change passphrase**. The Set Card Protection passphrase panel is displayed.
5. Enter the old passphrase, and click the **OK** button.
6. A screen is displayed asking **Do you want to set a passphrase?**. Select **Yes**.
7. Enter your new passphrase, and enter it again in the second box as confirmation of the change.
8. Click **OK**.

12.5.1.1.1. Changing a known softcard passphrase with KeySafe

To change a known passphrase for a softcard using KeySafe:

1. Start KeySafe. (For an introduction to KeySafe and information on starting the software, see Using KeySafe.)
2. Click the **Softcards** menu button, or select **Softcards** from the **Manage** menu. KeySafe takes you to the **List Softcards** panel.
3. Select the softcard for which you want to change the passphrase, and click the **Change passphrase** button. KeySafe takes you to the **Change/Recover Softcard passphrase** panel.

   If a softcard is listed as PIN Recovery Enabled = No, then you will be unable to change the passphrase.

4. Select the softcard whose passphrase you want to change, and click the **Change**

passphrase button. KeySafe takes you to the **Get Softcard Protection passphrase** panel.

5. Enter the old passphrase, and click the **OK** button.

   KeySafe either displays an error dialog (if the passphrase is not correct) or takes you to the `Set Softcard Protection` passphrase panel.

6. Enter your new passphrase, and enter it again in the second field to confirm the passphrase is correct.
7. Click the **OK** button.

   After changing a passphrase, KeySafe displays a dialog to confirm that the passphrase has been successfully changes.

8. Click the **OK** button to continue using KeySafe.

12.5.1.2. Changing known passphrase with cardpp

Each card in a card set can have its own individual passphrase. You can even have a set in which some cards have a passphrase and others do not. A passphrase can be of any length and can contain any characters that you can type.

With Security World Software v11.72 and later, passphrases are limited to a maximum length of 254 characters, when using `cardpp`. See Maximum passphrase length.

To change a known card's passphrase with the `cardpp` command-line utility, take the following steps:

1. Run the `cardpp` utility using the command:

   ```
   cardpp --change [-m|--module=<MODULE>]
   ```

   If you only have one HSM, `<MODULE>` is 1. If you do not specify an HSM number, `cardpp` uses all HSMs by default.

2. If prompted, insert the card whose passphrase you want to change. (If there is a card already in the slot, you are not prompted.)
3. If prompted, enter the existing passphrase for the card (If the card has no current passphrase you are not prompted.) If you enter the passphrase correctly, `cardpp` prompts you to enter the new passphrase.
4. Enter a new passphrase, and then enter it again to confirm it.

After you have confirmed the new passphrase, `cardpp` changes the card's passphrase.

12.5.1.3. Changing known softcard passphrase with ppmk

With Security World Software v11.72 and later, passphrases are limited to a maximum length of 254 characters, when using `ppmk`. See Maximum passphrase length for more information.

To change a known softcard's passphrase when you know the passphrase, follow these steps:

1. Give the following command:

   ```
   ppmk --change <NAME>|<IDENT>
   ```

   In this command, you can identify the softcard whose passphrase you want to change either by its name (`<NAME>`) or by its logical token hash as listed by `nfkminfo` (`<IDENT>`).

   `ppmk` prompts you to enter the old passphrase.

2. Type the old passphrase, and press **Enter**. If you enter the old passphrase correctly, `ppmk` prompts you to enter the new passphrase.
3. Type the old passphrase, and press **Enter**. Type the new passphrase again, and press **Enter** to confirm it.

   After you have confirmed the new passphrase, `ppmk` then changes the softcard's passphrase.

12.5.2. Changing unknown or lost passphrase

12.5.2.1. Changing unknown card passphrase with cardpp

If you generated your Security World with the passphrase replacement option, you can change the passphrase of a card even if you do not know its existing passphrase. Such a passphrase replacement operation requires authorization from the ACS.

To change an unknown card passphrase with the `cardpp` command-line utility:

- Run a command of the form:

  ```
  cardpp --recover [--module=<MODULE>]
  ```

In this command, <MODULE> specifies the number of the hardware security module to use. If you only have one hardware security module, <MODULE> is 1. If you do not specify a number, cardpp uses all hardware security modules by default.

- As prompted, insert the appropriate number of cards from the ACS required to authorize passphrase replacement.
- When prompted, insert the Operator Card whose passphrase you want to replace. To replace its passphrase:
  a. When prompted, type the new passphrase, and then press **Enter**.
  b. When prompted, type the new passphrase again to confirm it, and then press **Enter**.

  cardpp sets the new passphrase, and then prompts you for another Operator Card.
- Repeat the process in the previous step to change the passphrase on further cards, or press **Q** to quit.

Only insert Administrator Cards into a hardware security module that is connected to a trusted server.

12.5.2.2. Replacing unknown passphrase with ppmk

If you generated your Security World with the passphrase replacement option, you can change the passphrase of a softcard even if you do not know its existing passphrase. Such a passphrase replacement operation requires authorization from the ACS.

To change an unknown softcard passphrase with the ppmk command-line utility:

1. Run a command of the form:

   ```
   preload --admin=p ppmk --recover <NAME>|<IDENT>
   ```

   In this command, you can identify the softcard by its <NAME> or by its <IDENT> (its logical token hash as shown in output from the nfkminfo command-line utility).
2. As prompted, insert the appropriate number of cards from the ACS required to authorize passphrase replacement.
3. When prompted, type the new passphrase, and then press **Enter**.
4. When prompted, type the new passphrase again to confirm it, and then press **Enter**.

   If the passphrase does not match, ppmk prompts you to input and confirm the passphrase again.

After you successfully confirm the new passphrase, `ppmk` finishes configuring the softcard to use the new passphrase.

Only insert Administrator Cards into a hardware security module that is connected to a trusted server.

12.6. Replacing Operator Card Sets

Replacing an OCS requires authorization from the ACS of the Security World to which it belongs. You cannot replace an OCS unless you have the required number of cards from the appropriate ACS.

If you have lost a card from a card set, or you want to migrate from standard nShield cards to nShield Remote Administration Cards, you should use one of the following:

- The `rocs` utility
- The KeySafe **Replace Operator Card Set** option.

  Accessed from the `Card Operations` panel.

You cannot mix standard nShield cards with nShield Remote Administration Cards. in the same set.

We recommend that after you have replaced an OCS, you then erase the remaining cards in the old card set and remove the old card set from the Security World. For more information, see Erasing cards and softcards.

Deleting the information about an OCS from the host does not remove the data for keys protected by that card set. On the KeySafe **Key Operations** panel), such keys are listed as being protected by **Deleted Card Set**.

To prevent you from losing access to your keys if the smart card you are using as the Operator Card is lost or damaged, Entrust supplies several utilities that can recover the keys protected by the lost Operator Card to another token

- KeySafe includes an option to replace OCSs on the Card Operations panel (click the **Replace OCS** navigation button).
- The `rocs` command-line utility provides an interactive method or a command-line only method to replace OCSs.

Replacing one OCS with another OCS also transfers the keys protected by the first OCS to the protection of the new OCS.

When you replace an OCS or softcard and recover its keys to a different OCS or softcard, the key material is not changed by the process. The process deletes the original host data (that is, the encrypted version of the key or keys and the smart card or softcard data file) and replaces this data with host data protected by the new OCS or softcard.

To replace an OCS or softcard, you must:

- Have enabled OCS and softcard replacement when you created the Security World

  If you did not enable OCS and softcard replacement, or if you created the Security World with an early version of the `pkcs-init` command-line utility that did not support OCS and softcard replacement, you cannot recover keys from lost or damaged smart cards or softcards.

- Have created the original OCS using `createocs`, `createocs-simple`, KeySafe, or the nShield PKCS #11 library version 1.6 or later

  If you initialized the token using `ckinittoken` from the nShield PKCS #11 library version 1.5 or earlier, you must contact Support to arrange for them to convert the token to the new format while you still possess a valid card.

- Have a sufficient number of cards from the ACS to authorize recovery and replacement

  All recovery and replacement operations require authorization from the ACS. If any of the smart cards in the ACS are lost or damaged, immediately replace the entire ACS.

- Have initialized a second OCS using `createocs`, `createocs-simple`, KeySafe, or the nShield PKCS #11 library version 1.6 or later.

  The new OCS need not have the same *K*/*N* policy as the old set.

If you are sharing the Security World across several host computers, you must ensure that the changes to the host data are propagated to all your computers. One way to achieve this is to use client cooperation. For more information, see Setting up client cooperation.

12.6.1. Replacing OCSs with KeySafe

In order to replace an OCS, you must have another OCS onto which to copy the first set's data. If you do not already have an existing second OCS, you must create a new one. For

more information, see Creating Operator Card Sets (OCSs).

When you have a second OCS ready, follow these steps in order to replace the first OCS:

1. Start KeySafe. (For an introduction to KeySafe and information on starting the software, see Using KeySafe.)
2. Click the **Card Sets** menu button, or select **Card Sets** from the **Manage** menu. KeySafe takes you to the **List Operator Card Sets** panel.
3. Click **Replace card set**. KeySafe takes you to the **Replace card set** panel.

   This panel lists existing OCSs in tabular form. For each card set it displays:

Attribute	Description
Name	The name of the card set.
Required (K)	The number of cards needed to re-create a key.
Total (N)	The total number of cards in the set.
Persistent	Indicates whether or not the card set is persistent.
Timeout	The timeout value of the card, in seconds
Recoverable Key Count	The number of private keys protected by this card set that are recoverable.
Nonrecoverable Key Count	The number of private keys protected by this card set that are not recoverable.

   You can click and drag with your mouse in order to resize the column widths and to rearrange the column order of this table. Clicking a column heading sorts the rows in ascending order based on that column heading.

4. Select an OCS that you want to replace, and click **Replace card set**.

   If an OCS does not have any recoverable keys, it cannot be replaced.

5. KeySafe takes you to the **Load Administrator Card Set** panel, where it prompts you to insert cards from the ACS in order to authorize the action. Each time you insert an Administrator Card into the smart card of the hardware security module slot, you must click the **OK** button to load the card.

   Only insert your ACS into a module that is connected to a trusted server.

6. When you have loaded enough cards from the ACS to authorize the procedure, KeySafe takes you to the **Load Operator Card Set** panel, where it prompts you to

insert the OCS that is to protect the recoverable keys (this is the OCS onto which you are copying data from the OCS you are replacing). Each time you insert a card from the new OCS into the smart card slot of the hardware security module, you must click the **OK** button.

When you have loaded enough cards from the new OCS, KeySafe creates new working versions of the recoverable keys that are protected by this card set.

KeySafe deletes the original host data for all recovered keys and replaces this data with host data that is protected by the new OCS. If there are no nonrecoverable keys protected by the card set, KeySafe also removes the old card set from the Security World. However, if the OCS has nonrecoverable keys, the host data for the original card set and for the nonrecoverable keys is not deleted. These keys can only be accessed with the original OCS. If you want to delete these files, use the **Remove OCS** option.

7. When the process is complete, KeySafe displays a dialog indicating that the OCS has been successfully replaced. Click the **OK** button. KeySafe returns you the Replace Operator Card Set panel, where you may replace another OCS or choose a different operation.

12.6.2. Replacing OCSs or softcards with rocs

You can use the `rocs` command-line utility interactively, or you can supply all the parameters using the command line.

12.6.2.1. Using rocs interactively

To use the `rocs` command-line utility interactively, run it without any parameters:

```
rocs
```

`rocs` displays the following prompt:

```
'rocs' key recovery tool
Useful commands: 'help', 'help intro', 'quit'.
rocs >
```

In order to use `rocs` to replace an OCS or recover keys to a softcard, take the following steps:

1. You must select a hardware security module to use by using the `module` command, which is described in the section module <number>.

2. List the OCSs and softcards in the current Security World by using the `list cardsets` command, which is described in the section list cardsets.
3. Select the OCS or softcard to which you want to transfer the keys by using the `target` command, which is described in the section target <cardset-spec>.

Keys protected by an OCS can only be recovered to another OCS, and not to a softcard. Likewise, softcard-protected keys can only be recovered to another softcard, and not to an OCS.

4. List the keys in the current Security World using the `list keys` command, which is described in the section list keys.
5. Select the keys that are to be recovered (from a different OCS or softcard than the one you selected for key transfer) by using the `mark` command, which is described in the section mark <key-spec>.
6. If you have selected any keys by mistake, deselect them by using the `unmark` command, which is described in the section unmark <key-spec>.
7. After you have selected the keys that are to be recovered, transfer these keys by using the `recover` command, which is described in the section recover.

   `rocs` prompts you to insert a card from the ACS.

8. Insert a card from the ACS.

   `rocs` prompts you for the passphrase for this card. This action is repeated until you have loaded the required number of cards from the ACS.

   If you do not have the required number of cards from the ACS, press **Q** and then **Enter**. The `rocs` utility returns you to the `rocs >` prompt without processing any keys.

Only insert Administrator Cards into a hardware security module that is connected to a trusted server.

9. If you are recovering keys to an OCS:

   a. `rocs` prompts you to insert a card from the first OCS that you have selected as the target. OCSs are processed in ascending numerical order as listed by the `list cardsets` command.
   b. Insert a card from this OCS.
   c. `rocs` prompts you for the passphrase for this card. This action is repeated until you have loaded the required number of cards from the OCS.

   If you are recovering keys to a softcard, `rocs` prompts you for the passphrase for the softcard that you have selected as the target.

If you decide that you do not want to transfer the keys to the selected card set or softcard, press **Q** and then **Enter** to quit. `rocs` returns you to the `rocs >` prompt and does not process any further OCSs or softcards.

When you have loaded the target softcard or the required number of cards from the target OCS, `rocs` transfers the selected keys to the target OCS or softcard.

If you have selected other target OCSs or softcards, `rocs` prompts for a card from the next OCS.

10. Repeat step 9 for each selected target.
11. If you have transferred the correct keys, write the key blobs to disk by using the `save` function (described in the section save <key-spec>). If you have transferred a key by mistake, you can restore it to its original protection by using the `revert` command (described in the section revert <key-spec>).

At the `rocs` prompt, you can use the following commands:

- `help <topic>`
- `help intro`
- `list cardsets`
- `list keys`
- `mark <key-spec>`
- `module <number>`
- `quit`
- `recover`
- `rescan`
- `revert <key-spec>`
- `save <key-spec>`
- `status`
- `target <cardset-spec>`
- `unmark <key-spec>`

You can specify a command by typing enough characters to identify the command uniquely. For example, for the `status` command, you can type `st` and then press **Enter**.

12.6.2.1.1. help

With no arguments specified, `help` shows a list of available commands with brief usage mes

sages and a list of other help topics. With an argument, `help` shows detailed help information about a given topic.

`help intro` displays a brief step-by-step guide to using `rocs`.

12.6.2.1.2. list cardsets

This command lists the OCSs and softcards in the current Security World.

For example:

```
No.
Name                    Keys (recov) Sharing
 1 test                    6 (6)        3 of 5; 20 minute timeout
 2 test2                   3 (2)        2 of 3
 3 test3                   1 (1)        1 of 1; persistent
```

In this output:

Output	Description
`No.`	The card set or softcard number, which you can use to identify this card set in `rocs` commands.
`Name`	The OCS or softcard name.
`Keys`	The number of keys protected by this OCS or softcard.
`(recov)`	The number of keys protected by this OCS or softcard.
`Sharing`	The *K* of *N* parameters for this OCS.
`persistent`	The OCS is persistent and does not have a time-out set.
### `minute timeout`	The OCS is persistent and has a time-out set.

12.6.2.1.3. list keys

This command lists the keys in the current Security World, as in the following example:

```
No.
Name                     App        Protected by
 1 rsa-test                 hwcrhk     module
 2 Id: uc63e0ca3cb032d71c1c pkcs11     test2
R 3 Server-Cert              pkcs11     test --> test2
 4 Id: uc63e0ca3cb032d71c1c pkcs11     test --> test3
 5 Server-Cert              pkcs11     module (test ---> fred2)
```

In this output:

Output	Description
`No.`	The key number, which you can use in `mark` and `unmark` commands.
`Name`	The key name.
`App`	The application with which the key is associated.
`Protected by`	This indicates the protection method (see table below).

In this output, the protection methods include:

Method	Description
module	Key protected by the Security World.
name	Key protected by the named OCS or softcard.
name-->name2	Key protected by the OCS or softcard *name1* marked for recovery to OCS or softcard *name2*.
module (name)	PKCS #11 public object. These are protected by the Security World but associated with a specific OCS or softcard.
module (*name-->name2*)	PKCS #11 public object marked for recovery.

12.6.2.1.4. mark <key-spec>

This command marks the listed keys that are to be recovered to the target OCS or softcard. You can mark one or more keys by number, *ident*, OCS or softcard, or hash. For more information, see Specifying keys.

To mark more than one key at a time, ensure that each *key-spec* is separated from the other by spaces, as in the following example:

```
mark key-spec1 key-spec2 key-spec3
```

If you have not selected a target OCS or softcard, or if `rocs` cannot parse the *key-spec*, then `rocs` displays an error message.

You can mark and remark the keys to be recovered to various target OCSs or softcards. Remarking a key displaces the first target in favor of the second target.

Keys protected by an OCS can only be recovered to another OCS, and not to a softcard. Likewise, softcard-protected keys can only be recovered to another softcard, and not to an OCS.

12.6.2.1.5. module <number>

This command selects the hardware security module to be used. The module number must correspond to a hardware security module in the current Security World. If the hardware security module does not exist, is not in the Security World, or is otherwise unusable, then `rocs` displays an error message and does not change to the selected module.

12.6.2.1.6. quit

This command allows you to leave `rocs`. If you attempt to `quit` when you have recovered keys but have not saved them, `rocs` displays a warning.

12.6.2.1.7. recover

This command transfers the marked keys to their target OCSs or softcards. This operation is not permanent until you save these keys by using the `save` command.

12.6.2.1.8. rescan

This command updates the card set and key information.

12.6.2.1.9. revert <key-spec>

This command returns keys that have been recovered, but not saved, to being protected by the original protection method. If the selected keys have not been recovered, `rocs` displays an error message.

12.6.2.1.10. save <key-spec>

This command writes the new key blobs to disk. If you specify a key-spec, only those keys are saved. Otherwise, all recovered keys are saved.

12.6.2.1.11. status

This command lists the currently selected hardware security module and target OCS or soft card.

12.6.2.1.12. target <cardset-spec>

This command selects a given OCS or softcard as the target. You can specify the card set or softcard name, the number returned by `list cardsets`, or the hash.

12.6.2.1.13. unmark <key-spec>

This command unmarks the listed keys. Unmarked keys are not recovered.

12.6.2.2. Using rocs from the command line

You can select all the options for `rocs` using the command line by running a command of the form:

```
rocs -m|--module=<MODULE> [-t|--target=<CARDSET-SPEC>] [-k|--keys=<KEYS-SPEC>] [-c|--cardset=<CARDSET-SPEC>] [-i|--interactive]
```

In this command:

Option	Description
`-m|--module=<MODULE>`	These options specify the number of the hardware security module to use.
`-t|--target=<CARDSET-SPEC>`	These options specify the OCS or softcard to be used to protect the keys. For more information, see Specifying card sets.
`-k|--keys=<KEYS-SPEC>`	These options select the keys to be recovered. For more information, see Specifying keys.
`-c|--cardset=<CARDSET-SPEC>`	These options select all keys that are protected by the given OCS or softcard. For more information, see Specifying card sets.
`-i|--interactive`	These options force `rocs` to start interactively even if you have already selected keys.

You must specify the target before you specify keys.

You can use multiple `--keys=<KEYS-SPEC>` and `--cardset=<CARDSET-SPEC>` options, if necessary.

You can specify multiple targets on one command line by including separate `--keys=<KEYS-SPEC>` or `--cardset=<CARDSET-SPEC>` options for each target. If a key is defined by `--keys=<KEYS-SPEC>` or `--cardset=<CARDSET-SPEC>` options for more than one target, it is transferred to the last target for which it is defined.

If you have selected a hardware security module, a target OCS or softcard, and keys to recover but have not specified the `--interactive` option, `rocs` automatically recovers the keys. `rocs` prompts you for the ACS and OCS or softcard. For more information, see Using rocs interactively.

If you use `rocs` from the command line, all keys are recovered and saved automatically. You cannot revert the keys unless you still have cards

> from the original OCS.

If you do not specify the target and keys to recover, or if you specify the `--interactive` option, `rocs` starts in interactive mode with the selections you have made. You can then use further `rocs` commands to modify your selection before using the `recover` and `save` commands to transfer the keys.

12.6.2.3. Specifying card sets

The value of `<CARDSET-SPEC>` identifies one or more OCSs or softcards. It may have any of the following forms:

Value	Description
`[number]` *cardset-number*	A value of this form selects the OCS or softcard with the given number from the list produced by the `list cardsets` command.
`[name]` *cardset-name*	A value of this form selects card sets or softcards by their names (the card set or softcard name may be a wildcard pattern in order to select all matching OCSs or softcards).
`hash` *cardset-hash*	A value of this form selects the OCS or softcard with the given hash.

In order to specify multiple OCSs or softcards, include several *CARDSET-SPEC*'s using the command line.

> Keys protected by an OCS can only be recovered to another OCS, and not to a softcard. Likewise, softcard-protected keys can only be recovered to another softcard, and not to an OCS.

12.6.2.4. Specifying keys

The `--keys=<KEYS-SPEC>` option identifies one or more keys. It may have any of the following forms:

Value	Description

mark `key-number`	A value of this form selects the key with the given number from the list produced by the list keys command. Examples of usage are:
	`rocs -t <target_OCS> -k <key_number>`
	and
	`rocs -t <target_OCS> -k "mark 56"`

*appname_*:`keyident`	A value of this form selects keys by their internal application name and `ident`. You must supply at least one of `appname` or `keyident`, but you can use wildcard patterns for either or both in order to select all matching keys. An example of usage is: `rocs -t <target_OCS> --keys="simple:simplekey"`
`hash` *keyhash*	A value of this form selects the key with the given key hash. An example of usage is: `rocs -t <target_OCS> --keys="hash e364[...]"`
`--cardset` *cardset-spec*	A value of this form selects all keys protected by a given card set.

12.7. Replacing the Administrator Card Set

Replacing the ACS requires a quorum of cards from the current ACS (*K*/*N*) to perform the following sequence of tasks:

1. loading the secret information that is to be used to protect the archived copy of the Security World key.
2. creating a new secret that is to be shared between a new set of cards.
3. creating a new archive that is to be protected by this secret.

If you discover that one of the cards in the current ACS has been damaged or lost, or you want to migrate from standard nShield cards to nShield Remote Administration Cards, you should use one of the following to create a new set:

- The `racs` utility.

When using the `racs` utility, you cannot redefine the quantities in a K of N relationship for an ACS. The K of N relationship defined in the original ACS persists in the new ACS.

- The KeySafe **Replace Administrator Card Set** option.

  Accessed from the `Card Operations` panel.

If further cards are damaged, you may not be able to re-create your Security World.

You cannot mix nShield cards with nShield Remote Administration Cards in the same set.

Replacing the ACS modifies the `world` file. In order to use the new ACS on other machines in the Security World, you must copy the updated `world` file to all the machines in the Security World after replacing the ACS. Failure to do so could result in loss of administrative access to the Security World.

We recommend that you erase your old Administrator Cards as soon as you have created the new ACS. An attacker with the old ACS and a copy of the old host data could still re-create all your keys. With a copy of a current backup, they could even access keys that were created after you replaced the ACS.

Before you start to replace an ACS, you must ensure that you have enough blank cards to create a complete new ACS. If you start the procedure without enough cards, you will have to cancel the procedure part way through.

12.7.1. Replacing an ACS with KeySafe

When you have enough cards to create a complete new ACS ready and a quorum of the ACS you want to replace, follow these steps:

1. Start KeySafe. (For an introduction to KeySafe and information on starting the software, see Using KeySafe).
2. Click the **Card sets** menu button, or select **Card sets** from the **Manage** menu. KeySafe takes you to the **List Operator Card Sets** panel.
3. Click the **Replace ACS** navigation button, and KeySafe takes you to the **Replace Administrator Card Set** panel.
4. If you are sure that you want to replace the ACS, click the **Replace ACS** command button
5. KeySafe takes you to the **Load Administrator Card Set** panel, where it prompts you to

insert cards from the ACS in order to authorize the action. Each time you insert an Administrator Card into the module's smart card slot, you must click the **OK** button to load the card.

Only insert cards from your ACS into a module that is connected to a trusted server.

6. When you have loaded enough Administrator Cards to authorize the action, KeySafe takes you to the Create Administrator Card Set panel, where it prompts you to insert the cards that are to form the ACS. These must be blank cards or cards that KeySafe can erase. KeySafe will not let you use cards from the existing ACS. If you do not have enough cards to form a complete new ACS, cancel the operation now.

When creating a card set, KeySafe recognizes cards that belongs to the set even before the card set is complete. If you accidentally insert a card to be written again after it has already been written, KeySafe displays a warning.

7. When you insert a blank card, KeySafe takes you to the **Set Card Protection passphrase** panel.
8. If you want to set a passphrase for this Administrator Card:
   a. Select the **Yes** option.
   b. Enter the same passphrase in both text fields.
   c. Click the **OK** button.

   KeySafe then prompts you for the next card (if any). A given passphrase is associated with a specific card, so each card can have a different passphrase. You can change these passphrases at any time by using the KeySafe **Examine/Change Card** option (available from the **List Operator Card Sets** panel) or the `cardpp` command-line utility.
9. If you do not want to set a passphrase for this Administrator Card:
   a. Select the **No** option.
   b. Click the **OK** button.
10. After you have created all the Administrator Cards, KeySafe displays a message confirming that the ACS has been successfully replaced.
11. Click the **OK** button, and KeySafe returns you to its introduction panel.

When you have finished replacing the ACS, erase the old Administrator Cards; for more information, see Erasing cards and softcards.

12.7.2. Replacing an Administrator Card Set using racs

The `racs` utility creates a new ACS to replace a set that was created with the `new-world` utility

When using the `racs` utility, you cannot redefine the quantities in a K of N relationship for an ACS. The K of N relationship defined in the original ACS persists in the new ACS.

1. Ensure the hardware security module is in operational mode.
2. Run a command of the form:

   ```
   racs [-m|--module=MODULE]
   ```

   In this command, the -m|--module=*MODULE* option specifies the `ModuleID` (*MODULE*) of the module to use.

3. When prompted, insert the appropriate quorum of Administrator Cards to authorize the replacement.
4. When prompted that `racs` is writing the new ACS, insert blank cards as necessary on which to write the replacement Administrator Cards.
5. When you have finished replacing the ACS, erase the old Administrator Cards. For more information, see Erasing cards and softcards.

13. Application interfaces

This chapter explains how to use an HSM with various types of application:

- nCipherKM JCA/JCE CSP
- PKCS #11 applications
- nShield native and Custom applications
- CodeSafe applications

You can use KeySafe or the `generatekey` utility to generate or import keys for use with your applications (see Working with keys). By default, KeySafe uses the same mechanisms and supports the same applications as the `generatekey` utility.

You must add the user of any application that uses an nShield HSM to the group **nfast** before the application runs.

13.1. nCipherKM JCA/JCE CSP

The nCipherKM JCA/JCE CSP (Cryptographic Service Provider) allows Java applications and services to access the secure cryptographic operations and key management provided by Entrust hardware. This provider is used with the standard JCE (Java Cryptographic Extension) programming interface.

To use the nCipherKM JCA/JCE CSP, you must install:

- the nShield Java package which includes the nShield Java jars and KeySafe.

For more information about the bundles and components supplied on your Security World Software installation media, see the User Guide.

The following versions of Java have been tested to work with, and are supported by, your nShield Security World Software:

- Java 8 (or Java 1.8x)
- Java 11
- Java 17

We recommend that you ensure Java is installed before you install the Security World Software. The Java executable must be on your system path.

If you can do so, please use the latest Java version currently supported by Entrust that is compatible with your requirements. Java versions before those shown are no longer supported. If you are maintaining older Java versions for legacy reasons, and need compatibil-

ity with current nShield software, please contact Entrust nShield Technical Support, https://nshieldsupport.entrust.com.

To install Java you may need installation packages specific to your operating system, which may depend on other pre-installed packages to be able to work.

Suggested links from which you may download Java software as appropriate for your operating system:

- http://www.oracle.com/technetwork/java/index.html
- http://www.oracle.com/technetwork/java/all-142825.html

Detailed documentation for the JCE interface can be found on the Oracle Technology web page https://docs.oracle.com/en/java/javase/11/security/java-cryptography-architecture-jca-reference-guide.html.

Softcards are not supported for use with the nCipherKM JCA/JCE CSP in Security Worlds that are compliant with FIPS 140 Level 3.

13.1.1. Installing the nCipherKM JCA/JCE CSP

To install the nCipherKM JCA/JCE CSP:

1. In the hardserver configuration file, ensure that:
   - `priv_port` (the port on which the hardserver listens for local privileged TCP connections) is set to 9001
   - `nonpriv_port` (the port on which the hardserver listens for local nonprivileged TCP connections) is set to 9000.

     If you need to change either or both of these port settings, you restart the hardserver before continuing the nCipherKM JCA/JCE CSP installation process. For more information, see Stopping and restarting the hardserver.
2. For Java 8 only. Copy the `nCipherKM.jar` file to the extensions folder of your local Java Virtual Machine installation from the following directory:
   - `/opt/nfast/java/classes`

     The location of the extensions folder depends on the type of your local Java Virtual Machine (JVM) installation:

JVM type	Extensions folder
Java Developer Kit (JDK)	`$JAVA_HOME/jre/lib/ext`

JVM type	Extensions folder
Java Runtime Environment (JRE)	`$JAVA_HOME/lib/ext`

In these paths, `$JAVA_HOME` is the home directory of the Java installation (commonly specified in the `JAVA_HOME` environment variable). If you are using Java 11 or Java 17 you do not need to copy the jar file.

3. Add `$JAVA_HOME/bin` to your PATH system variable
4. For Java 8 only. Install the unlimited strength JCE jurisdiction policy files that are appropriate to your version of Java. JDK 9 and later ship with, and use by default, the unlimited policy files.

   The Java Virtual Machine imposes limits on the cryptographic strength that may be used by default with JCE providers. Replace the default policy configuration files with the unlimited strength policy files.

   To install the unlimited strength JCE jurisdiction policy files:

   a. If necessary, download the archive containing the Java Cryptography Extension (JCE) Unlimited Strength Jurisdiction Policy Files from the Web site of your Java Virtual Machine vendor.

   The Java Cryptography Extension (JCE) Unlimited Strength Jurisdiction Policy Files are covered and controlled by U.S. Export Control laws and may be subject to the export or import laws in other countries. We recommend that you take legal advice before downloading these files from your Java Virtual Machine vendor.

   b. Extract the files `local_policy.jar` and `US_export_policy.jar` from Java Virtual Machine vendor's Java Cryptography Extension (JCE) Unlimited Strength Jurisdiction Policy File archive.
   c. Copy the extracted files `local_policy.jar` and `US_export_policy.jar` into the security directory for your local Java Virtual Machine (JVM) installation:

JVM type	Extensions folder
Java Developer Kit (JDK)	`$JAVA_HOME/jre/lib/security`
Java Runtime Environment (JRE)	`$JAVA_HOME/lib/security`

   In these paths, `$JAVA_HOME` is the home directory of the Java installation (commonly specified in the `JAVA_HOME` environment variable).

Copying the files `local_policy.jar` and `US_export_policy.jar` into the appropriate folder must overwrite any existing files with the same names.

5. Add the nCipherKM provider to the `java.security` file located in the security directory for your local Java Virtual Machine (JVM) installation: `security.provider.<n>=com.ncipher.provider.km.nCipherKM`, where `<n>` is the position in the list of providers, for example:

```
security.provider.1=sun.security.provider.Sun
security.provider.2=sun.security.rsa.SunRsaSign
security.provider.3=com.sun.net.ssl.internal.ssl.Provider
security.provider.4=com.sun.crypto.provider.SunJCE
security.provider.5=sun.security.jgss.SunProvider
security.provider.6=com.sun.security.sasl.Provider
security.provider.7=com.ncipher.provider.km.nCipherKM
```

   For Java 11 and Java 17 you do not need to specify the fully qualified class name for the provider. Instead you can just use the provider name: `security.provider.<n>=nCipherKM`.

   The JVM uses this file to select the provider from which to request a mechanism instance. If your JCE application does not request the nCipherKM provider by name, or if it fails to load keys, you might need to move the nCipherKM provider to the top of the list: `security.provider.1=com.ncipher.provider.km.nCipherKM`. Do not change the relative order of the other providers in the list.

   Ensure you do not list multiple providers with the same number (for example, ensure your list of providers does not include two instances of `security.provider.1`, both `com.ncipher.provider.km.nCipherKM` and another provider). If you add the nCipherKM provider as `security.provider.1`, ensure that the subsequent providers are re-numbered correctly.

6. Save your updates to the file `java.security`.

   When you have installed the nCipherKM JCA/JCE CSP, you must have created a Security World before you can test or use it. For more information about creating a Security World, see Creating a Security World.

   If you have a Java Enterprise Edition Application Server running, you must restart it before the installed nCipherKM provider is loaded into the Application Server virtual machine and ready for use.

13.1.1.1. Testing the nCipherKM JCA/JCE CSP installation

After installation, you can test that the nCipherKM JCA/JCE CSP is functioning correctly by running the command.

For Java 8:

```
java com.ncipher.provider.InstallationTest
```

For Java 11 and Java 17:

```
java --module-path /opt/nfast/java/classes com.ncipher.provider.InstallationTest
```

For this command to work, you must have added `$JAVA_HOME` to your `PATH` system variable).

If the nCipherKM JCA/JCE CSP is functioning correctly, output from this command has the following form:

```
Installed providers:
1: nCipherKM
2: SUN
3: SunRsaSign
4: SunJSSE
5: SunJCE
6: SunJGSS
7: SunSASL
Unlimited strength jurisdiction files are installed.
The nCipher provider is correctly installed.
nCipher JCE services:
Alg.Alias.Cipher.1.2.840.113549.1.1.1
Alg.Alias.Cipher.1.2.840.113549.3.4
Alg.Alias.Cipher.AES
Alg.Alias.Cipher.DES3
```

If the `nCipherKM` provider is installed but is not registered at the top of the providers list in the `java.security` file, the `InstallationTest` command produces output that includes the message:

```
The nCipher provider is installed, but is not registered at the top of the providers list in the java.security
file.
See the user guide for more information about the recommended system configuration.
```

In such a case, edit the `java.security` file (located in the security directory for your local JVM installation) so that the nCipherKM provider is registered in the first position in that file's list of providers. For more information about the `java.security` file, see Installing the nCipherKM JCA/JCE CSP.

If the nCipherKM provider is not installed at all, or you have not created a Security World, or if you have not configured ports correctly in the hardserver configuration file, the `InstallationTest` command produces output that includes the message:

```
The nCipher provider is not correctly installed.
```

In such case:

- Check that you have configured ports correctly, as described in Installing the nCipherKM JCA/JCE CSP. For more information about hardserver configuration file settings, see server_startup.
- Check that you have created a Security World. If you have not created a Security World, create a Security World. For more information, see Creating a Security World.
- If you have already created a Security World, repeat the nCipherKM JCA/JCE CSP installation process as described in Installing the nCipherKM JCA/JCE CSP.

After making any changes to the nCipherKM JCA/JCE CSP installation, run the `InstallationTest` command again and check the output.

Whether or not the nCipherKM provider is correctly installed, if the unlimited strength jurisdiction files are not installed or (not correctly installed), the `InstallationTest` command produces output that includes the message:

```
Unlimited strength jurisdiction files are NOT installed.
```

This message means that, because the Java Virtual Machine imposes limits on the cryptographic strength that you can use by default with JCE providers, you must replace the default policy configuration files with the unlimited strength policy files. For information about how to install the unlimited strength jurisdiction files, see Installing the nCipherKM JCA/JCE CSP.

13.1.2. Named Modules in Java 11 and Java 17

The nCipherKM Provider has been implemented as a named module. This means that, for Java 11 and Java 17, if you have added the provider to your `java.security` file, then you can run your application with the `nCipherKM.jar` on the module-path and the Java ServiceLoader class will automatically find it, e.g.

```
java --module-path /opt/nfast/java/classes com.ncipher.provider.InstallationTest
```

Alternatively, you can specify the location of the nCipherKM jar on the classpath:

```
java --class-path /opt/nfast/java/classes/nCipherKM.jar com.ncipher.provider.InstallationTest
```

13.1.3. keytool

Use the Java `keytool` utility to read and edit an nShield KeyStore. You must specify the correct `nCipher.sworld` KeyStore type when you run the `keytool` utility.

To generate a new key in an OCS-protected KeyStore with the Java `keytool` utility, run the following command:

```
keytool -genkeypair -storetype nCipher.sworld -keyalg RSA -sigalg SHA1withRSA -storepass <KeyStore_passphrase>
-keystore <KeyStore_path>
```

In this example command, `KeyStore_passphrase` is the passphrase for the OCS protecting the KeyStore. `KeyStore_path` is the path to the KeyStore.

To generate a new key in a module-protected KeyStore with the Java `keytool` utility, run the following command:

```
keytool -J-Dprotect=module -J-DignorePassphrase=true -genkeypair -storetype nCipher.sworld -keyalg RSA -sigalg
SHA1withRSA -keystore <KeyStore_path>
```

In this example command, `<KeyStore_path>` is the path to the KeyStore.

By default, the `keytool` utilities use the `MD5withRSA` signature algorithm to sign certificates used with a KeyStore. This signature mechanism is unavailable on modules with firmware version 2.33.60 or later.

13.1.4. Using keys

Only the nCipherKM provider can use keys stored in an nShield KeyStore because the under lying key material is held separately in the Security World.

You can always store nShield keys in an nShield KeyStore. You can also store keys generated by a third-party provider into an nShield KeyStore if both of the following conditions apply:

- the key type is known to the nCipherKM provider
- the Security World is *not* compliant with FIPS 140 Level 3.

When you generate an nShield key (or create it from imported key material), that key is associated with an ACL (Access Control List). This ACL prevents the key from being used for

operations for which it is unsuited and enforces requirements that certain tokens be presented; for example, the ACL can specify that signing key cannot be used for encryption.

13.1.5. System properties

You can use system properties to control the provider. You set system properties when starting the Java Virtual Machine using a command such as:

```
java -D<property>=<value> <MyJavaApplication>
```

In this example command, `<property>` represents any system property, `<value>` represents the value set for that property, and `<MyJavaApplication>` is the name of the Java application you are starting. You can set multiple system properties in a single command, for example:

```
java -Dprotect=module -DignorePassphrase=true <MyJavaApplication>
```

The available system properties and their functions as controlled by setting different values for a property are described in the following table:

Property	Function for different values
`JCECSP_DEBUG`	This property is a bit mask for which different values specify different debugging functions; the default value is `0`. For details about the effects of setting different values for this property, see JCECSP_DEBUG property values.
`JCECSP_DEBUGFILE`	This property specifies a path to the file to which logging output is to be written. Set this property if the `JCECSP_DEBUG` property is set to a value other than the default of `0`. For details about the effects of setting different values for this property, see JCECSP_DEBUG property values. In a production environment, we recommend that you disable debug logging to prevent sensitive information being made available to an attacker.
`protect`	This property specifies the type of protection to be used for key generation and nCipherKM KeyStore instances. You can set the value of this property to one of `module`, `<SOFTCARD_NAME:SOFTCARD_IDENT>`, or `cardset`. OCS protection (`cardset`) uses the card from the first slot of the first usable hardware security module. To find the logical token hash `<IDENT>` of a softcard, run the command `nfkminfo --softcard-list`.
`module`	This property lets you override the default HSM and select a specific HSM to use for HSM and OCS protection. Set the value of this property as the ESN of the HSM you want to use.

Property	Function for different values
slot	This property lets you override the default slot for OCS-protection and select a specific slot to use. Set this the value of this property as the number of the slot you want to use.
ignorePassphrase	If the value of this property is set to true, the nCipherKM provider ignores the passphrase provided in its KeyStore implementation. This feature is included to allow the Oracle or IBM keytool utilities to be used with module-protected keys. The keytool utilities require a passphrase be provided; setting this property allows a dummy passphrase to be used.
seeintegname	Setting the value of this property to the name of an SEE integrity key causes the provider to generate SEE application keys. These keys may only be used by an SEE application signed with the named key.
com.ncipher.provider.announcemode	The default value for this property is auto, which uses firmware auto-detection to disable algorithms in the provider that cannot be supported across all installed HSMs. Setting the value of this property to on forces the provider to advertise all mechanisms at start-up. Setting the value of this property to off forces the provider to advertise no mechanisms at start-up.
com.ncipher.provider.enable	For the value of this property, you supply a comma-separated list of mechanism names that are to be forced on, regardless of the announce mode selected.
com.ncipher.provider.disable	For the value of this property, you supply a comma-separated list of mechanism names that are to be forced off, regardless of the announce mode selected. Any mechanism supplied in the value for the com.ncipher.provider.disable property overrides the same mechanism if it is supplied in the value for the com.ncipher.provider.enable property.

13.1.5.1. JCECSP_DEBUG property values

The JCECSP_DEBUG system property is a bit mask for which you can set different values to control the debugging functions. The following table describes the effects of different values that you can set for this property:

JCECSP_DEBUG value	Function
0	If this property has no bits set, no debugging information is reported. This is the default setting.
1	If this property has the bit 1 set, minimal debugging information (for example, version information and critical errors) is reported.
2	If this property has the bit 2 set, comprehensive debugging information is reported.

JCECSP_DEBUG value	Function
4	If this property has the bit 3 set, debugging information relating to creation and destruction of memory and HSM resources is reported.
8	If this property has the bit 4 set, debugFunc and debugFuncEnd generate debugging information for functions that call them.
16	If this property has the bit 5 set, debugFunc and debugFuncEnd display the values for all the arguments that are passed in to them.
32	If this property has the bit 6 set, context information is reported with each debugging message (for example, the ThreadID and the current time).
64	If this property has the bit 7 set, the time elapsed during each logged function is calculated, and information on the number of times a function is called and by which function it was called is reported.
128	If this property has the bit 8 set, debugging information for NFJAVA is reported in the debugging file.
256	If this property has the bit 9 set, the call stack is printed for every debug message.

To set multiple logging functions, add up the JCECSP_DEBUG values for the debugging functions you want to set, and specify the total as the value for JCECSP_DEBUG. For example, if you want to set the debugging to use both function tracing (bit 4) and function tracing with parameters (bit 5), add the JCECSP_DEBUG values shown in the table for these debugging functions (8 + 16 = 24) and specify this total (24) as the value to use for JCECSP_DEBUG.

13.1.6. Compatibility

The nCipherKM JCA/JCE CSP supports both module-protected keys and OCS-protected keys. The CSP currently supports 1/*N* OCSs and a single protection type for each nCipherKM JCE KeyStore.

You can use the nCipherKM JCA/JCE CSP with Security Worlds that comply with FIPS 140 at either Level 2 or Level 3.

In a Security World that complies with FIPS 140 Level 3, it is not possible to import keys generated by other JCE providers.

The nCipherKM JCA/JCE CSP supports load-sharing for keys that are stored in the nCipherKM KeyStore. This feature allows a server to spread the load of cryptographic operations across multiple connected HSMs, providing greater scalability.

We recommend that you use load-sharing unless you have existing

code that is designed to run with multiple HSMs. To share keys with load-sharing, you must create a 1/*N* OCS with at least as many cards as you have HSMs. All the cards in the OCS must have the same passphrase.

The nCipherKM JCA/JCE CSP does not support HSM Pool mode. If you want to use HSM Pool mode with a Java application that only uses module protected keys, one option may be to use the Sun PKCS #11 provider to access the nShield PKCS #11 library instead of using nCipherKM JCA/JCE CSP.

Keys generated or imported by the nCipherKM JCA/JCE CSP are not recorded into the Security World until:

1. The key is added to an nCipherKM KeyStore (by using a call to `setKeyEntry()` or `setCertificateEntry()`).
2. That nCipherKM KeyStore is then stored (by using a call to `store()`).

The passphrase used with the KeyStore must be the passphrase of the card from the OCS that protects the keys in the KeyStore.

13.2. nShield PKCS #11 library

To use the nShield PKCS #11 library, you must tell the application the name and location of the library. The exact method for doing this depends on the application.

Instructions for using the nShield PKCS #11 library with specific applications are available from Entrust nShield Technical Support, https://nshieldsupport.entrust.com.

Depending on the application, you may need to set the path and library name `/opt/nfast/toolkits/pkcs11/libcknfast.so` in a dialog or configuration file.

The nShield PKCS #11 library has security options which you must configure before you use the PKCS #11 library. For more information, see PKCS #11 library with Security Assurance Mechanism.

From version 1.7, the nShield PKCS #11 library can be used with FIPS 140 Level 3 compliant Security Worlds. This version of the library also introduces load-sharing mode. This feature provides support for multiple hardware security modules that are connected to a single server, spreading the load of cryptographic operations between the HSMs in order to provide scalability in terms of performance.

To share OCS protected keys with load-sharing mode, you must create a 1/*N* OCS that con

tains at least as many cards as you have HSMs. All the cards on the OCS must have the same passphrase.

With module firmware version 2.65.2 or later, if your application only uses module protected keys, you can use HSM Pool mode as an alternative to using load-sharing mode. HSM Pool mode supports returning or adding a hardware security module to the pool without restarting the system.

If you are using the `preload` command-line utility in conjunction with the nShield PKCS #11 library, you can create *K*/*N* OCSs.

13.2.1. Choosing functions

Some PKCS #11 applications enable you to choose which functions you want to perform on the PKCS #11 token and which functions you want to perform in your application.

The following paragraphs in this section describe the functions that an nShield HSM can provide.

13.2.1.1. Generating random numbers and keys

The nShield HSM includes a hardware random number generator. A hardware random number generator provides greater security than the pseudo-random number generators provided by host computers. Therefore, always use the nShield HSM to generate random numbers and keys.

13.2.1.2. Digital signatures

The nShield PKCS #11 library can use the nShield HSM to sign and verify messages using the following algorithms:

- DSA
- RSA
- DES3_MAC
- AES
- ECDSA (if the appropriate feature is enabled)

An nShield hardware security module is specifically optimized for public key algorithms, and therefore it will provide significant acceleration for DSA, RSA and ECDSA signature generation and verification. You should always choose to perform asymmetric signature generation and verification with an nShield HSM.

13.2.1.3. Asymmetric encryption

The nShield PKCS #11 library can use an nShield HSM to perform asymmetric encryption and decryption with the RSA algorithm.

The nShield HSM is specifically optimized for asymmetric algorithms, so you should always choose to perform asymmetric operations with the nShield HSM.

13.2.1.4. Symmetric encryption

The nShield PKCS #11 library can use the nShield HSM to perform symmetric encryption with the following algorithms:

- DES
- Triple DES
- AES

Because of limitations on throughput, these operations can be slower on the nShield HSM than on the host computer. However, although the nShield HSM may be slower than the host under a light load, you may find that under a heavy load the advantage gained from off-loading the symmetric cryptography (which frees the host CPU for other tasks) means that you achieve better overall performance.

13.2.1.5. Message digest

The nShield PKCS #11 library can perform message digest operations with MD5, SHA-1, SHA-224, SHA-256, SHA-384, and SHA-512 algorithms. However, for reasons of throughput, the library performs these operations on the host computer.

13.2.1.6. Mechanisms

The mechanisms currently supported by the nShield PKCS #11 library, including some vendor-supplied mechanisms, are listed in the *Cryptographic API Integration Guide*.

13.2.1.7. Key wrapping

The nShield PKCS #11 library can use an nShield HSM to wrap (encrypt) a private or secret key, or to unwrap (decrypt) a wrapped key.

The mechanisms supported by the nShield PKCS #11 library, including some vendor-supplied mechanisms, are listed in the *Cryptographic API Integration Guide*.

13.2.2. PKCS #11 library with Security Assurance Mechanism

It is possible for an application to use the PKCS #11 API in ways that do not necessarily provide the expected security benefits, or which might introduce additional weaknesses. For example, the PKCS #11 standard requires the nShield library to be able to generate keys that are extractable from the HSM in plaintext. An application could use this ability in error, when a secure key would be more appropriate.

The PKCS #11 library with the Security Assurance Mechanism (SAM), `libcknfast`, can help users to identify potential weaknesses, and help developers create secure PKCS #11 applications more easily.

The SAM in the PKCS #11 library is intended to detect operations that reveal questionable behavior by the application. If these occur, the application fails with an explanation of the cause of failure.

After a review of your security policy and the way the application uses the PKCS #11 library with the SAM, if there are questionable operations that are considered to be acceptable and pose no security risk, the PKCS #11 library can be configured to permit some, or all, of them by means of the `CKNFAST_OVERRIDE_SECURITY_ASSURANCES` environment variable (described in CKNFAST_OVERRIDE_SECURITY_ASSURANCES).

To ensure the security of your keys, you must review any messages returned by the PKCS #11 library before changing the settings of the `CKNFAST_OVERRIDE_SECURITY_ASSURANCES` environment variable.

The `CKNFAST_OVERRIDE_SECURITY_ASSURANCES` environment variable uses a semicolon separated list of parameters, with associated values, to explicitly allow operations that could compromise the security of cryptographic keys if the operations are not well understood.

If no parameters, or the `none` parameter, are supplied to the `CKNFAST_OVERRIDE_SECURITY_ASSURANCES`, the PKCS #11 library fails to perform the operation in question, and issues a warning, when the following operations are detected:

- Creating short-term session keys as long-term objects
- Creating keys that can be exported as plain text
- Importing keys from external sources
- Creating or importing wrapping keys
- Creating or importing unwrapping keys
- Creating keys with weak algorithms (such as DES)
- Creating keys with short key lengths.

For more information about parameters and diagnostic warnings, see CKNFAST_OVER-

RIDE_SECURITY_ASSURANCES.

13.2.2.1. Key security

Questionable operations largely relate to the concept of a key being *secure*. A private or secret key is considered insecure if there is some reason for believing that its value may be available outside the HSM. Public keys are never considered insecure; by definition they are intended to be public.

An explicitly insecure PKCS #11 key is one where `CKA_SENSITIVE` is set to false. If an application uses a key that is insecure but `CKA_SENSITIVE` is not set to false, it is possible that the application is using an inadequate concept of key security, and that the library disallows use of that key by default. Use of insecure keys should, by default, be restricted to short-term session keys, and applications should explicitly recognize the insecurity.

13.2.3. Using the nShield PKCS #11 library

After you have loaded the nShield PKCS #11 library, it is added to your application's list of cryptographic HSMs or PKCS #11 slots.

Whether or not the library uses load-sharing mode depends on the value of the `CKNFAST_LOADSHARING` environment variable, described in CKNFAST_LOADSHARING. Whether or not the library uses HSM Pool mode depends on the value of the `CKNFAST_HSM_POOL` environment variable, described in CKNFAST_HSM_POOL.

13.2.3.1. nShield PKCS #11 library with load-sharing mode

If load-sharing mode is enabled, the nShield PKCS #11 library creates a virtual slot for each OCS in the Security World (returning the name of the card set) unless you have set `CKNFAST_CARDSET_HASH` (as described in CKNFAST_CARDSET_HASH).

An additional virtual slot may be returned (with the label of `accelerator`), depending on the value given to the variable `CKNFAST_NO_ACCELERATOR_SLOTS` (described in CKNFAST_NO_ACCELERATOR_SLOTS). Accelerator slots can:

- Be used to support session objects
- Be used to create module-protected keys
- Not be used to create private objects.

When you insert a smart card from an OCS in the current Security World, the nShield PKCS #11 library treats this card as a PKCS #11 token that is present in the virtual slot for

that OCS.

After the PKCS #11 token is present, you can open a session to that token. Until you log in, a session can only access public objects that belong to that PKCS #11 token.

The PKCS #11 token is present until you remove the last card belonging to the OCS. When you remove the token, the nShield PKCS #11 library closes any open sessions.

Logging in gives access to the private objects that are protected by the PKCS #11 token. Logging in requires the passphrase for the OCS. The exact mechanism for supplying the passphrase depends on the application that you are running.

The PKCS #11 token is shared across all the HSMs that have a smart card from the OCS in the reader at the point that you log in. After you have logged in, inserting additional cards from this OCS has no effect.

If you remove a smart card that belongs to a logged-in token, the nShield PKCS #11 library closes any open sessions and marks the token as being not present (unless the OCS is persistent). Removing a card from a persistent OCS has no effect, and the PKCS #11 token remains present until you log out.

13.2.3.2. nShield PKCS #11 library with HSM Pool mode

If HSM Pool mode is enabled, the nShield PKCS #11 library exposes a single pool of HSMs and a single virtual slot for a fixed token with the label `accelerator`. This accelerator slot can be used to create module protected keys and to support session objects. HSM Pool mode does not support token protected keys, any pre-existing OCS or softcard protected keys are hidden from PKCS #11. In FIPS 140 Level 3 Security Worlds, keys cannot be created in HSM Pool mode, however keys created outside HSM Pool mode, for example using `generatekey` or a non-Pool mode PKCS #11 application, can be used in HSM Pool mode.

13.2.3.3. nShield PKCS #11 library without load-sharing

There will be two entries for each HSM, unless you have set `CKNFAST_NO_ACCELERATOR_SLOTS`.

The entry called `accelerator` cannot be used to create private objects. It can be used to create module-protected keys.

Use the second of the two entries (which has the same name as the Operator Card that is currently in a smart card reader) to protect your keys or token objects.

PKCS #11 does not allow two tokens to be present in the same slot. Therefore, when you insert a smart card into a reader, the nShield PKCS #11 library logs out any previously

logged-in token from the slot and closes any open sessions.

13.2.3.4. nShield PKCS #11 library with the preload utility

You can use the `preload` command-line utility to preload *K*/*N* OCSs before actually using PKCS #11 applications. The `preload` utility loads the logical token and then passes it to the PKCS #11 utilities.

You must provide any required passphrase for the tokens when using `preload` to load the card set. However, because the application is not aware that the card set has been pre-loaded, the application operates normally when handling the login activity (including prompting for a passphrase), but the PKCS #11 library will not actually check the supplied passphrase. `preload` must be also used with the `cksotool` utility to perform operations that require the PKCS #11 Security Officer role.

Normally, `preload` uses environment variables to pass information to the program using the preloaded objects, including the PKCS #11 library. Therefore, if the application you are using is one that clears its environment before the PKCS #11 library is loaded, you must set the appropriate values in the `cknfastrc` file (see nShield PKCS #11 library environment variables). The current environment variables remain usable. The default setting for the `CKNFAST_LOADSHARING` environment variable changes from specifying load-sharing as disabled to specifying load-sharing as enabled. Moreover, in load-sharing mode, the loaded card set is used to set the environment variable `CKNFAST_CARDSET_HASH` so that only the loaded card set is visible as a slot.

The `NFAST_NFKM_TOKENSFILE` environment variable must also be set in the `cknfastrc` file to the location of the preload file (see Environment variables).

A logical token preloaded by `preload` for use with the nShield PKCS #11 library is the only such token available to the application for the complete invocation of the library. You can use more than one HSM with the same card set.

If the loaded card set is non-persistent, then a card must be left in each HSM on which the set has been loaded during the start-up sequence. After a non-persistent card has been removed, the token is not present even if the card is reinserted.

If load-sharing has been specifically switched off, you see multiple slots with the same label.

13.2.4. nShield PKCS #11 library environment variables

The nShield PKCS #11 library uses the following environment variables:

- CKNFAST_ASSUME_SINGLE_PROCESS
- CKNFAST_ASSURANCE_LOG
- CKNFAST_CARDSET_HASH
- CKNFAST_CONCATENATIONKDF_X963_COMPLIANCE
- CKNFAST_DEBUG
- CKNFAST_DEBUGDIR
- CKNFAST_DEBUGFILE
- CKNFAST_DH_LSB
- CKNFAST_FAKE_ACCELERATOR_LOGIN
- CKNFAST_HSM_POOL
- CKNFAST_JCE_COMPATIBILITY
- CKNFAST_LOADSHARING
- CKNFAST_LOAD_KEYS
- CKNFAST_NO_ACCELERATOR_SLOTS
- CKNFAST_NO_SYMMETRIC
- CKNFAST_NO_UNWRAP
- CKNFAST_NONREMOVABLE
- CKNFAST_NVRAM_KEY_STORAGE
- CKNFAST_OVERRIDE_SECURITY_ASSURANCES
- CKNFAST_RELOAD_KEYS
- CKNFAST_SEED_MAC_ZERO
- CKNFAST_SESSION_THREADSAFE
- CKNFAST_TOKENS_PERSISTENT
- CKNFAST_USE_THREAD_UPCALLS
- CKNFAST_WRITE_PROTECTED

If you used the default values in the installation script, you should not need to change any of these environment variables.

You can set environment variables in the file `cknfastrc`. This file must be in the `/opt/nfast/` directory of the client.

The `cknfastrc` file should be saved without any suffix (such as `.txt`).

Each line of the file `cknfastrc` must be of the following form:

```
<variable>=<value>
```

Variables set in the environment are used in preference to those set in the resource file.

Changing the values of these variables after you start your application has no effect until you restart the application.

If the description of a variable does not explicitly state what values you can set, the values you set are normally `1` or `0`, `Y` or *N*.

For more information concerning Security World Software environment variables that are not specific to PKCS #11 and which are used to config ure the behavior of your nShield installation, see the Security World Soft ware installation instructions.

13.2.4.1. CKNFAST_ASSUME_SINGLE_PROCESS

By default, this variable is set to `1`. This specifies that only token objects that are loaded at the time `C_Initialize` is called are visible.

Setting this variable to `0` means that token objects created in one process become visible in another process when it calls `C_FindObjects`. Existing objects are also checked for modifica tion on disc; if the key file has been modified, then the key is reloaded. Calling `C_SetAttrib-uteValues` or `C_GetAttributeValues` also checks whether the object to be changed has been modified in another process and reloads it to ensure the most recent copy is changed.

Setting the variable to `0` can slow the library down because of the additional checking needed if a large number of keys are being changed and a large number of existing objects must be reloaded.

13.2.4.2. CKNFAST_ASSURANCE_LOG

This variable is used to direct all warnings from the Security Assurance Mechanism to a spe cific log file.

13.2.4.3. CKNFAST_CARDSET_HASH

This variable enables you to specify a specific card set to be used in load-sharing mode. If this variable is set, only the virtual smart card slot that matches the specified hash is present (plus the accelerator slot). The hash that you use to identify the card set in `CKN-FAST_CARDSET_HASH` is the SHA-1 hash of the secret on the card. Use the `nfkminfo` com-mand-line utility to identify this hash for the card set that you want to use: it is listed as `hkltu`. For more information about using `nfkminfo`, see *nfkminfo: information utility*.

13.2.4.4. CKNFAST_CONCATENATIONKDF_X963_COMPLIANCE

Sets the correct use of ECDH derive with concatenate KDF using the ANSI X9.63 specification as per the PKCS#11 standard.

The default is ANSI X9.63 to match that of the PKCS #11 Specification.

ECDH derive with concatenate KDF SP800-56a can use the standard PKCS #11 v3 `CKD_SHA[x]_SP800_KDF` values.

13.2.4.5. CKNFAST_DEBUG

This variable is set to enable PKCS #11 debugging. The values you can set are in the range `0` - `11`. If you are using `NFLOG_*` for debugging, you must set `CKNFAST_DEBUG` to `1`.

Value	Description
`0`	None (default setting)
`1`	Fatal error
`2`	General error
`3`	Fix-up error
`4`	Warnings
`5`	Application errors
`6`	Assumptions made by the nShield PKCS #11 library
`7`	API function calls
`8`	API return values
`9`	API function argument values
`10`	Details
`11`	Mutex locking detail

13.2.4.6. CKNFAST_DEBUGDIR

If this variable is set to the name of a writeable directory, log files are written to the specified directory. The name of each log file contains a process ID. This can make debugging easier for applications that fork a lot of child processes.

13.2.4.7. CKNFAST_DEBUGFILE

You can use this variable to write the output for CKNFAST_DEBUG (Path name > file name).

13.2.4.8. CKNFAST_DH_LSB

If this variable is set the least significant bytes of the result of DH/ECDH key agreement using the CKM_DH_PKCS_DERIVE, CKM_X9_42_DH_DERIVE or CKM_ECDH1_DERIVE mechanisms are taken. This is in line with the PKCS#11 specification. If this variable is not set the most signif icant bytes will be used. The latter behavior is consistent with Security World software prior to v12.81.

13.2.4.9. CKNFAST_FAKE_ACCELERATOR_LOGIN

If this variable is set, the nShield PKCS #11 library accepts a PIN for a module-protected key, as required by Sun Java Enterprise System (JES), but then discards it. This means that a Sun JES user requesting a certificate protected by a load-shared HSM can enter an arbi- trary PIN and obtain the certificate.

CKNFAST_FAKE_ACCELERATOR slots allow the creation of objects with CKA_PRIVATE=TRUE in the template even though the login is "fake" and the objects are not private.

- Examining the attributes shows CKA_PRIVATE as FALSE.
- A search for the object will not find it if the search criteria includes CKA_PRIVATE=TRUE.

13.2.4.10. CKNFAST_HSM_POOL

HSM Pool mode is determined by the state of the CKNFAST_HSM_POOL environment variable.

Set the environment variable to *1*, *y* or *Y* to enable HSM Pool mode for the PKCS #11 applica tion, or set to *0*, *n* or *N* to explicitly disable HSM Pool mode for the PKCS #11 application.

HSM Pool mode takes precedence over load-sharing mode. HSM Pool mode only supports module protected keys so do not use CKNFAST_NO_ACCELERATOR_SLOTS to disable the acceler ator slot.

13.2.4.11. CKNFAST_JCE_COMPATIBILITY

This property is included to allow the saving of objects when using Java PKCS#11 providers.

13.2.4.12. CKNFAST_LOADSHARING

Load-sharing mode is determined by the state of the CKNFAST_LOADSHARING environment

variable.

To enable load-sharing mode, set the environment variable `CKNFAST_LOADSHARING` to a value that starts with something other than `0`, *n*, or *N* and ensure that the `CKNFAST_HSM_POOL` environment variable is not set. The virtual slot behavior then operates.

To use softcards with PKCS #11, you must have `CKNFAST_LOADSHARING` set to a nonzero value. When using pre-loaded softcards or other objects, the PKCS #11 library automatically sets `CKNFAST_LOADSHARING=1` (load-sharing mode on) unless it has been explicitly set to `0` (load-sharing mode off).

13.2.4.13. CKNFAST_NO_ACCELERATOR_SLOTS

If this variable is set, the nShield PKCS #11 library does not create the accelerator slot, and thus the library only presents the smart card slots (real or virtual, depending on whether load-sharing is in use).

Do not set this environment variable if you want to use the accelerator slot to create or load module-protected keys.

Setting this environment variable has no effect on `ckcheckinst` because `ckcheckinst` needs to list accelerator slots.

13.2.4.14. CKNFAST_NO_SYMMETRIC

If this variable is set, the nShield PKCS #11 library does not advertise any symmetric key operations.

13.2.4.15. CKNFAST_NO_UNWRAP

If this variable is set, the nShield PKCS #11 library does not advertise the `c_wrap` and `c_unwrap` commands. You should set this variable if you are using Sun Java Enterprise System (JES) or Netscape Certificate Management Server as it ensures that a standard SSL handshake is carried out. If this variable is not set, Sun JES or Netscape Certificate Management Server make extra calls, which reduces the speed of the library.

13.2.4.16. CKNFAST_NONREMOVABLE

When this environment variable is set, the state changes of the inserted card set are ignored by the nShield PKCS #11 library.

Since protection by non-persistent cards is enforced by the HSM, not the library, this variable does not make it possible to use keys after a non-persistent card is removed, or after a timeout expires.

13.2.4.17. CKNFAST_NVRAM_KEY_STORAGE

When this environment variable is set, the PKCS #11 library generates only keys in non-volatile memory (NVRAM). You must also ensure this environment variable is set in order to delete NVRAM-stored keys.

13.2.4.18. CKNFAST_OVERRIDE_SECURITY_ASSURANCES

This variable can be assigned one or more of the following parameters, with an associated value where appropriate, to override the specified security assurances in key operations where this is deemed acceptable:

- `all`
- `none`
- `tokenkeys`
- `longterm [=<days>]`
- `explicitness`
- `import`
- `unwrap_mech`
- `unwrap_kek`
- `derive_kek`
- `derive_xor`
- `derive_concatenate`
- `unwrap_rsa_aes_kwp`
- `weak_<algorithm>`
- `shortkey_<algorithm>=<bitlength>`
- `silent`.

Each parameter specified is separated by a semicolon. Using the command line, enter the following to set the variable:

```
CKNFAST_OVERRIDE_SECURITY_ASSURANCES="<parameter1>;<parameter2>=<value3>"
```

In the configuration file, enter the following to set the variable:

```
CKNFAST_OVERRIDE_SECURITY_ASSURANCES=<parameter1>;<parameter2>=<value3>
```

Unknown parameters generate a warning; see Diagnostic warnings about questionable operations.

The meaning of these parameters is described in the rest of this section.

13.2.4.18.1. all

The `all` parameter overrides all security checks and has the same effect as supplying all the other `CKNFAST_OVERRIDE_SECURITY_ASSURANCES` parameters except the `none` parameter. Using the `all` parameter prevents the library from performing any of the security checks and allows the library to perform potentially insecure operations. This parameter cannot be used with any other parameters.

13.2.4.18.2. none

The `none` parameter does not override any of the security checks and has the same effect as supplying no parameters. Using the `none` parameter allows the library to perform all security checks and warn about potentially insecure operations without performing them. This parameter cannot be used with any other parameters.

13.2.4.18.3. tokenkeys

The `tokenkeys` parameter permits applications to request that insecure keys are stored long-term by the cryptographic hardware and library.

Some PKCS #11 applications create short-term session keys as long-term objects in the cryptographic provider, for which strong protection by the HSM is not important. Therefore, provided that you intend to create long-term keys, the need to set this token does not always indicate a potential problem because the `longterm` keys restriction is triggered automatically. If you set the `tokenkeys` parameter, ensure that your Quality Assurance process tests all of your installation's functionality at least 48 hours after the system was set up to check that the key lifetimes are as expected.

When the `tokenkeys` parameter is set, the effect on the PKCS #11 library is to permit insecure Token keys. By default, any attempts to create, generate, or unwrap insecure keys with `CKA_TOKEN=true` fails with `CKR_TEMPLATE_INCONSISTENT` and a log message that explains the insecurity. When tokenkeys is included as a parameter for `CKNFAST_OVERRIDE_SECURITY_ASSURANCES`, attempts to create, generate, or unwrap insecure keys with `CKA_TOKEN=true` are allowed.

13.2.4.18.4. longterm[=*days*]

The `longterm` parameter permits an insecure key to be used for `days` after it was created. Usually insecure keys may not be used more than 48 hours after their creation. If `days` is not specified, there is no time limit.

A need to set this variable usually means that some important keys that should be protected by the HSM's security are not secure.

When the `longterm` parameter is set, the PKCS #11 API permits the use of the following func tions with an insecure key up to the specified number of `days` after its creation:

- `C_Sign` and `C_SignUpdate`
- `C_Verify` and `C_VerifyUpdate`
- `C_Encrypt` and `C_EncryptUpdate`
- `C_Decrypt` and `C_DecryptUpdate`.

By default these functions fail with `CKR_FUNCTION_FAILED`, or `CKR_KEY_FUNCTION_NOT_PERMIT-TED`, and a log message that explains the insecurity of these functions when used with an insecure private or secret key more than 48 hours after the creation of the key as indicated by `time()` on the host.

When the `longterm` parameter is set, the functions `C_SignInit`, `C_VerifyInit`, `C_Encryp-tInit`, and `C_DecryptInit` check the `CKA_CREATION_DATE` against the current time.

13.2.4.18.5. explicitness

The `explicitness` parameter permits applications to create insecure keys without explicitly recognizing that they are insecure. An insecure key is a key that is deemed sensitive, but can be wrapped and extracted from the HSM by any untrusted key. A secure key must have the `CKA_WRAP_WITH_TRUSTED` attribute.

A need to set the `explicitness` parameter does not necessarily indicate a problem, but does usually indicate that a review of the application's security policies and use of the PKCS #11 API should be carried out.

Unless the `explicitness` parameter is set, attempts to create, generate, or unwrap insecure keys with `CKA_SENSITIVE=true`, or to set `CKA_SENSITIVE=true` on an existing key, fail by default with `CKR_TEMPLATE_INCONSISTENT` and a log message explaining the insecurity. How-ever, when the `explicitness` parameter is set, these operations are allowed.

13.2.4.18.6. import

The `import` parameter allows keys that are to be imported into the HSM's protection from insecure external sources to be treated as secure, provided that the application requests security for them. Usually, the library treats imported keys as insecure for the purposes of checking the security policy of the application. Even though the imported copy may be secure, insecure copies of the key may still exist on the host and elsewhere.

If you are migrating from software storage to hardware protection of keys, you must enable the `import` parameter at the time of migration. You can disable `import` again after migrating the keys.

Setting this variable at any other time indicates that the library regards the key as secure, even though it is not always kept within a secure envi ronment.

When the `import` parameter is set, the PKCS #11 API treats keys that are imported through `C_CreateObject` or `C_UnwrapKey` as secure (provided there is no other reason to treat them as insecure). By default, keys which are imported through `C_CreateObject` or `C_UnwrapKey` without this option in effect are marked as being insecure. Only the setting of the parameter at the time of import is relevant.

13.2.4.18.7. unwrap_mech

The `unwrap_mech` parameter allows you to create keys with `CKA_UNWRAP=true` and `CKA_DECRYPT=false`.

By default, when `unwrap_mech` is not supplied as a parameter for `CKNFAST_OVERRIDE_SECURITY_ASSURANCES`, trying to create a key with `CKA_UNWRAP=true` and `CKA_DECRYPT=false` fails with `CKR_TEMPLATE_INCONSISTENT`.

When `CKA_UNWRAP` is set to `true` and `CKA_DECRYPT` is not specified in the template, then `CKA_DECRYPT` is automatically set to `true`.

13.2.4.18.8. unwrap_kek

When a key is transferred into the HSM in encrypted form, the key is usually treated as insecure unless the key that was used for the decryption only allows the import and export of keys and not the decryption of arbitrary messages. This behavior is necessary to prevent an unauthorized application from simply decrypting the encrypted key instead of importing it. However, because PKCS #11 wrapping mechanisms are insecure, all unwrapping keys have `CKA_DECRYPT=true`.

By default, keys that are unwrapped with a key that has `CKA_DECRYPT` permission are consid-

ered insecure. When the unwrap_kek parameter is set, the PKCS #11 API considers keys that are unwrapped with a key that also has CKA_DECRYPT permission as secure (provided there is no other reason to treat them as insecure).

13.2.4.18.9. derive_kek

By default, keys that have been derived by using CKM_DES3_ECB_ENCRYPT_DATA with a key that has CKA_ENCRYPT permission are considered insecure. However, when the derive_kek parameter is set, the PKCS #11 API considers keys that are derived with a key that has CKA_ENCRYPT permission as secure (provided that there is no other reason to treat them as insecure).

13.2.4.18.10. derive_xor

Normally, you can only use only extractable keys with CKM_XOR_BASE_AND_DATA and, on unextractable keys, only CKM_DES3_ECB_ENCRYPT_DATA is allowed by CKA_DERIVE. However, when the derive_xor parameter is set, the PKCS #11 API also allows such functions with keys that are not extractable and treats them as secure (provided that there is no other reason to treat them as insecure).

13.2.4.18.11. derive_concatenate

Normally, you can only use session keys with CKM_CONCATENATE_BASE_AND_KEY for use with the operation C_DeriveKey. However, when the derive_concatenate parameter is set, the PKCS #11 API also allows such functions with keys that are long term (token) keys. The PKCS #11 API treats these keys as secure, provided there is no other reason to treat them as insecure. Even if the all parameter is set, if you do not include the CKA_ALLOWED_MECHANISMS with CKM_CONCATENATE_BASE_AND_KEY, this C_DeriveKey operation will not be allowed.

13.2.4.18.12. unwrap_rsa_aes_kwp

The C_UnwrapKey operation with CKM_RSA_AES_KEY_WRAP imports the temporary AES key with an nCore API ACL that permits unwrapping of the wrapped target key by the temporary AES key. When using the C_UnwrapKey operation with only a user supplied template (pTemplate) it is possible to create this ACL such that it permits a one-time unwrap of only the wrapped target key. When the RSA unwrapping key has CKA_UNWRAP_TEMPLATE set it is necessary to construct the ACL when the RSA key is created in order to setup the partitioning guarantees from the CKA_UNWRAP_TEMPLATE. The intended wrapped target keys are unknown at this time, which means the ACL must permit a one-time unwrap of any key.

The Security Assurance Mechanism (SAM) considers this scenario insecure by default and

therefore the use of the C_UnwrapKey operation with CKM_RSA_AES_KEY_WRAP is disabled when the RSA unwrapping key has CKA_UNWRAP_TEMPLATE set. When the unwrap_rsa_aes_kwp parameter is set the SAM enables the C_UnwrapKey operation with CKM_RSA_AES_KEY_WRAP in this scenario. The RSA unwrapping key must also explicitly allow the CKM_RSA_AES_KEY_WRAP mechanism via CKA_ALLOWED_MECHANISMS in addition to setting the unwrap_rsa_aes_kwp (or all) parameter; otherwise, the C_UnwrapKey operation will remain disabled when the RSA unwrapping key has CKA_UNWRAP_TEMPLATE set.

13.2.4.18.13. weak_<algorithm>

The weak_<algorithm> parameter allows you to treat keys used with a weak algorithm as secure. For example, DES is not secure, but setting the parameter weak_des means that such keys are considered secure. You can apply the weak_<algorithm> parameter to all keys that have a short fixed key length or whose algorithms have other security problems. As a guide, weak algorithms are those whose work factor to break is less than approximately 80 bits.

13.2.4.18.14. shortkey_<algorithm=bitlength>

The shortkey_<algorithm=bitlength> parameter permits excessively short keys for the specified <algorithm> to be treated as secure. The parameter <bitlength> specifies the min imum length, in bits, that is to be considered secure. For example, RSA keys must usually be at least 1024 bits long in order to be treated as secure, but shortkey_rsa=768 would allow 768-bit RSA keys to be treated as secure.

13.2.4.18.15. silent

The silent parameter turns off the warning output. Checks are still performed and still return failures correctly according to the other variables that are set.

13.2.4.18.16. Diagnostic warnings about questionable operations

When the CKNFAST_OVERRIDE_SECURITY_ASSURANCES environment variable is set to a value other than all, diagnostic messages are always generated for questionable operations. Each message contains the following elements:

- The PKCS #11 label of the key, if available
- The PKCS #11 identifier of the key, if available
- The hash of the key
- A summary of the problem.

If the problem is not that a questionable operation has been permitted because of a setting in CKNFAST_OVERRIDE_SECURITY_ASSURANCES it could be that an operation has failed. In such a case, the setting required to authorize the operation is noted.

By default, these messages are sent to stderr. If a file name has been specified in the CKNFAST_ASSURANCE_LOG environment variable, diagnostic messages are also written to this file.

If CKNFAST_DEBUG is 1 or greater and a file is specified in CKNFAST_DEBUGFILE, the PCKS #11 library Security Assurance Mechanism log information is sent to the specified file. These variables must be set whenever generatekey or KeySafe are used.

If a file is specified in CKNFAST_ASSURANCES_LOG and no file is specified in CKNFAST_DEBUGFILE (or if CKNFAST_DEBUG is 0), diagnostic messages are sent to stderr as well as to the file specified in CKNFAST_ASSURANCES_LOG.

13.2.4.19. CKNFAST_SEED_MAC_ZERO

Set this variable to use zero padding for the Korean SEED MAC mechanisms (CK_SEED_MAC and CKM_SEED_MAC_GENERAL). If this variable is not set, or is set to n, then the SEED MAC mechanisms will use the default PKCS #5 padding scheme.

13.2.4.20. CKNFAST_SESSION_THREADSAFE

You must set this environment variable to yes if you are using the Sun PKCS #11 provider when running nCipherKM JCA/JCE code.

13.2.4.21. CKNFAST_TOKENS_PERSISTENT

This variable controls whether or not the Operator Cards that are created by your PKCS #11 application are persistent. If this variable is set when your application calls the PKCS #11 function that creates tokens, the Operator Card created is persistent.

Use of the nShield PKCS #11 library to create tokens is deprecated, because it can only create 1/1 tokens in FIPS 140 Level 2 Security Worlds. Use KeySafe or one of the command-line utilities to create OCSs.

13.2.4.22. CKNFAST_USE_THREAD_UPCALLS

If this variable is set and CKF_OS_LOCKING_OK is passed to C_Initialize, NFastApp_SetThreadUpcalls is called by means of nfast_usencthreads and only a single NFastApp_Connection is

used, shared between all threads.

If this variable is set and mutex callbacks are passed to `C_Initialize` but `CKF_OS_LOCKING_OK` is not passed, `C_Initialize` fails with `CKR_FUNCTION_FAILED`. (`NFastApp_SetThreadUpcalls` requires more callbacks than just the mutex ones that PKCS #11 supports.)

If neither mutex callbacks nor `CKF_OS_LOCKING_OK` is passed, this variable is ignored. Only a single connection is used because the application must be single threaded in this case.

13.2.4.23. CKNFAST_LOAD_KEYS

This variable will load private objects at `C_Login` time, rather than at the first cryptographic operation.

13.2.4.24. CKNFAST_WRITE_PROTECTED

Set this variable to make your OCS or softcard (token) write-protected. If a token is write-protected, you cannot:

- Generate certificate, data, and key objects for that token.
- Modify attributes of an existing object.

This environment variable does not prevent you from deleting an object from your token.

13.2.4.25. CKNFAST_RELOAD_KEYS

Set this variable to enable PKCS #11 key reloading. See section *PKCS _11 with key reloading* in the *Cryptographic API Integration Guide*.

Key reloading requires load sharing-mode to operate, and enables it automatically if `CKNFAST_LOADSHARING` is not set.

13.2.5. Checking the installation of the nShield PKCS #11 library

After you have created a Security World, ensure that the nShield PKCS #11 library has been successfully installed by using the `ckcheckinst` command-line utility.

To verify the installation of the nShield PKCS #11 library, follow these steps:

1. Give the command `ckcheckinst`.

   If you have an invalid Security World (for example, if all your HSMs are in the initializa-

tion state), `ckcheckinst` quits with the following error message:

```
ckcheckinst: C_Initialize failed rv = 00000006
Is the security world initialized? (Use nfkminfo to check)
```

If your Security World is valid, `ckcheckinst` displays information similar to the following:

```
PKCS#11 library interface version 2.40
 flags 0
 manufacturerID "nCipher Corp. Ltd "
 libraryDescription "nCipher PKCS#11 1.#.# "
 implementation version 1.##
 Load sharing and Failover enabled

slot Status Label
===== ====== ===== 0 Fixed token "accelerator "
1 Operator card "card2 "
2 Operator card "card3 "
Select slot Number to run library test or 'R'etry or to 'E'xit:
```

In this example output:

- `PKCS #11 library interface version 2.40` refers to the version of the PKCS #11 specification supported
- `implementation version 1.##` refers to the version of the nCipher PKCS #11 library
- `Loadsharing and Failover enabled` is shown if load-sharing has been enabled. Alternatively `Pool mode enabled` is shown if Pool mode has been enabled.

Slots that contain a valid Operator Card are indicated by the status `Operator card` and the card's label. A fixed token is always available and is listed as slot 0.

If you insert a blank card or an unrecognized card (for example, an Operator Card from a different Security World or an Administrator Card), this is indicated in the `Status` column. The corresponding slot number is not available.

If you are using the `preload` command-line utility in conjunction with the nShield PKCS #11 library, you can only see the token that you loaded with the `preload` utility. In load-sharing mode, the loaded card set is used to set the environment variable `CKNFAST_CARDSET_HASH`, so only this card set is visible as a slot.

If there is no card in a slot, `ckcheckinst` displays `No token present` beside the relevant slot numbers.

`ckcheckinst` gives you the following choices:

```
No removable tokens present.
Please insert an operator card into at least one available slot and
enter 'R' retry.
If you have not created an operator card or there are no physical slots, enter a fixed token slot number,
or 'E' to exit this program and create a card set before continuing.
```

2. If there are no available slots with cards in them, you can choose one of the following actions:
   - Insert a valid Operator Card, and press **R**
   - choose a fixed token slot
   - Press **E** to quit, then create an OCS, and run `ckcheckinst` again.

   When there is at least one slot with a valid token, input a slot number, and press **Enter**. In a FIPS 140 Level 3 compliant Security World, `ckcheckinst` prompts you to enter the passphrase for the selected Operator Card.
3. Type the passphrase, and press **Enter**.

   `ckcheckinst` displays the results of the tests:

   ```
   Test Pass/Failed
   ---- -----------
   1 Generate RSA key pair Pass
   2 Generate DSA key pair Pass
   3 Encryption/Decryption Pass
   4 Signing/Verify Pass
   Deleted test keys ok
   PKCS11 Library test successful.
   ```

   If any tests fail, `ckcheckinst` displays a message indicating the failure and quits. It does not run any subsequent tests.

   If `ckcheckinst` fails:
   - Check that the hardserver is running
   - Use the `enquiry` and `nfkminfo` world.

   If all seems in order, reinstall the nShield library.

13.2.6. How the nShield PKCS #11 library protects keys

Session objects are created on an HSM and never leave that HSM. The following table lists the protection for different types of PKCS #11 token objects:

	Smart card Slot	Accelerator Slot
Private Token Object	Operator Card Set	not supported

	Smart card Slot	Accelerator Slot
Public Token Object	Security World	Security World
Public key	well known HSM key	well known HSM key

`Operator Card Set`

The object is stored as an nShield key blob encrypted by the OCS key. You must log in to this OCS before you can load this object.

`security world`

The object is stored as an nShield key blob encrypted by the Security World key. This object can be loaded on to any HSM in the Security World. The nShield PKCS #11 library only allows access if a card from this OCS is present.

`well-known module key`

Public keys are encrypted under a well-known HSM key. This encryption is for programming convenience only and does not provide security. These keys can be loaded on any nShield HSM.

13.3. nShield native and custom applications

Use the nShield native option for applications that were written using nShield key management software and that expect keys to be both protected by the Security World and stored in the Security World data structure.

Use the `custom` external application option for applications that were written using nShield key management software and that expect their keys to be in standalone files.

KeySafe does not place any restrictions on the OCS that is used to protect nShield native or `custom` application keys. You must make sure that your application is capable of loading the card set.

13.4. CodeSafe applications

If you have enabled the Secure Execution Engine (SEE), your system can run CodeSafe applications that implement special functionality.

If you wish to use the SEE to run applications, it must have been ordered and enabled as described in Enabling optional features.

An SEE application is typically a standalone SEE machine that is loaded automatically by the hardserver (for example, a CodeSafe C application).

Check the documentation that your application vendor supplies for information about any signatures that you may require to set up and use the application, as well as for any other installation and configuration information.

CodeSafe applications are standalone applications, but each CodeSafe C application can consist of multiple parts, and its installation can include several configuration steps. For instructions on installing and configuring each application, see your application vendor's documentation.

You may need to use the `hardserver`, `loadmache`, and `tct2` utilities when configuring and loading an application; see the *CodeSafe Developer Guide* for more information.

To use `see-sock-serv` directly, you must select `BSDlib sockserv`.

14. Remote Operator

This chapter explains:

- The concept of Remote Operator
- How to configure Remote Operator.

If you wish to use the Remote Operator feature, you must have enabled it as described in Enabling optional features. The Remote Operator feature must have been ordered for, and enabled on, the nShield module that you intend to use as the remote, unattended module.

14.1. About Remote Operator

The Remote Operator feature enables the contents of a smart card inserted into the slot of one module (the *attended module*) to be securely transmitted and loaded onto another module (an *unattended module*). This is useful when you need to load an OCS-protected key onto a machine to which you do not have physical access (because, for example, it is in a secure area).

For Remote Operator to work, the modules must be in the same Security World. You insert the required cards from the OCS into a slot in the attended module. From this module, the contents of the OCS are transmitted over secure channels to the unattended module, which then loads them. You do not need physical access to the unattended module in order to load the OCS onto it.

The following limitations apply to Remote Operator:

- You cannot access non-persistent card sets remotely
- You cannot use the `createocs` command-line utility to write new cards or card sets remotely.

You can export a slot from an attended module and import a slot to any (unattended) module in the Security World. Before you can import a slot to one module, you must first export it from another module.

14.2. Configuring Remote Operator

This section explains how to configure Remote Operator.

14.2.1. Overview of configuring Remote Operator

Before you can use Remote Operator, you must perform the following initial configuration tasks:

1. Configure the HSMs for Remote Operator.

   The HSMs must be in the same Security World, and must have been initialized with remote card set reading enabled.

   Both the attended and the unattended HSM must be in operational mode before they can import or export slots. * The HSM must be in pre-initialization mode. See Checking and changing the mode on an nShield 5s module for more about changing the mode.

2. Configure the HSM hardservers on their respective host machines for slot import and export, as appropriate.

Starting from 12.81, you can export and import dynamic slots as Remote Operator slots.

After the initial configuration is complete, to use Remote Operator you must:

1. Create a Remote OCS (that is, an OCS with the correct permissions for Remote Opera tor).
2. Generate keys that are protected by the Remote OCS.
3. Ensure your application is configured to use keys protected by the Remote OCS.

14.2.2. Configuring HSMs for Remote Operator

1. Ensure both HSMs are initialized into the same Security World; see Adding or restoring an HSM to the Security World.

   By default, HSMs are initialized with remote card-set reading enabled. If you do not want an HSM to be able to read remote card sets, you can initialize it by running the `new-world` with the `-S` *MOD ULE* (where *MODULE* is the HSM's ID number).

2. For the unattended HSM:
   a. Check whether the Remote Operator feature is enabled by running the `enquiry` command-line utility. The output for the HSM must include `Remote Share` in its list of Features.
   b. Check whether the correct software, with permission to receive remote shares, is present by running the `nfkminfo` command-line utility.

The output from this selection must show that `flags` are set to include **ShareTarget**, as in the following example:

```
Module #1
generation 2
state 0x2 Usable
flags 0x10000 ShareTarget
n_slots 3
esn 8851-43DF-3795
hkml 391eb12cf98c112094c1d3ca06c54bfe3c07a103
```

14.2.3. Configuring slot import and export

For information about the parameters controlled by the hardserver configuration file, see:

- slot_exports
- slot_imports
- slot_mapping

Before you can configure hardservers for Remote Operator, ensure that:

- You have configured the attended and unattended HSMs for Remote Operator as described in Configuring HSMs for Remote Operator.
- Your network firewall settings are correct. See the *Installation Guide* for more information about firewall settings.

When the HSMs have been configured, use one of the following methods to configure slot import and export:

- Use the `cfg-remoteslots` utility.
- Update the HSM configuration file, see Configuring hardservers for Remote Operator using the HSM configuration file.

14.2.3.1. Configuring hardservers for Remote Operator using the HSM configuration file

1. On the attended HSM's host machine, configure the hardserver to allow slot 0 of the local HSM (with ESN AAAA-AAAA-AAAA) to be exported to a remote HSM (with ESN BBBB-BBBB-BBBB, hosted by the machine with the IP address 222.222.222.222):
   a. Edit the `slot_exports` section of the hardserver configuration file by adding lines of the form:

```
local_esn=AAAA-AAAA-AAAA
local_slotid=0
remote_ip=222.222.222.222
```

```
remote_esn=BBBB-BBBB-BBBB
```

b. Run the `cfg-reread` command-line utility to prompt the hardserver to read the con figuration changes.

2. On the unattended module's host machine, configure the hardserver to import slot 0 from the remote attended module (with ESN *AAAA-AAAA-AAAA*, hosted by the machine with the IP address *111.111.111.111*) to the local module (with ESN *BBBB-BBB-BBBB*).

   a. Edit the `slot_imports` section of the hardserver configuration file by adding lines of the form:

   ```
   local_esn=BBBB-BBBB-BBBB
   local_slotid=2
   remote_ip=111.111.111.111
   remote_esn=AAAA-AAAA-AAAA
   remote_slotid=0
   ```

   This example assigns the imported slot to ID `2`.

3. Check the Remote Operator slot configuration:

   ```
   slotinfo -m 1
   ```

   If slot import was successful, the output from this command includes the line:

   ```
   Slot Type           Token   IC    Flags   Details
   #0   Smartcard      present 3     A
   #1   Software Tkn   -       0
   #2   Smartcard      -       0     AR
   ```

   The `R` in the `Flags` column indicates that slot `2` is a Remote Operator slot.

Applications running on the unattended machine can now use slot `2` to load OCSs that are presented to slot `0` on the attended machine. If any of the cards require a passphrase, the application must pass this to the unattended HSM in the usual way.

For the application to be able to load the OCS onto the unattended HSM, it must be able to read the card set files associated with the OCS from the local Key Management Data directory. If the OCS was created on a different machine, you must copy the card set files in the Key Management Data directory onto the unattended machine (either manually or by using client cooperation; for more information, see Setting up client cooperation).

The same applies for any keys that an application on an unattended HSM needs to load but that were not generated on that machine.

14.2.4. Using Remote Operator with applications requiring cards in slot 0

If you want to use Remote Operator, but have an application that expects cards to be presented in slot 0, you must configure a slot mapping for each affected HSM.

1. Do the following:
   a. Use the `slot_imports` section in the hardserver configuration file to import remote slots from HSMs in the same Security World for each relevant HSM.
   b. Use the slot_mapping section in the hardserver configuration file to define the remote slot which is to be swapped with slot #0 for each relevant HSM.

You can check the mapping by:

- Running the command:

  ```
  slotinfo -m 1
  ```

  For example, if remote slot #2 has been mapped to slot #0, the output from this command includes the lines:

  ```
  Slot Type Token IC Flags Details
  #0 Smartcard - 1 AR
  #1 Software Tkn - 0
  #2 Smartcard - 0 A
  ```

- The `R` in the `Flags` column indicates that slot `#0` is now a Remote Slot

Slot mapping can also be configured for a dynamic remote slot, i.e. a dynamic slot in a different HSM which has been imported to the relevant HSM. The Flags column will contain the flags ARD.

When dynamic slots are added to an HSM after the initial configuration was done with only remote slots, the dynamic slots will take precedence over the remote slots. The slot numbers of the remote slots will therefore change. You will have to revise the slot mapping and specify the new slot number of the remote slot.

14.2.5. Using Remote Operator on Remapped Slots

If a slot has been mapped to slot #0 on the attended HSM, it is still possible to export the local slot to an unattended HSM. Further, if the mapped slot is a dynamic slot, it is possible to export it as well. To do this, do the following:

1. On the attended HSM's host machine, configure the hardserver to allow the export of the relevant slot by refering to it by its original slotID.
   a. To export the local slot, local_slotid=0.
   b. To export a dynamic slot, local_slotid=2 (or higher if the HSM is configured with multiple dynamic slots).
2. On the unattended HSM's host machine, configure the hardserver to import the relevant slot by refering to it by its new slotID.
   a. To import the exported local slot, remote_slotid=2 (or higher, same as the slotID specified in the mapping section of the attended HSM's configuration file).
   b. To import the exported dynamic slot, remote_slotid=0.

14.2.6. Configuration Example for Using Remote Administration and Remote Operator Concurrently

Below is an example of the relevant portions of a hardserver config file to achieve concurrent usage of Remote Administration and Remote Operator. It is broken up and explained per config file section.

The `dynamic_slots` section allocates exactly 1 dynamic slot to each of modules 1 and 2.

```
[dynamic_slots]
esn=BBBB-BBBB-BBBB
slotcount=1
-----
esn=AAAA-AAAA-AAAA
slotcount=1
```

The `slot_imports` section first imports module 1 slot #0 to module 2 slot #3 and then imports module 1 slot #2 to module 2 slot #4.

```
[slot_imports]
local_esn=AAAA-AAAA-AAAA
remote_ip=127.0.0.1
remote_port=9004
remote_esn=BBBB-BBBB-BBBB
remote_slotid=0
-----
local_esn=AAAA-AAAA-AAAA
remote_ip=127.0.0.1
remote_port=9004
remote_esn=BBBB-BBBB-BBBB
remote_slotid=2
```

The `slot_exports` section allows module 1 slot #0 and module 1 slot #2 to be exported by that module.

```
[slot_exports]
local_esn=BBBB-BBBB-BBBB
local_slotid=0
-----
local_esn=BBBB-BBBB-BBBB
local_slotid=2
```

The `slot_mapping` section swaps module 2 slot #0 and module 2 slot #2.

```
[slot_mapping]
esn=AAAA-AAAA-AAAA
slot=2
```

After making the changes above to the hardserver configuration file:

1. Re-read the hardserver configuration file by running `cfg-reread`.
2. Clear the modules by running `nopclearfail`.

This is the expected system configuration output for the relevant modules:

```
slotinfo -m1
Slot Type             Token    IC     Flags    Details
#0   Smartcard        -        0      A
#1   Software Tkn     -        0
#2   Smartcard        -        0      AD

slotinfo -m2
Slot Type             Token    IC     Flags    Details
#0   Smartcard        -        0      AD
#1   Software Tkn     -        0
#2   Smartcard        -        0      A
#3   Smartcard        -        0      AR
#4   Smartcard        -        0      ARD
```

14.2.7. Using Remote Operator with Remote Administration with Older Versions of the Software

Versions of Remote Operator older than 12.81 do not support its concurrent use with the Remote Administrator feature. In such a case, the following features are not supported:

- Exporting and importing dynamic slots
- Mapping remote slots to slot #0
- Automatic assignment of slotID when importing slots

It is possible to use some of the features when the attended HSM (exporting end) has the new version of the software (12.81+) and the unattended HSM (importing end) has an older version (pre-12.81).

A dynamic slot which has been exported by the attended HSM can be imported to the unat

tended HSM. Its local slotID will need to be manually specified if the unattended HSM has any dynamic slots configured. This is due to the default import slot (slot #2) being occupied by the dynamic slot. The unattended HSM can remap its dynamic slots to slot #0, but cannot remap any of its imported slots.

14.3. Creating OCSs and keys for Remote Operator

When you have configured the HSMs and hardservers for Remote Operator, you can create Remote OCSs and generate keys protected by them. These Remote OCSs and keys can be used by applications running on the unattended HSM.

For the most part, card sets and keys intended to be used with Remote Operator are similar to their ordinary, non-Remote counterparts.

14.3.1. Creating OCSs for use with Remote Operator

You can generate Remote OCSs by using KeySafe or by running the `createocs` command-line utility with the `-q|--remotely_readable` option specified. The cards in a Remote OCS must be created as persistent; see Persistent Operator Card Sets.

To check whether the card in a slot is from a Remote OCS, run the `nfkminfo` command-line utility. The output displays slot section information similar to the following:

```
Module #1 Slot #0 IC 1
generation         1
phystype           SmartCard
slotlistflags      0x2
state              0x5
Operator flags     0x20000 RemoteEnabled
shareno            1
shares             LTU(Remote)
error              OK
```

In this example output, the `RemoteEnabled` flag indicates the card in the slot is from a Remote OCS.

If you create a Remote OCS on the attended machine, then you must copy the Key Management Data files on the attended machine to the unattended machine.

Both the attended and unattended HSMs must be in the same Security World before you generate a Remote OCS. If you are not using client cooperation, the Key Management Data directories must be manually synchronized after you generate the Remote OCS.

If you already have recoverable keys protected by a non-Remote OCS, you can transfer them to a new Remote OCS by using KeySafe or the `replaceocs` command-line utility.

14.3.2. Loading Remote Operator Card Sets

Once configured, the Remote Operator slots can be used by all the standard nShield libraries. A Remote Operator slot can be used to load any OCSs that have been created to allow remote loading. For more information about the applications to use with remote cards, see Application interfaces. For more information about Remote Operator slots, see Remote Operator.

After an OCS has been inserted into a Remote Operator slot, for each time a given card is inserted, the module only allows each share on that card to be read one time. If there is a second attempt to read shares from that card before the card is reinserted, the operation fails with a `UseLimitsUnavailable` error.

14.3.3. Generating keys for use with Remote Operator

After you have created a Remote OCS, to generate keys protected by it you can run KeySafe or the `generatekey` and `preload` command-line utilities on the unattended module, inserting cards to the slot attached to the attended module. For more information about generating and working with keys, see Working with keys.

If you generate keys protected by a Remote OCS on the attended module, then you must copy the files in the Key Management Data directory on the attended machine to the unattended module.

KeySafe can list imported slots, but cannot use them.

If you already have an OCS-protected key that you want to use, but the protecting OCS is not a Remote OCS, you can use KeySafe to protect the key under a new Remote OCS if the key was originally generated with the key recovery option enabled.

However, if the key was not generated with key recovery enabled, you cannot protect it under a different OCS. In such a case, you must generate a new key to be protected by a Remote OCS.

14.3.4. Configuring the application

After you have configured the HSMs and hardservers for Remote Operator, created a Remote OCS, and generated keys protected by the Remote OCS, configure the application with which you want to use these keys as appropriate for the particular application.

After you have configured the application, start it remotely from the attended machine. Insert cards from the OCS into the attended machine's exported slot as prompted.

15. Working with keys

This chapter explains how to use the facilities we provide to work with keys. There is often more than one way of performing a particular task. The methods available for working with keys are:

- KeySafe
- `generatekey` and related utilities

15.1. Generating keys

Whenever possible, generate a new key instead of importing an existing key. Because existing keys have been stored in a known format on your hard disk, there is a risk that the existing key has been compromised. Key material can also persist on backup media.

Some applications can generate keys directly.

When you attempt to generate keys for a Security World that complies with FIPS 140 Level 3, you are prompted to insert an Administrator Card or Operator Card. You may need to specify to the application, the slot you are going to use to insert the card. You need to insert the card only once in a session.

For softcard protected key generation, you must use an Operator Card Set.

Generating a key creates both a key and a certificate request for the following application types:

- `embed` (OpenSSL)
- `kpm`

These requests are generated in PKCS #10 format with base-64 encoding.

15.1.1. Generating keys using the command line

Keys are generated using the command line with the `generatekey` utility. The `--generate` option creates a new key on the host computer that is protected either by the module or by an Operator Card set from the Security World. No key material is stored in an unencrypted form on the host computer.

When you generate a key with `generatekey`, choose a new identifier for the key and use whichever application type is appropriate. The key identifier can only contain digits, lower-

case ASCII letters, and hyphens (-).

Any uppercase letters you enter in the key identifier are converted to lowercase when the key is generated.

You can use `generatekey` in two ways:

- In interactive mode, by issuing commands without parameters and supplying the required information when prompted by the utility
- In batch mode, by supplying some or all of the required parameters using the command line (`generatekey` prompts interactively for any missing but required parameters).

In interactive mode, you can input `abort` at any prompt to terminate the process.

Batch mode is useful for scripting. In batch mode, if any required parameters are omitted, `generatekey` does not prompt for the missing information but instead will either use available defaults or fail. If you specify one or more parameters incorrectly, an error is displayed and the command fails.

If the Security World was created with audit logging selected then you can request that the usage of a key for cryptographic operations is logged in the audit log. By default only key generation and destruction is logged. For further information see Audit Logging.

To generate a key, use the command:

```
generatekey --generate [OPTIONS] <APPNAME> [<NAME>=<VALUE> ...]
```

In this command:

- `--generate` option specifies that this instance of `generatekey` is generating a key. Other options can be specified to perform tasks such as importing or retargeting keys. To see a list of options run the command `generatekey --help`.
- the `<APPNAME>` parameter specifies the name of the application for which the key is to be generated. For details of the available application types (`APPNAME`), Key application type (APPNAME).
- The `<NAME>=<VALUE>` syntax is used to specify the properties of the key being generated. For details of the available application types (`APPNAME`), see Key properties (NAME=VALUE).

For details of the available application types (`APPNAME`) and parameters that control other key properties (`NAME=VALUE`), see Key generation options and parameters and parameters.

In interactive mode, `generatekey` prompts you for any required parameters or actions that have not been included in the command. When you give the command:

1. Enter parameters for the command, as requested. If you enter a parameter incorrectly, the request for that information is repeated and you can re-enter the parameter.
2. When all the parameters have been collected, `generatekey` displays the final settings. In a FIPS 140 Level 3 compliant Security World, you are prompted to insert a card for FIPS authorization if no such card is present.
3. If prompted, insert an Administrator Card or an Operator Card from the current Security World.
4. If you want to protect the key with an OCS, you are prompted to insert the relevant cards and input passphrases, as required.

15.1.1.1. Example of key generation with generatekey

To generate a simple RSA key in batch mode, protected by module protection, use the command:

```
generatekey --generate --batch simple type=rsa size=2048 plainname=keya ident=abcd certreq=yes
```

The `generatekey` utility prompts you to insert a quorum of Operator Cards from the `operatorone` OCS. After you have inserted the appropriate number of cards, `generatekey` generates the key.

Although it is not explicitly specified, the created key is recoverable by default if OCS and softcard replacement is enabled for the Security World.

15.1.2. Generating keys with KeySafe

In order to generate a key with KeySafe, follow these steps:

1. Start KeySafe. (For an introduction to KeySafe and for information on starting the software, see Using KeySafe.)
2. Click the **Keys** menu button, or select **Keys** from the **Manage** menu. KeySafe takes you to the **Keys** panel, which shows the keys in the Security World.
3. Click the **Create** button to open the **Generate Key** panel.
4. Select an application with which you want to use your key from the list, and then click the **Next** button. KeySafe takes you to the **Key Generation Parameters** panel.
5. Select and enter your desired parameters for key generation.

   The types of information that you need to provide on the **Key Generation Parameters** panel differs slightly depending on the application you selected on the **Generate Key** panel.

6. When you have supplied your desired key generation parameters, click the **Commit** button.

In order to generate a key protected by a FIPS 140 Level 3 compliant Security World, you need authorization from an Operator Card or Administrator Card from the current Security World. Follow the onscreen instructions.

7. If you choose to generate a key that is protected by a smart card or softcard, KeySafe takes you to a panel from which you can load the protecting card or softcard. Follow the onscreen instructions, inserting any necessary Operator Cards and supplying any passphrases as needed.
8. KeySafe displays a message indicating that the key has been successfully generated. Click the **OK** button.
9. KeySafe returns you to the **Generate Key** panel, from which you can generate another key or choose another operation.

15.1.3. Generating NVRAM-stored keys

NVRAM key storage provides a mechanism for generating keys stored in a module's non-volatile memory and hence within the physical boundary of an nShield module. You can store only a few keys in this way: the number depends on the memory capacity of the module, the size of the key and whether the key has recovery data associated with it.

We recommend that you *do not store keys in NVRAM unless you must do so to satisfy regulatory requirements.* NVRAM key storage was introduced only for users who must store keys within the physical boundary of a module to comply with regulatory requirements. NVRAM-stored keys provide no additional security benefits and their use exposes your ACS to increased risk. Storing keys in nonvolatile memory also reduces load-balancing and recovery capabilities. Because of these factors, we recommend you always use standard Security World keys unless explicitly required to use NVRAM-stored keys.

When you generate an NVRAM-stored key, you must have sufficient nonvolatile memory available in the module or the command fails.

You need backup and recovery procedures, which must be consistent with regulatory requirements, to protect your NVRAM-stored keys. Do *NOT* use Remote Administration to back-up keys to a smart card, as, in transit, the keys would not be physically protected from access by the

host system.

An NVRAM-stored key can only be loaded successfully by using the `preload` command-line utility on the generating module. Attempts to load such a key on other modules that have NVRAM fail with `UnknownID` errors.

We provide the `nvram-backup` utility to enable the copying of files, including NVRAM-stored keys, between a module's nonvolatile memory and a smart card.

15.2. Importing keys

Importing a key takes an unprotected key stored on the host and stores it in the Security World in encrypted form.

We recommend generating a new key (or retargeting a key from within the Security World) instead of importing an existing key whenever possible. The import operation does not delete any copies of the key material from the host, and because existing keys have been stored in a known format on your hard disk (and key material can persist on backup media), there is a risk that an existing key has been compromised. It is your responsibility to ensure any unprotected key material is deleted. If a key was compromised before importation, then importing it does not make it secure again.

The following key types can be imported by the tools we provide:

- RSA keys in PEM-encoded PKCS #1 format (from a file). The PEM key that contains the key to import must not require a passphrase.
- DES, DES2 and Triple DES keys (entered in hex).

You cannot import keys into a Security World that complies with FIPS 140 Level 3. Attempting to import keys into a FIPS 140 Level 3 Security World returns an error.

This request is a PKCS #10 format request in base-64 encoding.

15.2.1. Importing keys from the command line

You can import keys using the `generatekey` utility. To import a key, give the command:

```
generatekey --import [<OPTIONS>] <APPNAME> [<NAME>=<VALUE> ...]
```

This command uses the following options:

Option	Description
`--import`	This option specifies key importation.
`<OPTIONS>`	You can specify particular options when running `generatekey` that control details of key importation.
`<APPNAME>`	This option specifies the name of the application for which the key is to be imported. This must be an application for which `generatekey` can generate keys.
`<NAME>=<VALUE>`	This specifies a list of parameters for the application.

For RSA keys, you can include `pemreadfile=filename` in the command to specify the file name of the PEM file that contains the key. Otherwise, you are prompted for this information during import.

In interactive mode, you are prompted for any required parameters or actions that have not been included in the command:

- Enter parameters, as requested. If you enter a parameter incorrectly, the request for that information is repeated and you can re-enter the parameter.
- If you want to protect the key with an OCS, you are prompted to insert the relevant cards and input passphrases, as required.
- If prompted, insert an Administrator Card or an Operator Card from the current Security World.

15.2.1.1. Example of key importation with generatekey

To import an RSA key stored in `/opt/projects/key.pem` for use with an nShield native application and protect it with the Security World, use the command:

```
generatekey --generatekey --import simple pemreadfile=/opt/projects/key.pem plainname=importedkey ident=abc
protect=module
```

In this example, `generatekey` requires you to input `RSA` for the key type.

Although not explicitly specified, this key is, by default, recoverable if OCS and softcard replacement is enabled for the Security World.

15.2.2. Importing keys with KeySafe

Any user who has write access to the directory that contains the Security World can import a key.

In order to import a key with KeySafe, follow these steps:

1. Start KeySafe. (For an introduction to KeySafe and for information on starting the software, see Using KeySafe.)
2. Click the **Keys** menu button, or select **Keys** from the **Manage** menu. KeySafe takes you to the **Keys** panel.
3. Click **Import** to open the **Import Key** panel.
4. Select the application associated with the key that you want to import, and then click the **Next** button. KeySafe takes you to the **Key Import Parameters** panel.
5. Select and enter the desired parameters for the key that you want to import.

   The types of information that you need to provide on the **Key Import Parameters** panel will differ slightly depending on the application you selected on the **Import Key** panel.

6. When you have supplied parameters for the key that you want to import, click the **Commit** button.
7. If you choose to import a key that is protected by a smart card, KeySafe takes you to the **Load Operator Card Set** panel. Follow the onscreen instructions, inserting the required number of Operator Cards and supplying any passphrases as needed.
8. KeySafe displays a message indicating that the key has been successfully imported. Click the **OK** button.
9. KeySafe returns you to the **Import Key** panel, from which you can import another key or choose another operation.

15.3. Listing supported applications with generatekey

To list supported applications, use the command:

```
generatekey --list-apps
```

15.4. Retargeting keys with generatekey

The `--retarget` option to takes an existing key in the Security World and makes it available for use by another application as if it had been expressly generated for use by that applica-

tion. Because no key material is exposed during retargeting, this operation is as secure as generating a new key.

When you retarget a key, `generatekey` does not remove the original key from the Security World. If required, you can use KeySafe to discard the original key.

When you retarget a key, you cannot change its protection method. You cannot change the key from module-protected to card-protected, or from card-protected to module-protected.

To retarget a key, use the command:

```
generatekey --retarget [<OPTIONS>] <APPNAME> [from-application=<appname>]
[from-ident=<keyident>]
```

In this command:

Option	Description
`--retarget`	This option specifies key importation.
`<OPTIONS>`	This option specifies any options to include when the command is run. Run the command `generatekey --help` for details about the available options.
`<APPNAME>`	This option specifies the name of the application for which the key is to be generated. This must be an application for which `generatekey` can generate keys.
`from-application=<appname>`	This option specifies the name of the application with which the key is currently associated.
`from-ident=<keyident>`	This option specifies the identifier of the key to be retargeted. You can find this identifier by using the `nfkminfo` command-line utility.

If `generatekey` cannot identify the key type for retargeting, you are prompted to specify the key type. Input the key type and press Enter.

15.5. Viewing keys

You can view existing keys in the Security World using KeySafe, or the `nfkminfo` command-line utility.

15.5.1. Viewing keys with KeySafe

In order to view a list of keys with KeySafe, follow these steps:

1. Start KeySafe. (For an introduction to KeySafe and for information on starting the software, see Using KeySafe.)
2. Click the **Keys** menu button, or select **Keys** from the **Manage** menu. KeySafe takes you to the **Keys** panel, which lists all the keys in the Security World. It displays the name of the key, the application for which it was created, the protection method that was used and whether the key is stored in NVRAM.

   If you click a key's listing, KeySafe displays additional information about that key, for example, the application with which the key is used, whether or not the key is recoverable, and the key's name, creation date, hash, instance, and copy ID.

   From the **Keys** panel, you can choose to:

   - Create a new key (see Generating keys with KeySafe)
   - import a key (see Importing keys with KeySafe)
   - discard a key from the Security World (see Discarding keys)

15.5.2. Viewing keys using the command line

The `nfkminfo` command-line utility is used to list keys. To list the keys that have been created in the current Security World, use one of the following commands:

```
nfkminfo -k [<APPNAME>[<IDENT>]]
```

```
nfkminfo -l [<APPNAME>[<APPNAME>...]]
```

The `-k|--key-list` option lists keys only. The `-l|--name-list` option lists keys and their names.

With either option, `<APPNAME>` is an optional application name. If you specify an application name, `nfkminfo` lists only the keys for that application. Commonly, `<APPNAME>` is often one of:

- `custom`
- `embed`
- `pkcs11`
- `kpm`
- `kps`

- `seeconf`
- `seeinteg`
- `simple`

You can also specify your own application names for *APPNAME* as appropriate to your system.

For example, user-defined application names can be created by using the `nfkm` library to generate arbitrary keys.

With the `--name-list` option, `<IDENT>` is the key identifier.

The command `nfkminfo --key-list` returns output of the form:

```
Key summary - 4 keys:
 AppName appname    Ident <ident> AppName <appname>
Ident <ident> AppName <appname>
Ident <ident> AppName <appname>               Ident <ident>
```

To list information about a specific key, specify the `--key-list` option with an application and key identifier:

```
nfkminfo --key-list <appname> <ident>
```

This command returns output of the form:

```
Key AppName <appname> Ident <ident> BlobKA length   752
 BlobPubKA length      316
 BlobRecoveryKA length 868
 name                  "name"
 hash                  hash recovery             Enabled
 protection            CardSet
 other flags           PublicKey +0x0
 cardset               hash_ktBlobKA
 format                6 Token
 other flags           0x0
 hkm                   hash_km hkt               hash_kt hkr       none
BlobRecoveryKA
 format                8 Indirect
 other flags           0x0
 hkm                   none
 hkt                   none
 hkr                   hash_krBlobPubKA
 format                5 Module
 other flags           0x0
 hkm                   hash_km hkt               none
 hkr                   none
No extra entries
```

To list keys and names, specify the `--name-list` option. The command `nfkminfo --name-list` returns output of the form:

```
Key summary - 30 keys
in format key_<appname>_<ident> '<name>')
  key_appname_ident'name '
  key_appname_ident'name '
  key_appname_ident'name '
  key_appname_ident'name '
  key_appname_ident'name '
  key_appname_ident'name '
  key_appname_ident'name '
```

15.6. Verifying Key Generation Certificates with nfkmverify

The `nfkmverify` command-line utility verifies key generation certificates. You can use `nfkmverify` to confirm how a particular Security World and key are protected. It also returns some information about the Security World and key.

The `nfkmverify` utility compares the details in the ACL of the key and those of the card set that currently protects the key.

A key that has been recovered to a different card set shows a discrepancy for every respect that the new card set differs from the old one. For example, a key recovered from a 2-of-1 card set to a 1-of-1 card set has a different card-set hash and a different number of cards, so two discrepancies are reported. The discrepancy is between the card set mentioned in the ACL of the key and the card set by which the key is currently protected (that is, the card set mentioned in the key blobs).

A key that has been transferred from another Security World shows discrepancies and fails to be verified. Entrust recommends that you verify keys in their original Security World at their time of generation.

If you must replace your Security World or card set, Entrust recommends that you generate new keys whenever possible. If you must transfer a key, perform key verification immediately before transferring the key; it is not always possible to verify a key after transferring it to a new Security World or changing the card set that protects it.

15.6.1. Usage

To verify the key generation certificates from the command line, run the command:

```
nfkmverify [-f|--force] [-v|--verbose] [-U|--unverifiable] [-m|--module=MODULE] [appname ident [appname ident
[...]]]
```

Optionally, the command can also include the following:

Option	Description
-h|--help	This option displays help for nfkmverify.
-V|--version	This option displays the version number for nfkmverify.
-u|--usage	This option displays a brief usage summary for nfkmverify.
-m|--module=MODULE	This option performs checks with module MODULE.
-f|--force	This option forces display of an output report that might be wrong.
-U|--unverifiable	This option permits operations to proceed even if the Security World is unverifiable.  If you need the -U|--unverifiable option, there may be some serious problems with your Security World.
-v|--verbose	This option prints full public keys and generation parameters.
-C|--certificate	This option checks the original ACL for the key using the key generation certificate. This is the default.
-L|--loaded	These options check the ACL of a loaded key instead of the generation certificate.
-R|--recov	This option checks the ACL of the key loaded from the recovery blob.
--allow-dh-unknown-sg-group	This option allows an operation to proceed even if a Diffie-Hellman key is using an unrecognized Sophie-Germain group.

15.7. Discarding keys

Discarding a key deletes the information about the key from the host disk. This option is only available in KeySafe.

If you have backup media, you can restore the information and the key becomes usable again. Likewise, if you have copies of the Security World data on several computers, erasing the data on one computer does not remove it from any other computer.

To destroy a key permanently you must either erase the OCS that is used to protect it or erase the Security World completely. There are no other ways to destroy a key permanently.

15.8. Restoring keys

We do not supply tools for restoring a key that has been discarded. However if you have kept a backup of the host data for the Security World, you can restore a key from the

backup data.

If you have NVRAM-stored keys, you must additionally ensure you have a backup of the key data stored on the relevant modules.

16. Using KeySafe

KeySafe provides a GUI based interface to perform many of the main tasks required to use an nShield Security World. This appendix describes KeySafe, the Security World management tool. It includes information about:

- Starting KeySafe
- Using the graphical user interface (GUI) for KeySafe
- Using buttons to select and run operations
- Using the keyboard to navigate KeySafe
- KeySafe error reporting.

To perform Security World management, card-set management, and key management tasks using KeySafe, see the relevant chapters of this guide.

By default, KeySafe uses the same mechanisms and supports the same features and applications as the `generatekey` utility.

16.1. Setting up KeySafe

1. You must have Java JRE/JDK 1.7, 1.8 or 11. We recommend that you install Java before you install the Security World Software. The Java executable must be on your path.

   Java software is available from http://www.oracle.com/technetwork/java/. If your security policy does not allow the use of downloaded software, these components can be obtained on removable media from Oracle or from your operating system vendor.

   The `keysafe.jar` file must be specified in the Java class path.

   After you have set up the path, check that you are using the correct Java version by running java with the `-version` option.

   Example:

   ```
   >>java -version
   java version "1.8.0_05"
   Java(TM) SE Runtime Environment (build 1.8.0_05-b13)
   Java HotSpot(TM) 64-Bit Server VM (build 25.5-b02, mixed mode)
   ```

2. The Security World Software must be installed.
3. In the configuration file at `opt/nfast/kmdata/config/config`, set the following port values in the `server startup` section:

```
nonpriv_port=9000
priv_port=9001
```

You must restart the hardserver after this change.

See the Installation Guide for more about ports and firewall settings.

16.2. Starting KeySafe

Ensure that Xwindows is properly configured and running before starting KeySafe.

Start KeySafe by running the `/opt/nfast/bin/ksafe` script (assuming you installed the Security World Software in the default `/opt/nfast/` directory).

16.3. About the KeySafe window

The KeySafe window is divided into two areas:

- The sidebar (on the left), subdivided into:
  - The menu buttons (at the top of the sidebar)
  - The Security World status pane (at the bottom of the sidebar)
- The main panel area (on the right).

This layout is consistent throughout the KeySafe application.

16.3.1. Sidebar

The sidebar provides access to different parts of the KeySafe application (with the menu buttons) and also displays information about both the current Security World and your module or modules (with the Module Status tree).

The options listed below are also available from the **Manage** menu.

16.3.2. Menu buttons

There are five menu buttons at the top of the sidebar:

Menu button	Description
Introduction	Clicking the **Introduction** menu button opens the introductory panel that KeySafe displays at startup.
World	Clicking the **World** menu button opens the **World Operations** panel, from which you can: • Add modules to a Security World • Remove modules from a Security World. You cannot perform these operations on a module that is not in the pre-initialization mode. Do not use the **Initialize** option. Creating a Security World from KeySafe is deprecated.
Card Sets	Clicking the **Card Sets** menu button opens the **List Operator Card Sets** panel, from which you can: • Examine or change an Operator Card Set or its passphrase • Create a new Operator Card Set • Replace an Operator or Administrator Card Set • Discard an Operator Card Set.
Softcards	Clicking the **Softcards** menu button opens the **List Softcards** panel, from which you can: • Create a new softcard • Change or recover the passphrase on a softcard • Discard a softcard
Keys	Clicking the **Keys** menu button opens the **Keys** panel, from which you can: • Create a key • Import a key • Discard a key • View details of a key.

While KeySafe is executing a command, the menu buttons are disabled. Their normal functionality returns when the command is completed.

16.3.3. Menus

Three menu options are available from the menu bar at the top of the screen:

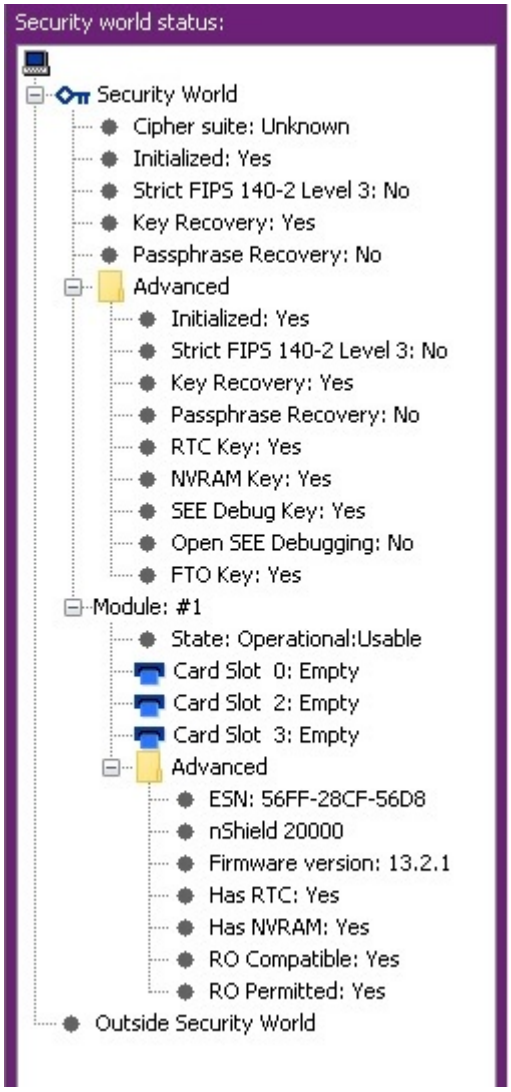
- **File**
  - **Exit** - displays a dialog asking whether you are sure you wish to quit KeySafe. Click

Yes (or press the **Enter** key) to close KeySafe. Click **No** to close the dialog and return to your KeySafe session.

- **Manage**
  - **Introduction** - opens the **Introduction** panel. See Introduction button.
  - **World** - opens the **World Operations** panel. See World button.
  - **Card sets** - opens the **List Operator Card Sets** panel. See Cardsets button.
  - **Softcards** - opens the **List Softcards** panel. See Soft Cardsets button.
  - **Keys** - opens the **Keys** panel. See Keys button.
- **Help**
  - **About KeySafe** - opens the **About KeySafe** panel, which displays current version numbers for KeySafe, kmjava and nfjava. You will need to quote these version num bers if you contact Support about KeySafe.

16.3.4. Module Status tree

The Module Status tree, in the lower part of the KeySafe sidebar, displays information about the current Security World and your modules in the form of a tree diagram.



At the top of the Module Status tree is an icon representing the computer on which the running copy of KeySafe is installed. The name of this computer is shown to the right of the icon.

Below the computer icon in the Module Status tree are icons and text identifiers representing the current Security World and your module(s). To the left of each icon is an expand/collapse toggle, or node. By default, when KeySafe starts, these nodes are collapsed and show a minus sign. Click the node to display expanded information about the Security World or module. Click the node again to collapse this information.

16.3.4.1. Security World information

At the top level of the Security World tree, the following information is displayed:

- **Cipher suite** — the type of key protecting the Security World
- **Initialized** — whether the Security World is initialized (Yes or No)

- **Strict FIPS 140 Level 3** — whether the Security World is operating at FIPS 140 Level 3 (Yes or No)
- **Key Recovery** — whether key recovery is enabled (Yes or No)
- **passphrase Recovery** — whether passphrase recovery is enabled (Yes or No). For more information, see passphrase replacement.

When the **Advanced** node is expanded, the following additional information is displayed:

- **RTC Key** — whether a real-time clock authorization key has been generated (Yes or No)
- **NVRAM Key** — whether a non-volatile memory authorization key has been generated (Yes or No)
- **SEE Debug Key** — whether SEE debugging has been enabled (Yes or No)
- **Open SEE Debugging** — whether Open SEE debugging has been enabled (Yes or No)
- **FTO Key** — whether a foreign token key has been generated (Yes or No)

16.3.4.2. Module information

Module information may be displayed either inside or outside the Security World. Modules that have not been incorporated into a Security World will be shown beneath the **Outside Security World** node.

At the top level of the Module tree, the following information is displayed:

- The module's state, which is one of the following:

Mode	Description
PreInitMode	The module is in pre-initialization mode.
InitMode	The module is in initialization mode.
Operational:Useable	The module is in the current Security World and useable for key operations.
Operational:Unknown	The mode of the module cannot be determined.
Operational:Uninitialized	The module key is set and the module must be initialized before use.
Operational:Factory	The module key is set to the factory default.
Operational:Foreign	The module is from an unknown Security World.
Operational:AccelOnly	The module is an acceleration-only module.

Mode	Description
Operational:Unchecked	Although the module appears to be in the current Security World, KeySafe cannot find a module initialization certificate (a **module_*ESN*** file) for this module
Failed	The module has failed.
PreMaintMode	The module is in the pre-maintenance mode.
MaintMode	The module is in the maintenance mode.

- the status of the smart card reader slot(s).

For FIPS 140 Level 3 Security Worlds, a **FIPS Auth Loaded** entry shows if an Administrator Card or Operator Card has been inserted to authorize an operation that requires a FIPS key.

The Module status tree has an **Advanced** folder that shows the following details when expanded:

- **ESN** — the module's electronic serial number (ESN), which is a unique identifier. You must quote a module's ESN if you need to contact Support. Keep a record of the ESN(s) associated with your module(s).
- the **HSM type** and model number
- **Firmware version** — the version of the module's firmware
- **Has RTC** — whether the module has a Real Time Clock (RTC)
- **Has NVRAM** — whether the module has nonvolatile memory (NVRAM).
- **RO Compatible** —
- **RO Permitted** —

16.3.5. Main panel area

The KeySafe main panel area is used to display information and options pertaining to a chosen operation. For example, clicking the **World** menu button takes you to the **World Operations** panel in the main panel area.

16.3.5.1. Navigation and command buttons

On each **Operations** panel there are a number of *navigation buttons*. Clicking a navigation button does *not* commit you to an action, but instead selects an operation and loads another panel of additional information and options related to the selected operation. From the **World Operations** panel, for example, clicking the **Erase Module** navigation button does

not itself erase a module, but rather loads the **Erase Module** panel.

On the next panel, the **Commit** button executes an operation, while the **Back** button returns to the previous panel. For example, on the **Erase Module** panel, clicking the **Commit** button will erase the module, while clicking the **Back** button will return to the **World Operations** panel.

Clicking the **Commit** button tells KeySafe to write or delete data: it is not necessarily possible to reverse such changes even if you subsequently cancel the operation. In some cases, clicking the **Commit** button causes KeySafe to display a dialog asking you to confirm your command. Such features help prevent you from accidentally destroying your data or keys.

Some panels require that you select options by means of radio buttons or that you enter data into text fields before clicking the **Commit** button. For example, if you click the **Reprogram Module** button on the **World Operations** panel, the next panel prompts you to choose whether the module can receive remote operator card shares.

Input may be in the form of radio buttons (allowing several options, one of which — the *default* — will be already selected) or text boxes (allowing you to enter text: no default value is set). If you click the **Commit** button without having entered data into a mandatory text field, or if KeySafe detects that the information you provided is inconsistent or invalid, KeySafe displays an error message. Click the message's **OK** button, and then provide or cor rect the necessary information.

After you successfully issue a command by clicking the **Commit** button, the menu buttons are disabled until the requested command is completed.

16.3.5.2. Navigating with the keyboard

The **Tab** key always takes you to the next field or button. If the cursor is not currently active in a text field, pressing the space bar or the **Enter** key activates the currently selected button (as if you had clicked it). Pressing the **Shift**-**Tab** button combination takes you to the previous field (if any) or deselects an automatically selected button (if any).

16.4. Errors

If KeySafe detects an error from which it cannot recover, it may display a Fatal Error message.

16.4.1. Unable to establish KeySafe session.

Error

```
Please ensure that the hardserver is running and accepting TCP connections.
Click OK to exit.
```

Possible causes

- The hardserver is unable to receive TCP connections. The server program communicates with clients by using named pipes or TCP sockets.
- The hardserver is not running, or is physically disconnected.

Suggested solutions

- Check the hardserver configuration file settings: see server_startup.
- To restart the hardserver:
  1. Exit KeySafe
  2. Restart the server (as described in Stopping and restarting the client hardserver)
  3. Restart KeySafe.

16.4.2. Unable to generate key

*Error reported by nShield hardware module in response to `GenerateKeyPair`:

```
nFast error: UnknownFlag
```

Possible causes

Your hardware or firmware may not be up to date.

Suggested solutions

To update your firmware:

1. Exit KeySafe
2. Update the firmware as described in Upgrading firmware
3. Restart KeySafe

The firmware upgrade process destroys all persistent data held in a key-management module. If your security system requires that the persistent data held in a key-management mod ule must survive intact during the upgrade or initialization of the key-management module, a backup and recovery mechanism of your `kmdata` directory must be implemented.

If you receive any error message titled **Unexpected Error**, contact Support with details of what you were doing, and the exact error message.

17. Warrant Management for nShield 5s

You do not need to manage the warrants. Entrust supplies these HSMs with the required warrants pre-installed and stored within the module. The Security World software fetches warrants from the module when they are needed.

This includes a KLF2 and a KLF3 warrant. The KLF3 warrant is currently unused and is installed in preparation for multi-tenant systems.

To view the warrants installed on a module, run `retrievewarrants`. This stores a copy of the warrants in the host file system.

18. Supplied utilities

This appendix describes the executable command-line utilities (utilities) that you can use for performing various configuration and administrative tasks related to your module.

These utilities exist in the `bin` subdirectory of your Security World Software installation. Unless noted, all utilities 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.

18.1. Utilities for general operations

Use the utilities described in this section to:

- Check the module configuration and verify that it functions as expected.
- Obtain statistics for checking the performance of the module.

18.1.1. `hsmadmin`

Manages the administration of the HSM using different subcommands. See *Administration of platform services* in the *Install Guide* for your nShield HSM.

18.1.2. `enquiry`

Obtain information about the hardserver (Security World Software server) and the modules connected to it.

- Check if the software has been installed correctly
- Check the firmware version
- Check if the Remote Operator feature is enabled
- Check the hardware status of nShield PCIe HSMs

See the Installation Guide for more information.

18.1.3. `checkmod`

Check modulo exponentiations performed on the module against the test data located in

the `opt/nfast/testdata` directory.

18.1.4. `cfg-mkdefault`

Create a default client configuration file for the hardserver configuration sections.

18.1.5. `cfg-remoteslots`

Configures Remote Operator slot imports and exports. See Remote Operator.

18.1.6. `cfg-reread`

Load the hardserver configuration from the configuration file.

18.1.7. `fet`

- Activate features on an nShield module connected to the host
- View the status of features on a connected module
- Verify that a feature has been successfully enabled on a connected module

To view the status of features, run the tool without a smart card. If a FEM card is not present, or if any of the features are not enabled successfully, the utility prompts you to indicate what to do next.

To enable features, and view the status of or verify features on an nShield HSM, use the front panel rather than the `fet` utility.

For more information, see Enabling optional features

18.1.8. `ncdate`

View, set, and update the time on a module's real-time clock.

18.1.9. `ncversions`

Obtain and verify the versions of the Security World Software components that are installed. This utility lists the following information:

- Versions of all components, irrespective of whether they are installed individually or as

part of a component bundle

- Version of each component bundle

18.1.9.1. nfdiag

Obtain information about the module and the host on which it is installed. This diagnostic utility can save information to either a ZIP file or a text file.

For more information, see nfdiag: diagnostics utility.

Run this utility only if requested to do so by Support.

a|hsmdiagnose a|Run automated health check against connected nShield HSMs. This diagnostic utility saves information to an XML file.

18.1.10. nopclearfail

Clear an HSM, put an HSM into the error state, retry a failed HSM, or change the HSM mode.

You must use a privileged connection to use this utility with the following parameters:

- change the mode of the HSM (`nopclearfail -I/M/O`)
- Clear the module (`nopclearfail -c`)

For information about changing the nShield HSM mode, see Checking and changing the mode on an nShield 5s module.

18.1.11. nvram-backup

Copy files between a module's NVRAM and a smart card, allowing files to be backed up and restored.

18.1.12. nvram-sw

View and modify information about NVRAM areas.

18.1.13. pubkey-find

Obtain information of the public key from a certificate or certificate request (in a Base-64 encoded PEM file).

18.2. randchk

Run a universal statistical test on random numbers returned by the module.

18.2.1. retrievewarrants

Retrieves warrants stored within the HSM and writes them to a file.

18.2.2. rtc

View and set the module's real-time clock.

By default, `rtc` reads the real-time clock of module 1.

- `--adjust`: The module uses the difference between its idea of the current time and the new time, together with how long it's been since the clock was last set, to compute how much its clock is drifting.
- `--set-clock`: The module's clock is set to either TIME, if it is provided as a list of six inte gers separated by non-digit characters, or to the host's current time.

18.2.3. slotinfo

- Obtain information about tokens in a module
- Format a smart card

18.2.4. snmpbulkwalk snmpget snmpgetnext snmptable snmpset snmptest snmptranslate snmpwalk

Obtain system, module, connection and software information from the SNMP agent.

For more information, see Using the SNMP command-line utilities.

18.2.5. stattree

Obtain statistics gathered by the Security World Software server and modules.

For more information, see stattree: information utility.

`sbin/logrotate-hardserver`

Archive the existing hardserver log from `/opt/nfast/log/hardserver.log` and re-open as a

fresh log file.

When run with no arguments, it will automatically archive the existing log to `/opt/nfast/log/archive/hardserver.DATETIME.log` (where *DATETIME* is the current date and time). The directory `/opt/nfast/log/archive/` is created if it does not already exist.

Optionally, a single argument can be provided with the full file name to archive the existing hardserver log to.

This script must be run as root.

18.3. Test analysis tools

Use the following utilities to test the cryptographic operational behavior of a module.

All the listed utilities, except the `floodtest` utility, are supported only on FIPS 140 Level 2 Security Worlds.

Utility	Enables you to...
`cryptest`	Test all defined symmetric cryptographic mechanisms.
`des_kat`	Perform DES known-answer tests. This utility indicates if any of them fail.
`floodtest`	Perform hardware speed-testing by using modular exponentiation.
`kptest`	Test the consistency of encryption and decryption, or of signature and verification, with the RSA and DSA algorithms.
`ncthread-test`	Stress test modules and test nCore API concurrent connection support.
`perfcheck`	Run various tests to measure the cryptographic performance of a module. For more information, see perfcheck: performance measurement checking tool.
`sigtest`	Measure module speed using RSA or DSA signatures or signature verifications.
`ncperftest`	Test the performance of various crypto commands using attached nShield hardware. Available since v12.10 it contains all the functionality in `sigtest` and `floodtest` as well as several new features and greater accuracy and throughput capability in performance management.

18.4. Security World utilities

Use the utilities described in this section to:

- Set up and manage Security Worlds.
- Create and manage card sets and passphrases.

- Generate keys and transfer keys between Security Worlds.

Utility	Enables you to...
`bulkerase`	Erase multiple smart cards including Administrator Cards, Operator Cards, and FEM activation cards, in the same session.  Do not use the `bulkerase` utility to erase Administrator Cards from the current Security World.
`cardpp`	Change, verify, and recover a passphrase of an Operator Card. For more information, see: • Verifying the passphrase of a card with cardpp. • Changing known passphrases with cardpp. • Changing unknown or lost passphrases.
`createocs`	Create and erase an OCS. For more information, see: • Creating an Operator Card Set from the command line. • Erasing cards from the command line.
`initunit`	Initialize an nShield module. For more information, see Erasing a module with initunit.
`generatekey`	Generate, import, or retarget keys. This utility is included in the `Core Tools` bundle, which contains all the Security World Software utilities. For more information, see: • Generating keys with the command line. • Importing keys from the command line. • Example of key generation with generatekey, for an example of key generation in batch mode. • Example of key importation with generatekey, for an example of importing an RSA key. • Listing supported applications with generatekey. • Retargeting keys with generatekey.
`kmfile-dump`	Obtain key management information from a Security World's key management data file.
`migrate-world`	Migrate existing keys to a destination Security World. For more information, see Security World migration.

Utility	Enables you to...
`mkaclx`	Generate non-standard cryptographic keys that can be used to perform specific functions, for example, to wrap keys and derive mechanisms. This utility includes options that are not available with the `generate-key` utility.  Ensure that you run the `mkaclx` utility with the options that are appropriate for your security infrastructure. If the appropriate options are not chosen, the security of existing keys might potentially be compromised.
`new-world`	Create and manage Security Worlds on nShield modules. You must use a privileged connection to use this utility with the following parameter: • Initialize the HSM (`new-world -e/i/l`) For more information, see: • Creating a Security World using new-world. • Adding a module to a Security World with new-world. • Erasing a module with new-world.
`nfkmcheck`	Check Security World data for consistency.
`nfkminfo`	Obtain information about a Security World and its associated cards and keys. For more information, see: • Displaying information about a Security World with nfkminfo. • Viewing card sets from the command line. • Viewing softcards with nfkminfo. • Viewing keys using the command line. • nfkminfo: information utility.
`nfkmverify`	Perform Security World verification. For more information, see Verifying Key Generation Certificates with nfkmverify.
`postrocs`	Transfer PKCS #11 keys to a new card set in the new Security World. When transferring keys by using either the `key-xfer-im` utility or the `migrate-world` utility, run the `postrocs` utility if there are any PKCS #11 keys that are protected by OCSs.  PKCS #11 keys either have `keys_pkcs_um` or `key_pkcs_uc` as the prefix.

Utility	Enables you to...
`ppmk`	• Create and manage softcards. Use this utility to: • View details of a softcard • Create and delete a softcard • View, change, and recover the passphrase of a softcard For more information, see: • Creating a softcard with ppmk. • Erasing softcards with ppmk. • Viewing softcards with ppmk. • Verifying the passphrase of a softcard with ppmk. • Changing known softcard passphrases with ppmk. • Replacing unknown passphrases with ppmk.
`preload`	Load keys into a module before an application is run in another session.
`racs`	Create a new ACS to replace an existing ACS. For more information, see Replacing an Administrator Card Set using racs.
`rocs`	• Restore an OCS from a quorum of its cards • Restore softcards For more information, see: • Replacing OCSs or softcards with rocs. • Using rocs from the command line.

18.5. CodeSafe utilities

Use the following helper utilities to develop and sign SEE machines. For more information about these utilities, see the *CodeSafe Developer Guide*.

Utility	Enables you to...
`elftool`	Convert ELF format executables into a format suitable for loading as an SEE machine.
`hsc_loadseemachine`	Load an SEE machine into each module that is configured to receive one, then publishes a newly created SEE World, if appropriate.
`loadsee-setup`	Set up the configuration of auto-loaded SEE machines.
`modstate`	View the signed module state.

Utility	Enables you to...

`see-sock-serv`
`see-stdioe-serv`
`see-stdioesock-serv`
`see-stdoe-serv`
	Activate or enable standard IO and socket connections for SEE machines using the `bsdlib` architecture.

`tct2` (Trusted Code Tool)	Sign, pack, and encrypt file archives so that they can be loaded onto an SEE-ready nShield module.

18.6. PKCS #11

Use the following utilities to manage the interfaces between the PKCS #11 library and the module.

Utility	Enables you to...
`ckcerttool`	Import a certificate as a PKCS #11 `CKO_CERTIFICATE` object of type `CKC_X_509`, and optionally, associate it with the corresponding private key.
`ckcheckinst`	Verify the installation of the nShield PKCS #11 libraries. For more information, see Checking the installation of the nCipher PKCS #11 library.
`ckimportbackend`	Generate keys for use with PKCS #11 applications. When you run the `generatekey` utility to generate PKCS #11 keys, the `ckimportbackend` utility is executed in the background.  Do not run this utility unless directed to do so by Support.
`cknfkmid`	View values of attributes of PKCS #11 objects.
`ckshahmac`	Perform a PKCS #11 test for vendor-defined `SHA1_HMAC` key signing and verification capabilities.
`cksigtest`	Measure module signing or encryption speed when used with nShield PKCS #11 library calls.

The Security World software enables you to use the following additional PKCS #11 utilities. For more information about these utilities, see the *Cryptographic API Integration Guide*.

Utility	Enables you to...
`ckinfo`	View PKCS #11 library, slot, and token information. Use this utility to verify that the library is functioning correctly.

Utility	Enables you to...
`cklist`	View details of objects on all slots. If invoked with a PIN argument, the utility lists public and private objects. If invoked with the `-n` (`--nopin`) option, the utility lists only the public objects. This utility does not output any potentially sensitive attributes, even if the object has `CKA_SENSITIVE` set to `FALSE`.
`ckmechinfo`	View details of the supported PKCS #11 mechanisms provided by the module.
`ckrsagen`	Test RSA key generation. You can use specific PKCS #11 attributes for generating RSA keys.
`cksotool`	Create a PKCS #11 Security Officer role, and manage its PIN.

18.7. Developer-specific utilities

Use the following utilities to ensure that the HSMs are functioning as expected and to test the cryptographic functionality at the nCore level.

Utility	Enables you to...
`pollbare`	Obtain information about state changes. The functionality of this test utility depends on whether the server or an HSM supports nCore API poll commands. To know if your server or HSM supports nCore API poll commands, run the `enquiry` utility.
`trial`	Test the nCore API commands. You can use this utility interactively or from a script file.

18.8. Utilities that require a privileged connection

You must be a privileged user, that is, use a privileged connection to the HSM, to run certain utilities with certain parameters.

Utility	Use case
`nopclearfail -I/M/O`	Change the mode of the HSM
`nopclearfail -c`	Clear the module
`new-world -e/i/l`	Initialize the HSM

19. Preload Utility

19.1. Overview

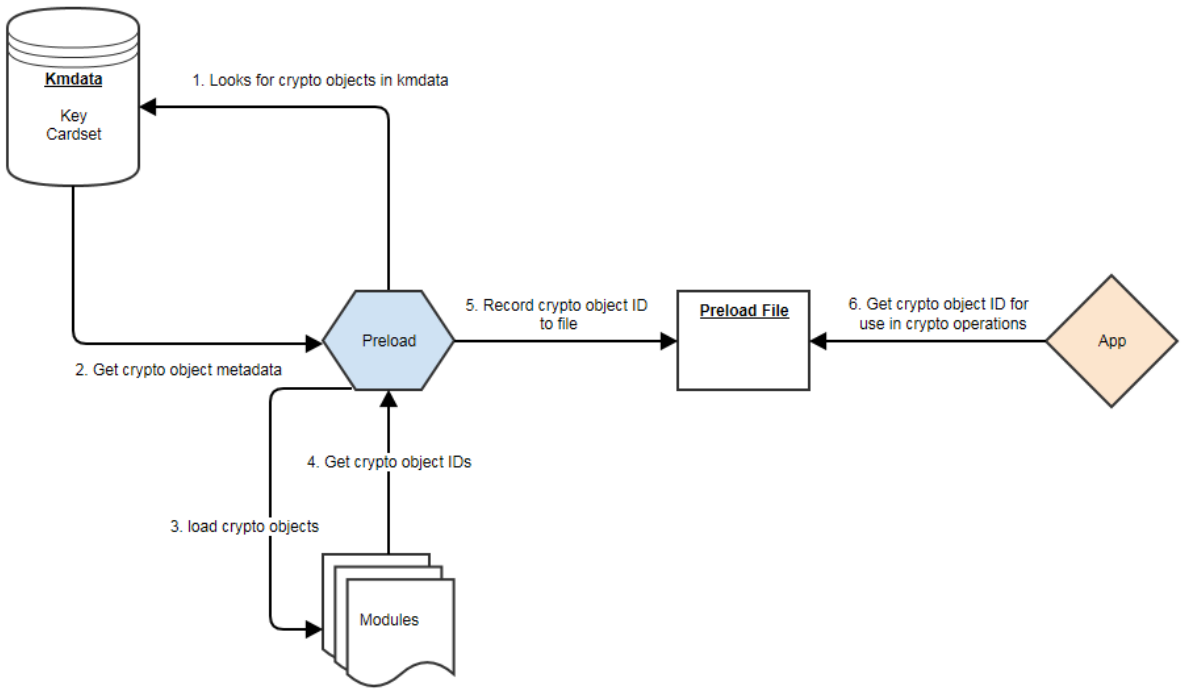
The preload utility loads persistent cryptographic objects (keys/OCS/softcards) onto a cho sen set of modules, then makes those objects available for use by applications. This removes the need for applications to load keys/cards themselves, and allows for easy shar- ing of keys/cards between multiple 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`. This binary calls the `preload.py` script found in `/opt/nfast/python/scripts`

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



19.2. Using Preload

19.2.1. 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 available 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
```

19.2.2. 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.

19.2.3. Preload Command Line Arguments

Argument	Effect
`--version`	show program's version number and exit

Argument	Effect
-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.
--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.

Argument	Effect
-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.

19.2.4. 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
```

19.3. 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`:

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

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

19.4. 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.

19.4.1. 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 protected 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
```

19.5. 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.

19.6. Admin Keys

19.6.1. 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`.

19.6.2. 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. E.G. kfips and ltfips.

19.7. 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 session.

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 (e.g. 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.

19.7.1. 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.

19.7.2. 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, error messages, etc. This is due to a necessary difference in implementation between the two modes, and is expected.

19.7.3. 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.

19.7.4. 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-availability 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 above section 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 (i.e. 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.

19.7.5. 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.

19.7.6. 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.

19.7.7. 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
```

19.7.7.1. 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

crypto object	name	protected by
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**).

19.7.8. 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
```

19.7.9. 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.

19.7.9.1. 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 section *PKCS _11 with key reloading* in the *Cryptographic API Integration Guide*.

19.7.10. 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`

19.8. Logging

By default `preload` logs to `stderr`.

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

e.g.

```
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`.

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`.

19.9. 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
```

20. Environment variables

This appendix describes the environmental variables used by Security World Software.

When you are using these environment variables on Windows to configure nShield services such as the hardserver (nFast Server service), these must be set as System variables only; not as User Variables. Any service for which the environment variable changes are intended must be restarted for the change to take effect.

Variable	Description	Win	Lnx
KERNEL_HEADERS	This variable allows you to specify the path to kernel headers (if, for example, they are not in the default directory). It is necessary for the configuration script to be able to find the kernel headers when building the PCI driver during software installation.	n	y
NFAST_CERTDIR	This variable specifies the path to the dynamic feature enabling Feature Certificates directory. You only need to change the value of this variable if you move the Installation directory. See NFAST_HOME, NFAST_KMDATA, and NFAST_LOGDIR.	y	y
NFAST_DEBUG	This variable enables debug logging for the hardserver and the PKCS #11 library. You must set NFAST_DEBUG equal to a value in the range 1 – 7 for debug messages to be logged (see Logging, debugging, and diagnostics). For more information, see also Logging and debugging information for PKCS #11 and Hardserver debugging.	y	y
NFAST_DEBUGSYSLOG	This variable redirects debug logging to syslog. The value of the environment variable should be one of the syslog facilities to be used. Prefixing the facility name with + enables traditional logging and syslog simultaneously.	y	Y
NFAST_HOME	This variable specifies the path to the Installation directory, which is set by the Security World Software installation script. You only need to change the value of this variable if you move the Installation directory. See NFAST_KMDATA, NFAST_CERTDIR, and NFAST_LOGDIR.	y	y
NFAST_KMDATA	This variable sets the location of the Key Management Data directory. You only need to change the value of this variable if you move the Key Management Data directory. See NFAST_HOME, NFAST_CERTDIR, NFAST_LOGDIR, and NFAST_KMLOCAL.	y	y

Variable	Description	Win	Lnx
NFAST_KMLOCAL	This variable specifies the location of the Key Management and Security World Data directory. If this environment variable is not set, by default the module looks for the Security World data in the local subdirectory of the Key Management Data directory. See NFAST_KMDATA.	y	y
NFAST_LOGDIR	This variable specifies the location of the Log Files directory. You only need to change the value of this variable if you move the Log Files directory. See NFAST_HOME, NFAST_KMDATA, and NFAST_CERTDIR.	y	y
NFAST_USER_LOGDIR	This variable specifies the location of log files that are specific to each user.	y	y
NFAST_NFKM_TOKENSFILE NFAST_NFKM_TOKENSSELECT	This variable sets the default values for a file in which ClientID and KeyIDs are stored by the preload command-line utility.	y	y
NFAST_SEE_MACHINEENCKEY_DEFAULT	This variable is the name of the SEEConf key needed to decrypt SEE-machine images. Running the command loadmache --encryptionkey=`IDENT (or `loadmache --unencrypted) overrides any value set by this variable.	y	y
NFAST_SEE_MACHINEENCKEY_<module>	This variable is the name of the SEEConf key needed to decrypt the SEE-machine image targeted for the specified module. It overrides NFAST_SEE_MACHINEENCKEY_DEFAULT for the specified module. Running the command loadmache --encryptionkey=<IDENT> (or loadmache --unencrypted) overrides any value set by this variable.	y	y
NFAST_SEE_MACHINEIMAGE_DEFAULT	This variable is the path of the SEE machine image to load on to any module for which a specific image is not defined. Supplying the machine-filename parameter when running the loadmache command-line utility overrides this variable. This variable is not affected when running the loadsee-setup or hsc_loadseemachine utilities.	y	y
NFAST_SEE_MACHINEIMAGE_<module>	This variable is the path of the SEE machine image to load on to the specified module. If set, this variable overrides the use of NFAST_SEE_MACHINEIMAGE_DEFAULT for the specified module. Supplying the <machine-filename> parameter when running the loadmache command-line utility overrides the NFAST_SEE_MACHINEIMAGE_<module> variable. This variable is not affected when running the loadsee-setup or hsc_loadseemachine utilities.	y	y

Variable	Description	Win	Lnx
NFAST_SEE_MACHINESIGHASH_DEFAULT	This variable is the default key hash of the vendor signing key (seeinteg) that signs SEE machine images. This variable is only required if you are using a dynamic SEE feature with an encrypted SEE machine. Running the command loadmache --sighash=<HASH> any value set in this variable.	y	y
NFAST_SEE_MACHINESIGHASH_<module>	This variable is the key hash of the vendor signing key (seeinteg) that signs SEE machine images for the specified module. It overrides NFAST_SEE_MACHINESIGHASH_DEFAULT for the specified module. This variable is only required if you are using a dynamic SEE feature with an encrypted SEE machine. Running the command loadmache --sighash=<HASH> any value set in this variable.	y	y
NFAST_SERVER NFAST_PRIVSERVER	If these variables are set in the hardserver's environment, the values specify: **On Linux**, the pathnames of the UNIX domain sockets that the hardserver uses for ordinary/privileged client connections to the hardserver. **On Windows**, the names of the Windows named pipes for ordinary/privileged client connections to the hardserver. These variables are available for this purpose for backward compatibility only; you should configure sockets in the hardserver configuration file, see server_startup	y	y
NFAST_SERVER_PORT NFAST_SERVER_PRIVPORT	If these variables are set in the hardserver's environment, the values specify the TCP port numbers that the nFast server uses for connections over TCP sockets. These variables are available for this purpose for backward compatibility only: you should configure ports in the hardserver configuration file, as described in server_startup. If you set these variables, they override the values in the hardserver configuration file.	y	y
NFLOG_CATEGORIES	This variable is used to filter log messages by supplying a colon-separated list of allowable message categories; see Logging, debugging, and diagnostics. If no value is supplied, all message categories are logged.	y	y

Variable	Description	Win	Lnx
NFLOG_SEVERITY	This variable is used to filter log messages by supplying a minimum severity level to be logged; see Logging, debugging, and diagnostics. If no value is supplied, the default severity level is WARNING.	y	y
NFLOG_DETAIL	This variable is used to filter log messages by supplying a bitmask of detail flags; see Logging, debugging, and diagnostics. The default is time+severity+writeable.	y	y
NFLOG_FILE	This variable is used to specify a filename (or file descriptor) in which log messages are to be written; see Logging, debugging, and diagnostics. The default is stderr (the equivalent of file descriptor &2).	y	y

21. Logging, debugging, and diagnostics

This appendix describes the settings and tools you can use to access the logging and debugging information generated by the Security World Software. You are also shown how to obtain system information using the `nfdiag` command-line utility.

21.1. Logging and debugging

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.

21.1.1. 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

Hexadecimal flag	Function	logdetail flags
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
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.

Category	Description
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.)
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` 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.

Category	Description
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.
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.

21.1.2. 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 `/opt/nfast/cknfastrc` file. Normally settings in the environment should override 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 "

21.1.3. Hardserver debugging

Hardserver debugging is controlled by specifying one or more servdbg-* categories (from the NFLOG_CATEGORIES environment variable) in the hardserver configuration file; see server_settings. 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.

Flag	Result
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.

21.1.4. Debugging information for Java

This section describes how you can specify the debugging information generated by Java.

21.1.4.1. 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)

21.1.4.2. 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.

21.2. 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.

21.2.1. nfdiag: diagnostics utility

The `nfdiag` command-line utility is a diagnostics tool that gathers information about the sys tem on which it is executed. It can save this information to either a `.zip` file or a text file.

Under normal operating conditions, you do not need to run `nfdiag`. You can run `nfdiag` before contacting Support and include its output file with any problem report.

21.2.1.1. Usage

Run `nfdiag` with the standard `-h|--help` option to display information about the options and parameters that control the program's behavior.

If you want to supply additional diagnostic files, run:

```
nfdiag -e|--extrainfo <FILENAME>
```

You can only attach plaintext files.

The `nfdiag` command-line utility is an interactive tool. When you run it, it prompts you to supply the following information:

Option	Actions to take
`which application(s) you are using`	Identify all application software installed on the machine on which any problem with the nShield product occurs.
`what APIs you are using`	Describe any custom software, especially any interaction it has with the nShield security system.
`a description of the problem`	Include as much detail as possible, including any error messages you have seen.
`a Support ticket number` (if you have one)	When you contact Support you are supplied with a Support ticket number. Enter this number to help Support expedite the collection of any information you have sent.
`a contact email address`	Supply an email address that has as few e-mail/spam filters as possible so that any additional files that Support sends to you are not blocked. We use the e-mail address you supply here *only* for communication directly related to your problem report.
`a contact name`	Enter your name (or the name of an appropriate person for contact by Support).
`a contact telephone number`	Include the appropriate country and any region code for your contact telephone number.

Except for a Support ticket number, nfdiag requires non-NULL answers to all its prompts for information.

Supplying this information helps `nfdiag` capture as much relevant information as possible for any problem report to Support. As you supply information at each prompt in turn, press **Enter** to confirm the information and continue to the next prompt. Information you supply cannot extend over multiple lines, but if you need to supply this level of information, you can include it in additional attached files, as described.

By default, `nfdiag` runs in verbose mode, providing feedback on each command that it executes and which log files are available. If the system is unable to execute a command, the verbose output from `nfdiag` shows where commands are stalling or waiting to time out.

At any time while `nfdiag` is running, you can type **Ctrl**-**C** to cancel its current commands and re-run it.

21.2.1.2. Output

After you have finished supplying information for each required prompt, `nfdiag` generates a plain text output file and displays its file name.

If the file `opt/nfast/log/logfile` exists, `nfdiag` automatically includes this file in its output. If the file `opt/nfast/log/logfile` does not exist, `nfdiag` warns you that it could not process this file. This warning does not affect the validity of the generated output file.

When complete, this output file contains the following:

- The information supplied interactively to nfdiag when run
- Details about the client machine
- Details about any environment variables
- Output from the following command-line utilities:
  - `enquiry`
  - `stattree`
  - `ncversions`
  - `nfkminfo`
- The contents of the following log files (if they are available):
  - `hardserver.log`
  - `keysafe.log`
  - `cmdadp.log`
  - `ncsnmpd.log`

- Any attached diagnostic files

Because the contents of the output file are plain text, they are human readable. You can choose to view the output file to ensure no sensitive information has been included.

The `nfdiag` utility does not capture any passphrases in the output file.

21.2.2. nfkminfo: information utility

The `nfkminfo` utility displays information about the Security World and the keys and card sets associated with it.

21.2.2.1. Usage

```
nfkminfo -w|--world-info [-r|--repeat] [-p|--preload-client-id]
```

```
nfkminfo -k|--key-list [<APPNAME> [<IDENT>]]
```

```
nfkminfo -l|--name-list [<APPNAME> [<APPNAME>...]]
```

```
nfkminfo [-c|--cardset-list]|[-s|--softcard-list] [<TOKENHASH>]
```

```
nfkminfo --cardset-list [<TOKENHASH>] --key-list [<APPNAME>[<APPNAME>]]|--name-list <APPNAME>[<IDENT>...]]
```

21.2.2.1.1. Security World options

`-w|--world-info`

This option specifies that you want to display general information about the Security World. These options are the default and need not be included explicitly.

`-r|--repeat`

This option displays the information repeatedly. There is a pause at the end of each set of information. The information is displayed again when you press Enter.

`-p|--preload-client-id`

This option displays the preloaded client ID value, if any.

21.2.2.1.2. Key, card set, and softcard options

`-k|--key-list [<APPNAME>[<APPNAME>]]`

This option lists keys without key names. If `<APPNAME>` is specified, only keys for these applications are listed.

`-l|--name-list [<APPNAME>[<IDENT>]]`

This option lists keys with their names. If `<APPNAME>` is specified, only keys for these applications are listed. If `<IDENT>` is listed, only the keys with the specified identifier are listed.

`-c|--cardset-list [<TOKENHASH>]`

If `<TOKENHASH>` is not specified, this option lists the card sets associated with the Security World. The output is similar to this:

```
Cardset list - 1 cardsets: (P)ersistent/(N)ot, (R)emoteable/(L)ocal-only
Operator logical token hash      k/n timeout name <hash>                          1/1 none-PL <name>
```

If `<TOKENHASH>` is specified, these options list the details of the card identified by *hash*. The output is similar to this:

```
Cardset
 name        "name"
 k-out-of-n  1/1
 flags       Persistent PINRecoveryForbidden(disabled) !RemoteEnabled
 timeout     none
 card names   ""
 hkltu       hash
 gentime 2005-10-14 10:56:54
Keys protected by cardset hash:
 AppName app Ident keyident
 AppName app Ident keyident
 ...     ...  ...   ...
```

`-s|--softcard-list TOKENHASH`

This option works like the `-c|--cardset-list` option, except it lists softcards instead of card sets. If `<TOKENHASH>` is not specified, this option lists the softcards associated with the Security World.

21.2.2.2. Security World output info

If you run `nfkminfo` with the `-w|--world-info` option, it displays information similar to that shown in these examples:

```
generation 1
state      0x70000 Initialised Usable Recovery !PINRecovery
```

```
!ExistingClient !RTC !NVRAM !FTO !SEEDebug
n_modules  1
hknso      hash_knso
hkm        hash_km
hkmwk      hash_knwk
hkre       hash_kre
hkra       hash_kra
ex.client  none

...
```

```
...
Module #1
 generation    1
 state         0x1 Usable
 flags         0x10000 ShareTarget
 n_slots       2
 esn           34F3-9CB4-753B
 hkml          hash_kml
 Module #1 Slot #0 IC 11
 generation    1
 phystype      SmartCard
 slotlistflags 0x2
 state         0x4 Operator
 flags         0x20000 RemoteEnabled
 shareno       2
 shares
 error         OK
 Cardset
 name          "fred"
 k-out-of-n    1/2
 flags         NotPersistent
 timeout       none
 card names    "" ""
 hkltu         hash_kt
 Module #1 Slot #1 IC 0
 generation    1
 phystype      SmartCard
 slotlistflags 0x2 SupportsAuthentication
 state         0x4 Admin
 flags         0x10000 passphrase
 shareno       1
 shares        LTNSO(PIN) LTM(PIN) LTR(PIN) LTNV(PIN) LTRTC(PIN) LTDSEE(PIN)
 LTFTO(PIN)
 error         OK
 No Cardset

No Pre-Loaded Objects
```

21.2.2.2.1. World

`nfkminfo` reports the following information about the Security World:

`generation`

This indicates the internal number.

`state`

This indicates the status of the current world:

`Initialised`	This indicates that the Security World has been initialized.
`Usable`	This indicates that there is at least one usable HSM in this Security World on this host.
`!Usable`	This indicates that there are no usable HSMs in this Security World on this host.
`Recovery`	This indicates that the Security World has the OCS and softcard replacement and the key recovery features enabled.
`!Recovery`	This indicates that the Security World has the OCS and softcard replacement and the key recovery features disabled.
`AdminAuthRequired`	This indicates that additional authorization is required for the following operations: • Key generation • Public key import • Operator cardset creation • Softcard creation. This authorization is supplied by presenting any operator or administration card from the same Security World. A passphrase is not required:
`ExistingClient`	This indicates that there is a Client ID set, for example, by `preload`. This Client ID is given in the `ex.client` output if the `--preload-client-id` flag was supplied.
`!ExistingClient`	This indicates that no Client ID is set. The `ex.client` output will be empty.
`AlwaysUseStrongPrimes`	This indicates that the Security World always generates RSA keys in a manner compliant with FIPS 186-3.
`!AlwaysUseStrongPrimes`	This indicates that the Security World leaves the choice of RSA key generation algorithm to individual clients.
`SEEDebug`	This indicates that the Security World has an SEE Debugging delegation key.
`!SEEDebug`	This indicates the Security World has no SEE Debugging delegation key.
`SEEDebugForAll`	This indicates no authorization is required for SEE Debugging.
`PINRecovery`	This indicates that the Security World has the passphrase replacement feature enabled.
`!PINRecovery`	This indicates that the Security World has the passphrase replacement feature disabled.
`FTO`	This indicates that the Security World has an FTO delegation key.
`!FTO`	This indicates that the Security World has no FTO delegation key.
`NVRAM`	This indicates that the Security World has an NVRAM delegation key.
`!NVRAM`	This indicates that the Security World has no NVRAM delegation key.

`RTC`	This indicates that the Security World has an RTC delegation key.
`!RTC`	This indicates that the Security World has no RTC delegation key.
`AuditLogging`	This indicates that Audit Logging is enabled for this Security World.
`!AuditLogging`	This indicates that Audit Logging is not enabled for this Security World.

`n_modules`

This indicates the number of nShield HSMs connected to this computer.

`hknso`

This indicates the SHA-1 hash of the Security Officer's key.

`hkm`

This indicates the SHA-1 hash of the Security World key.

`hkmwk`

This indicates the SHA-1 hash of a dummy key used to load the Administrator Card Set (the dummy key is the same on all HSMs that use Security Worlds and is not secret).

`hkre`

This indicates the SHA-1 hash of the recovery key pair.

`hkra`

This indicates the SHA-1 hash of the recovery authorization key.

`ex.client`

This indicates the `ClientID` required to use any pre-loaded keys and tokens.

`k-out-of-n`

This indicates the values of *K* and *N* for this Security World.

`other quora`

This indicates the number (quora) of Administrator Cards (*K*) required to perform certain other functions as configured for this Security World.

`ciphersuite`

This indicates the name of the Cipher suite that the Security World uses.

`Mode`

none	This indicates that the Security World is in an unregulated mode. The Security World can be configured to meet the needs of your security policy. This includes, but is not limited to, creating a Security World that is compliant with FIPS140 Level 2.
fips1402level3	This indicates that the Security World is in a mode compliant with FIPS 140 Level 3.
commoncriteriacmts	This indicates that the Security World is in a mode compliant with Common Criteria Protection Profile EN 419 221-5, for Cryptographic Modules for Trust Services.

Assigned Keys

max usage	This indicates the maximum key usage reauthorization condition for Assigned Keys. (common-criteria-cmts mode only).
max timeout	This indicates the maximum key timeout reauthorization condition for Assigned Keys (common-criteria-cmts mode only).

21.2.2.2.2. Module

For each HSM in the Security World, nfkminfo reports:

generation

This indicates the version of the HSM data.

state

This indicates one of the following:

PreInitMode	This indicates that the HSM is in the pre-initialization state.
InitMode	This indicates that the HSM is in the initialization state.
Unknown	This indicates that the HSM's state could not be determined.
Usable	This indicates that the HSM is programmed in the current Security World and can be used.
Uninitialized	This indicates that the HSM does not have the Security Officer's key set and that the HSM must be initialized before use.
Factory	This indicates that the HSM has module key zero only and that the Security Officer's key is set to the factory default.
Foreign	This indicates that the HSM is from an unknown Security World.
AccelOnly	This indicates that the HSM is acceleration only.

`Unchecked`	This indicates that, although the HSM appears to be in the current Security World, `nfkminfo` could not find a module initialization certificate (a `module_<ESN>` file) for this HSM.
`Failed`	This indicates that the HSM has failed. For nShield PCIe HSMs running firmware 2.61.2 and above, use the `enquiry` utility for further information about the failure reason.
`MaintMode`	This indicates that the HSM is in the maintenance state.

`flags`

This displays ShareTarget if the HSM has been initialized to allow reading of remote card sets.

`n_slots`

This indicates the number of slots on the HSM (there is one slot for each physical smart card reader, one slot for each soft token, one slot for each available Remote Operator slot and one slot for each associated Dynamic Slots).

`esn`

This indicates the electronic serial number of the HSM (if the HSM is not in the `Usable` state, the electronic serial number may not be available).

`hkml`

This indicates the hash of the HSM signing key (if the HSM is not in the `Usable` state, this value may not be available).

21.2.2.2.3. Slot

For each slot on the HSM, `nfkminfo` reports:

`IC`

This indicates the insertion count for this slot (which is 0 if there is no card in the slot).

`generation`

This indicates the version of the `slotinfo` structure.

`phystype`

This indicates the type of slot, which can be one of:

- `SmartCard`

- SoftToken

slotlistflags

These are flags describing the capabilities of the slot. Single letters in parentheses are the flag codes reported by the slotinfo utility:

0x2	(A) SupportsAuthentication This indicates that the slot supports token-level challenge-response authentication.
0x40000	(R) RemoteSlot This indicates that the slot is a Remote Operator slot that has been imported from a remote HSM.
0x80000	(D) DynamicSlot This indicates that it is a Dynamic Slot.
0x100000	(a) Associated This indicates that a Remote Administration Client has associated a card reader with this
0x200000	(t) TimedOut This indicates that no response has been received from the smartcard in this Dynamic Slot within the configured timeout.
0x400000	(f) SecureChannelFailed This indicates that the secure channel between the HSM and the smartcard in this Dynamic Slot has failed in some way.

state

This can be one or more of the following flags:

Blank	This indicates that the smart card in the reader is unformatted.
Admin	This indicates that the smart card in the reader is part of the Administrator Card Set.
Empty	This indicates that there is no smart card in the reader.
Error	This indicates that the smart card in the reader could not be read (the card may be from a different Security World).
Operator	This indicates that the smart card in the reader is an Operator Card.

flags

This displays passphrase if the smart card requires a passphrase.

shareno

This indicates the number of the card within the card set.

`shares`

If the card in the slot is an Operator Card, no values are displayed for `shares`.

If the card in the slot is an Administrator Card, values are displayed indicating what key shares are stored on the card. Each share is prefixed with the letters `LT` (Logical Token), and the remaining letters identify the key (for example, the value `LTNSO` indicates that a share of K_{NSO}, the Security Officer's key, is stored on the card).

`error`

This indicates the error status encountered if the smart card could not be read:

`OK`	This indicates that there were no errors.
`TokenAuthFailed`	This indicates that the smart card in the reader failed challenge response authentication (the card may come from a different Security World).
`PhysTokenNotPresent`	This indicates that there is no card in the reader.

If you purchased a developer kit, you can refer to the relevant developer documentation for a full list of error codes.

21.2.2.2.4. Card set

If there is an Operator Card in the reader, `nfkminfo` reports:

`name`

This indicates the name given to this card set.

`k-out-of-n`

This indicates the values of *K* and *N* for this card.

`flags`

This displays one or more of each of the following pairs of flags:

`NotPersistent`	This indicates that the Operator Card is not persistent.
`Persistent`	This indicates that the Operator Card is persistent.
`NotRemoteEnabled`	This indicates that the card in the slot is not from a Remote Operator Card Set.
`RemoteEnabled`	This indicates that the card in the slot is from a Remote Operator Card Set.

`PINRecoveryForbidden(disabled)`	This indicates that the card in the slot does not have passphrase replacement enabled. This is always true if passphrase replacement is disabled for the Security World.
`PINRecoveryRequired(enabled)`	This indicates that the card in the slot does have passphrase replacement enabled.

`timeout`

the period of time in seconds after which the HSM automatically removes the Operator Card Set. If timeout is set to `none`, the Operator Card Set does not time out.

`card`

lists the names of the cards in the set, not all software can give names to individual cards in a set.

`hkltu`

the SHA-1 hash of the secret on the card.

21.2.3. perfcheck: performance measurement checking tool

Use the `perfcheck` command-line utility to run various tests measuring the cryptographic performance of an nShield HSM.

Run `perfcheck` with the standard `-h|--help` option to display information about the options and parameters that control the program's behavior.

The available tests are grouped into suites:

- `kx` (key exchange)
- `keygen` (key generation)
- `signing` (signing)
- `verify` (verification)
- `enc` (encryption)
- `dec` (decryption)
- `misc` (miscellaneous).

To see the list of tests available in a particular suite, run a command of the form:

```
perfcheck --list suite
```

For example, to list all the `signing` tests, run the command:

```
perfcheck --list signing
>>> Suite `signing' -- Signing (222 tests)
>>>    1 - DSA using RIPEMD160 with 512-bit p and 160-bit q.
>>>    2 - DSA using RIPEMD160 with 1024-bit p and 160-bit q.
>>>    3 - DSA using RIPEMD160 with 2048-bit p and 160-bit q.
>>>    4 - DSA using RIPEMD160 with 3072-bit p and 160-bit q.
>>> ...
```

In the output, each listed test in the suite is identified with a number.

You can reference a test either by its number or by its name:

- by test number:

  ```
  perfcheck suite:test_number
  ```

  To use test 16 of the `signing` suite:

  ```
  perfcheck signing:16
  ```

- by test name:

  ```
  perfcheck "exact name"
  ```

  Example:

  ```
  perfcheck "signing:RSA using RSApPKCS1 with 2048-bit n."
  ```

The test numbers change between releases. If you want to rerun tests for comparison, refer ence the tests by their names.

`perfcheck` prints the results of individual tests to output as it goes along, and then prints a full report at the end. By default, `perfcheck` runs each test three times for both minimum and maximum queue sizes, and then collates the results in the final report. See `--help` for the options to adjust this behavior.

Optionally, `perfcheck` can write its output to a directory in multiple formats using the `--out putdir` option to specify a directory name. This will create a new subdirectory under the specified directory to write the output. The `--nosubdir` option can be added as well to write output to the specified directory directly, in which case that directory must not already exist. The output directory will contain `perfcheck.html`, `perfcheck.txt`, `perfcheck.csv`, and `perfcheck.json` files that contain the report in HTML, text, CSV, and JSON format respec- tively. JSON files that contain the detailed results of individual tests will also be written to the output directory.

Output reports from test suites include the following information about each test:

Value	Description
Queue	This value is the number of outstanding jobs in the queue when the test was run. By default, most tests run both with a queue of 1, and with a fully maxed out module queue, to give an indication of both one-at-a-time performance and the bandwidth for the operation. The queue can be set differently using the `--queue` option, in which case only that queue length will be run with, except for some `misc` suite tests which set their own queue.
Rate (Units/s)	This value is a measure of throughput. It is calculated by dividing the number of repetitions by total time. If a test has been rerun to improve accuracy, as is the case by default, then this is the mean across all the runs. Some tests, for example `enc`, set the Unit to something other than an operation, for example KB, to indicate the amount of data that can be encrypted.
Min latency (ms)	This value is the time in milliseconds that the quickest individual job across all the test runs took to round-trip.
Mean latency (ms)	This value is the mean time in milliseconds that jobs took to round-trip. If a test has been rerun, this is the mean of the mean latency values from each run.
Max latency (ms)	This value is the time in milliseconds that the slowest individual job across all the test runs took to round-trip.
CV (%)	This value is the coefficient of variation expressed as a percentage of the mean latency. It gives an indication of the variability in the time it takes individual jobs to complete. If a test has been rerun, this is the mean of the CV (%) values from each run.
Min rate (tps)	This is the estimated lower bound of the throughput for this queue size in transactions per second. The value becomes more accurate if more test runs of the same test are done. When it is compared against Mean rate (tps) and Max rate (tps), Min rate (tps) gives an indication of the variability between runs.
Mean rate (tps)	This is a measure of throughput. Unlike Rate (Units/s), it is expressed in transactions per second, that is, as the number of jobs that round-trip per second. Mean rate (tps) is included for comparison against the Min rate (tps) and Max rate (tps) figures.

Value	Description
Max rate (tps)	This is the estimated upper bound of the throughput for this queue size in transactions per second. The value becomes more accurate if more test runs of the same test are done. When it is compared against Min rate (tps) and Mean rate (tps), Max rate (tps) gives an indication of the variability between runs.
Reps	This value is the number of repetitions that were actually carried out, that is, the number of jobs that were round-tripped over all tests of this operation for this queue size. If a test was rerun, this is the sum of the repetitions for each run. The target repetitions for an individual run can be set using the `--repetitions` option but note that in most cases more repetitions will be run depending on the `--accuracy` setting provided that the timeout is not reached. It is recommended to set `--accuracy` rather than `--repetitions` to control the accuracy of the test instead of adjusting the repetitions.

21.2.3.1. How perfcheck calculates statistics

When an nCore command is submitted to an HSM by a client application, it is processed as follows:

1. The command is passed to the hardserver.
2. The client hardserver encrypts the command.
3. When the HSM is free, the command is submitted from the hardserver queue.
4. The command is executed by the HSM, and the reply is given to the hardserver.
5. The unit hardserver queues the reply.
6. The unit hardserver sends the command back to the client hardserver over the network.
7. When the client application is ready, the queued reply is returned to it.

Because an HSM can execute several commands at once, throughput is maximized by ensuring there is always at least one command in the hardserver queue (so that there are always commands available to give to the HSM).

The `perfcheck` utility sends multiple simultaneous nCore commands to keep the HSM busy. It can send more commands if a required number of repetitions has not yet been reached.

After sending some initial commands, `perfcheck` begins marking commands with the time at which are submitted; when a command comes back with a timestamp, `perfcheck` checks the amount of time needed to complete the command and updates the values for `Std dev`

and `Latency`. The value of `Total time` is the amount of time from sending the first job to receiving the final one.

21.2.4. stattree: information utility

The `stattree` utility returns the statistics gathered by the hardserver and HSMs.

21.2.4.1. Usage

```
stattree [<node> [<node> [...]]]
```

21.2.4.2. Output

Running the `stattree` utility displays a snapshot of statistics currently available on the host machine. Statistics are gathered both by the hardserver (relating to the server itself, and its current clients) and by each attached HSM.

Times are listed in seconds. Other numbers are integers, which are either real numbers, IP addresses, or counters. For example, a result `-CmdCount 74897` means that there have been 74,897 commands submitted.

A typical fragment of output from `stattree` looks like this:

```
+PerModule:
  +#1:
     +ModuleObjStats:
        -ObjectCount          5
        -ObjectsCreated       5
        -ObjectsDestroyed     0
     +ModuleEnvStats:
        -MemTotal             15327232
        -MemAllocKernel       126976
        -MemAllocUser         0
     +ModuleJobStats:
        -CmdCount             169780
        -ReplyCount           169778
        -CmdBytes             3538812
        -ReplyBytes           4492764
        -HostWriteCount       169772
        -HostWriteErrors      0
        -HostReadCount        437472
        -HostReadErrors       0
        -HostReadEmpty        100128
        -HostReadDeferred     167578
        -HostReadTerminated   0
        -PFNIssued            102578
        -PFNRejected          1
        -PFNCompleted         102577
        -ANIssued             1
        -CPULoadPercent       0
     +ModuleSerialStats:
```

```
        -HostReadCount        437476
        -HostReadDeferred     167580
        -HostReadReconnect    167579
        -HostReadErrors       0
        -HostWriteCount       169774
        -HostWriteErrors      0
     +ModuleDriverStats:
         -DriverIRQs           2547906
         -DriverReadIRQs       1274069
         -DriverWriteIRQs      1276373
         -DriverWriteFails     0
         -DriverWriteBlocks    1276373
         -DriverWriteBytes     49625888
         -DriverReadFails      0
         -DriverReadBlocks     0
         -DriverReadBytes      0
         -DriverEnsureFail     0
         -DriverEnsure         1274065
```

PerModule, ModuleObjStats, and ModuleEnvStats are node tags that identify classes of statistics. 1 identifies an instance node.

ObjectCount, MemTotal, and the remaining items at the same level are statistics IDs. Each has a corresponding value.

If <node> is provided, stattree uses the value given as the starting point of the tree and displays only information at or below that node in the tree. Values for <node> can be numeric or textual. For example, to view the object counts for local module number 3:

```
$ stattree PerModule 3 ModuleObjStats
+#PerModule:
  +#3:
     +#ModuleObjStats:
        -ObjectCount          6
        -ObjectsCreated       334
        -ObjectsDestroyed     328
```

The value of <node> must be a node tag; it must identify a node in the tree and not an individual statistic. Thus, the following command does not work:

```
$ stattree PerModule 3 ModuleObjStats ObjectCount
+#PerModule:
  +#3:
     +#ModuleObjStats:
Unable to convert 'ObjectCount' to number or tag name.
```

ModuleDriverStats fields:

Field	Description
DriverIRQs	Total number of interrupts
DriverReadIRQs	Read interrupts

Field	Description
DriverWriteIRQs	Write interrupts
DriverWriteFails	Write failures
DriverWriteBlocks	Blocks written
DriverWriteBytes	Bytes written
DriverReadFails	Read failures
DriverReadBlocks	Blocks read
DriverReadBytes	Bytes read
DriverEnsureFail	Read request failures
DriverEnsure	Read requests

21.2.4.2.1. Node tags

These hold statistics for each HSM:

Category	Contains
`ModuleJobStats`	This tag holds statistics for the Security World Software commands (jobs) processed by this HSM.
`ModulePCIStats`	This tag does not apply to nShield USB-attached HSMs.
`ServerGlobals`	Aggregate statistics for all commands processed by the hardserver since it started. The standard statistics (as described below) apply to the commands sent from the hardserver to HSMs. Commands processed internally by the server are not included here. The Uptime statistic gives the total running time of the server so far.
`Connections`	Statistics for connections between clients and the hardserver. There is one node for each currently active connection. Each node has an instance number that matches the log message generated by the server when that client connected. For example, when the hardserver message is `Information: New client #24 connected`, the client's statistics appear under node #24 in the `stattree` output.
`PerModule`	Statistics kept by the HSMs. There is one instance node for each HSM, numbered using the standard HSM numbering. The statistics provided by each HSM depend on the HSM type and firmware version.
`ModuleObjStats`	Statistics for the HSM's Object Store, which contains keys and other resources. These statistics may be useful in debugging applications that leak key handles, for example.
`ModuleEnvStats`	General statistics for the HSM's operating environment.

21.2.4.2.2. Statistics IDs

ID	Value
`Uptime`	The length of time (in seconds) since an HSM was last reset, the hardserver was started, or a client connection was made.
`CmdCount`	The total number of commands sent for processing from a client to the server, or from the server to an HSM. Contains the number of commands currently being processed.
`ReplyCount`	The total number of replies returned from server to client, or from HSM to server.
`CmdBytes`	The total length of all the command blocks sent for processing.
`ReplyBytes`	The total length of all the reply blocks received after completion.
`CmdMarshalErrors`	The number of times a command block was not understood when it was received. A nonzero value indicates either that the parties at each end of a con nection have mismatched version numbers (for example, a more recent hard-server has sent a command to a less recent HSM that the HSM does not under stand), or that the data transfer mechanism is faulty.
`ReplyMarshalErrors`	The number of times a reply was not understood when it was received. A nonzero value indicates either that the parties at each end of a connection have mismatched version numbers (for example, a more recent hardserver has sent a command to a less recent HSM that the HSM does not understand), or that the data transfer mechanism is faulty.
`ClientCount`	The number of client connections currently made to the server. This appears in the hardserver statistics.
`MaxClients`	The maximum number of client connections ever in use simultaneously to the hardserver. This gives an indication of the peak load experienced so far by the server.
`DeviceFails`	The number of times the hardserver has declared a device to have failed. The hardserver provides a diagnostic message when this occurs.
`DeviceRestarts`	The number of times the hardserver has attempted to restart an HSM after it has failed. The hardserver provides a Notice message when this occurs. The message does not indicate that the attempt was successful.
`QOutstanding`	The number of commands waiting for an HSM to become available on the specified client connection. When an HSM accepts a command from a client, this number decreases by 1 and `DevOutstanding` increases by 1. Commands that are processed purely by the server are never included in this count.
`DevOutstanding`	The number of commands sent by the specified client that are currently exe-cuting on one or more HSMs. When an HSM accepts a command from a client, this number increases by 1 and `QOutstanding` decreases by 1. Commands that are processed purely by the server are never included in this count.

ID	Value
LongOutstanding	The number of LongJobs sent by the specified client that are currently executing on one or more HSMs. When an HSM accepts a LongJobs command from a client, this number increases by 1 and QOutstanding decreases by 1. Commands that are processed purely by the server are never included in this count.
RemoteIPAddress	The remote IP address of a client who has this connection. A local client has the address 0.0.0.0.
HostWriteCount	The number of write operations (used to submit new commands) that have been received by the HSM from the host machine. One write operation may contain more than one command block. The operation is most efficient when this is the case.
HostWriteErrors	The number of times the HSM rejected the write data from the host. A nonzero value may indicate that data is being corrupted in transfer, or that the hardserver/device driver has got out of sync with the HSM's interface.
HostWriteBadData	Not currently reported by the HSM. Attempts to write bad data to the HSM are reflected in HostWriteErrors.
HostWriteOverruns	Not currently reported by the HSM. Write overruns are reflected in HostWriteErrors.
HostWriteNoMemory	Not currently reported by the HSM. Write failures due to a lack of memory are reflected in HostWriteErrors.
HostReadCount	The number of times a read operation to the HSM was attempted. The HSM can defer a read if it has no replies at the time, but expects some to be available later. Typically the HSM reports HostReadCount in two places: the number under ModuleJobStats counts a deferred read twice, once when it is initially deferred, and once when it finally returns some data. The number under ModulePCIStats counts this as one operation.
HostReadErrors	The number of times a read to an HSM failed because the parameters supplied with the read were incorrect. A nonzero value here typically indicates some problem with the host interface or device driver.
HostReadEmpty	The number of times a read from the HSM returned no data because there were no commands waiting for completion. In general, this only happens infrequently during HSM startup or reset. It can also happen if PauseForNotifications is disabled.
HostReadUnderruns	Not currently reported by the HSM.
HostReadDeferred	The number of times a read operation to the HSM was suspended because it was waiting for more replies to become available. When the HSM is working at full capacity, a sizeable proportion of the total reads are likely to be deferred.

ID	Value
HostReadTerminated	The number of times an HSM had to cancel a read operation which has been deferred. This normally happens only if the clear key is pressed while the HSM is executing commands. Otherwise it might indicate a device driver, interface, or firmware problem.
PFNIssued	The number of PauseForNotifications commands accepted by the HSM from the hardserver. This normally increases at a rate of roughly one every two seconds. If the hardserver has this facility disabled (or a very early version), this does not occur.
PFNRejected	The number of PauseForNotifications commands rejected by the HSM when received from the hardserver. This can happen during HSM startup or reset, but not in normal use. It indicates a hardserver bug or configuration problem.
PFNCompleted	The number of PauseForNotifications commands that have been completed by the HSM. Normally, this is one less than the PFNIssued figure because there is normally one such command outstanding.
ANIssued	The number of Asynchronous Notification messages issued by the HSM to the hardserver. These messages indicate such things as the clear key being pressed and the HSM being reset. In later firmware revisions inserting or removing the smartcard or changing the non-volatile memory also generate asynchronous notifications.
ChanJobsIssued	The number of fast channel jobs issued to the HSM. The fast channel facility is unsupported on current HSMs. This number should always be 0.
ChanJobsCompleted	The number of fast channel jobs completed by the HSM. The fast channel facility is unsupported on current HSMs. This number should always be 0.
CPULoadPercent	The current processing load on the HSM, represented as a number between 0 and 100. Because an HSM typically contains a number of different types of processing resources (for example, main CPU, and RSA acceleration), this figure is hard to interpret precisely. In general, HSMs report 100% CPU load when all RSA processing capacity is occupied; when performing non-RSA tasks the main CPU or another resource (such as the random number generator) can be saturated without this statistic reaching 100%.
HostIRQs	On PCI HSMs, the total number of interrupts received from the host. On current HSMs, approximately equal to the total of HostReadCount and HostWriteCount.
ChanJobErrors	The number of low-level (principally data transport) errors encountered while processing fast channel jobs. Should always be 0 on current HSMs.
HostDebugIRQs	On PCI HSMs, the number of debug interrupts received. This is used only for driver testing, and should be 0 in any production environment.
HostUnhandledIRQs	On PCI HSMs, the number of unidentified interrupts from the host. If this is nonzero, a driver or PCI bus problem is likely.

ID	Value
HostReadReconnect	On PCI HSMs, the number of deferred reads that have now completed. This should be the same as HostReadDeferred, or one less if a read is currently deferred.
ObjectsCreated	The number of times a new object has been put into the object store. This appears under the HSM's ModuleObjStats node.
ObjectsDestroyed	The number of items in the HSM's object store that have been deleted and their corresponding memory released.
ObjectCount	The current number of objects (keys, logical tokens, buffers, SEE Worlds) in the object store. This is equal to ObjectsCreated minus ObjectsDestroyed. An empty HSM contains a small number of objects that are always present.
CurrentTempC	The current temperature (in degrees Celsius) of the HSM main circuit board. First-generation HSMs do not have a temperature sensor and do not return temperature statistics.
MaxTempC	The maximum temperature recorded by the HSM's temperature sensor. This is stored in non-volatile memory, which is cleared only when the unit is initialized. First-generation HSMs do not have a temperature sensor and do not return temperature statistics.
MinTempC	The minimum temperature recorded by the HSM's temperature sensor. This is stored in non-volatile memory, which is cleared only when the unit is initialized. First-generation HSMs do not have a temperature sensor and do not return temperature statistics.
MemTotal	The total amount of RAM (both allocated and free) available to the HSM. This is the installed RAM size minus various fixed overheads.
NVMFreeSpace	The total amount of free space in the NVRAM of the HSM, in bytes.
NVMWearLevel	The wear level of the HSM's NVRAM, expressed as a percentage of the ratio between the erase count and the endurance.
NVMWornBlocks	The percentage of worn blocks in the NVRAM of the HSM.

21.3. 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 KeyIDs 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.

21.4. Logging and debugging of platform services

If problems are encountered when installing or commissioning an nShield HSM which prevent the `ncoreapi` service from running it will not be possible to access many of the tools above.

In this situation `hsmdiagnose` may help identify network and hardware issues that are preventing the system from starting.

If the platform services are running, information can be retrieved from logs stored within the HSM by `hsmadmin logs`. This command may also be used to clear the logs once the problem has been resolved.

`hsmadmin logs` has an option to write the log to an XML file. You should always write the log to a file before clearing the log so that the historic logs are available for reference.

22. Hardserver configuration files

The default location of the hardserver configuration file is `/opt/nfast/kmdata/config/config`.

The hardserver configuration file has the following sections that you can update to configure the hardserver on an nShield module. If a section is not present, it is assumed to have no entries.

22.1. Hardserver configuration files

Hardserver configuration files are text files. They must contain only characters with ASCII values between 32 and 127, and the tab, line break, and return characters.

Lines starting with a # character are comments and are ignored. Some comments that document the configuration options are generated by the configuration process. You can add your own comments, but in some cases they may later be overwritten.

A hardserver configuration file begins with a single line that specifies the version of the file syntax. This syntax-version line has the format:

```
syntax-version=n
```

In this syntax-version line example, *n* represents the version of the syntax in which the file is written. The system can process a file with a lower syntax version than the one it uses, but not one with a higher version.

After the syntax-version line, the rest of the configuration file consists of sections that can be edited to control different aspects of hardserver behavior. Each section begins with its name in square brackets, as in this example:

```
[slot_imports]
```

You can update the parameters defined in most of these sections to configure the way that the hardserver handles secure transactions between modules connected to the host computer and applications that run on the host computer.

Some sections are updated automatically and should not be edited manually. For more information, see the descriptions of individual sections.

In each section, the bracketed name is followed by a specified set of fields. Each field is on a separate line. Each field begins with its name, followed by an equals sign (=) and a value of

the appropriate type. White space can be included at either end of the line (for example, in order to indent lines as an aid to clarity).

Some types of field are grouped into entries. An entry is a set of fields of different types that define an instance of an object (for example, a particular client as distinct from other clients). Entries in the same section are separated by a line that contains one or more hyphens (`-`). Blank lines and comments are allowed between the fields in an entry.

Strings are case sensitive in the section names and field names.

If a particular section is not present in the configuration file, it is assumed to have no entries.

22.2. General hardserver configuration settings

22.2.1. server_settings

The `server_settings` section defines the settings for the client hardserver you can modify while the hardserver is running.

These flags are used by the `NFLOG_DETAIL` environment variable (see Environment variables to control logging).

The section contains the following fields:

Field	Description
loglevel	This field specifies the level of logging performed by the hardserver. It takes a value that is one of the following: • info • notice • client • remoteserver • error • serious • internal • startup • fatal • fatalinternal The default is info. For more information, see Logging, debugging, and diagnostics. If the NFAST_SERVERLOGLEVEL environment variable is set, it overrides any loglevel value set in the configuration file. NFAST_SERVERLOGLEVEL is a legacy debug variable.
logdetail	This field specifies the level of detail logged by the hardserver. You can supply one or more flags in a space-separated list. For more information about the flags, see the table below.
connect_retry	This field specifies the number of seconds to wait before retrying a remote connection to a client hardserver. The default is 10.
connect_maxqueue	This field specifies the maximum number of jobs which can be queued on the hardserver. The default is 4096: this is also the maximum value. Setting connect_maxqueue to a high value allows high throughput, but may cause long latency if the hardserver goes down.
connect_broken	This field specifies the number of seconds of inactivity allowed before a connection to a client hardserver is declared broken. The default is 90.
connect_keepalive	This field specifies the number of seconds between keepalive packets for remote connections to a client hardserver. The default is 10.
accept_keepidle	This field specifies the number of seconds before the first keepalive packet for remote incoming connections. The default is 30. Ideally, accept_keepalive should be at least twice the value of the connect_keepalive setting on the unattended machines.

Field	Description
accept_keepalive	This field specifies the number of seconds between keepalive packets for remote incoming connections. The socket will be closed after up to ten consecutive probe failures. The default is 10. Ideally, accept_keepalive should be a value such that (10 * accept_keepalive) > connect_broken on the unattended machine. Using the default values for both these fields will fulfil this requirement.
connect_command_block	When the module has failed, this field specifies the number of seconds the hardserver should wait before failing commands directed to that module with a NetworkError message. For commands to have a chance of succeeding after the module has failed this value should be greater than that of connect_retry. If it is set to 0, commands to a module are failed with NetworkError immediately, as soon as the module. The default is 35.
max_pci_if_vers	This field specifies the maximum PCI interface version number. If max_pci_if_vers is set to 0 (the default), there is no limit.
enable_remote_mode	If this field is set to yes (the default value) in the module configuration file, nShield HSM mode changing using the nopclearfail utility is enabled. If set to no, mode changing using nopclearfail is disabled.  Do not set enable_remote_mode in the client configuration file.
enable_remote_reboot	If this field is set to yes (the default value) in the module configuration file, the nShield HSM remote reboot using the nopclearfail is enabled. If set to no, remote reboot using nopclearfail is disabled. Run cfg-pushnethsm to push the new config file to the module.
enable_remote_upgrade	If this field is set to yes (the default value) in the module configuration file, the nShield HSM remote upgrade using the nopclearfail is enabled. If set to no, remote upgrade using nopclearfail is disabled. Run cfg-pushnethsm to push the new config file to the module.

These flags are those used by the NFLOG_DETAIL environment variable (see Environment variables to control logging).

You can supply a number of flags with the logdetail field, which specifies the level of detail logged by the hardserver (see the table above). Supply the flags in a space separated list:

Flag	Description
external_time	This flag specifies the external time (that is, the time according to your machine's local clock) with the log entry.
external_date	This flag specifies the external date (that is, the date according to your machine's local clock) with the log entry.

Flag	Description
external_tid	This flag specifies the external thread ID with the log entry.
external_time_t	This flag specifies the external time_ti (that is, the time in machine clock ticks rather than local time) with the log entry.
stack_backtrace	This flag specifies the stack backtrace with the log entry.
stack_file	This flag specifies the stack file with the log entry.
stack_line	This flag specifies 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_severity	This flag specifies the message severity (a severity level as used by the NFLOG_SEVERITY environment variable) with the log entry.
msg_categories	This flag specifies the message category (a category as used by the NFLOG_CATEGORIES environment variable) with the log entry.
msg_writeable	This flag specifies message writeables, and extra information that can be written to the log entry, if any such exist.
msg_file	This flag specifies the message file 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	This flag specifies 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.
options_utc	This flag showing the date and time in UTC (Coordinated Universal Time) instead of local time.
unix_file_descriptor_max	This field sets the number of file descriptors the hardserver must be capable of having open concurrently on Linux. The value must be an integer. If unix_file_descriptor_max is set to **0** (the default), the value will be ignored by the hardserver. If it is set to a positive value, the hardserver will refuse to start if the file descriptor hard limit on the system is less than that value. This configuration entry can be used to ensure that the hardserver is capable of satisfying the maximum number of hardserver connections that applications may make use of.

22.2.2. server_performance

The server_performance section defines the performance settings for the client hardserver. These are read only at hardserver start-up.

This section contains the following fields:

Field	Description
`enable_scaling`	This field determines whether multi-threaded performance scaling is enabled or not. If this field is set to **auto** (or not set), the hardserver automatically chooses the best option for the available hardware (enabled when using an nShield network-attached HSM, for which scaling is currently optimized, and disabled if using an nShield PCIe or USB-attached HSM). It can explicitly be enabled by setting to **yes**, and explicitly disabled by setting to **no**.
`target_concurrency`	This field allows the level of concurrency to be tuned. The value must be an integer and will only come into effect when multi-threaded performance scaling is enabled. If target_concurrency is set to **0** (the default), the value will be automatically configured by the hardserver based on the available number of physical CPU cores. The target concurrency configured is written to the hardserver log.

22.2.3. module_settings

The `module_settings` section defines the settings for the module that can be changed while the hardserver is running.

The section contains the following fields:

Field	Description
`esn`	This field specifies the electronic serial number of the module.
`priority`	This field specifies the priority of the module. The value for this field can be an integer from 1 (highest) to 100 (lowest). The default is 100.

22.2.4. server_remotecomms

The `server_remotecomms` section defines the remote communication settings for the client hardserver. These are read only at hardserver start-up.

This section contains the following fields:

Field	Description
`impath_port`	This field specifies the port on which the hardserver listens for incoming impath connections. The default is 9004. Setting this field to `0` specifies that the hardserver does not listen for incoming connections. Ensure that firewall settings are consistent with port settings. See the Installation Guide for more information about firewall settings.

22.2.5. server_startup

The `server_startup` section defines the settings for the hardserver that are loaded at start-up. Any changes you make to the settings in this section do not take effect until after you restart the hardserver. For more information, see Stopping and restarting the hardserver.

The section contains the following fields:

Field	Description
`unix_socket_name`	This field specifies the name of the socket to use for non-privileged connections on Linux. The default is `/dev/nfast/nserver`. If the `NFAST_SERVER` environment variable is set, it overrides any value set for `unix_socket_name` in the hard server configuration file. For more information about environment variables, see Environment variables.
`unix_privsocket_name`	This field specifies the name of the socket to use for privileged connections on Linux. The default is `/dev/nfast/privnserver`. If the `NFAST_PRIVSERVER` environment variable is set, it overrides any value set for `unix_privsocket_name` in the hardserver configuration file.
`nt_pipe_name`	This field is not used on Linux.
`nt_pipe_users`	This field is not used on Linux.
`nt_privpipe_name`	This field is not used on Linux.
`nt_privpipe_users`	This field is not used on Linux.
`nonpriv_port`	This field specifies the port on which the hardserver listens for local non-privileged TCP connections. The value `0` (which is the default) specifies none. Java clients default to connecting to port 9000. Ensure that your network firewall settings are correct. See the Installation Guide for more information about fire wall settings. If the `NFAST_SERVER_PORT` environment variable is set, it overrides any value set for `nonpriv_port` in the hardserver configuration file.
`priv_port`	This field specifies the port on which the hardserver listens for local privileged TCP connections. The value `0` (which is the default) specifies none. Java clients default to connecting to port 9001. If the `NFAST_SERVER_PRIVPORT` environment variable is set, it overrides any value set for `priv_port` in the hardserver configuration file.

22.2.6. load_seemachine

The `load_seemachine` section of the hardserver configuration file defines SEE machines that the module should load and, if required, start for use by other clients. Each SEE machine is defined by the following fields:

Field	Description
`module`	This field specifies the module on to which to load the SEE machine. The value must be an integer. A module with this ID must be configured on the client computer.
`machine_file`	This field specifies the file name of the SEE machine.
`userdata`	This field specifies the `userdata` file name to pass to the SEE machine on start-up. If this field is blank (`" "`), the SEE machine is loaded but not started. By default, this field is blank.
`worldid_pubname`	This field specifies the `PublishedObject` name to use for publishing the `KeyID` of the started SEE machine. If this field is blank (`" "`), the `KeyID` is not published. This field is ignored if the value of the `userdata` field is blank.
`postload_prog`	This field specifies the program to run after loading the SEE machine in order to perform any initialization required by the SEE machine or its clients. The specified program must accept an argument of the form `-m module#`. To run `see-sock-serv` directly on the nShield HSM, set this field to `sockserv`.
`postload_args`	This field specifies arguments to pass to the program specified by the `postload_prog` field. The argument `-m module#` is automatically passed as the first argument. The `postload_args` field is ignored if `postload_prog` is not specified or is blank. To run `see-sock-serv` directly on the nShield HSM, set this field to `-p` *`pubname`*.
`pull_rfs`	This field specifies whether the SEE machine name and userdata should be pulled from the RFS. The default is `0`: set to `1` to pull the SEE machine and user data from the RFS before loading on the remote module. This field will be ignored if set on client machine configurations.  This field will not be added to existing configuration files if you are upgrading an image. If you require the new functionality enabled by this field, you can add the field to the `load_seemachine` section of your existing configuration file.

22.2.7. slot_imports

The `slot_imports` section defines slots from remote modules that will be available to the local computer. Each slot is defined by the following fields:

Field	Description
`local_esn`	This field specifies the ESN of the local module importing the slot.

Field	Description
local_slotid	This field specifies the SlotID to use to refer to the slot when it is imported on the local module. The default is 0, and provides automatic assignment to the lowest available slotID after any configured dynamic slots.
remote_ip	This field specifies the IP address of the machine that hosts the slot to import.
remote_port	This field specifies the port for connecting to the nShield HSM.
remote_esn	This field specifies the ESN of the remote module from which to import the slot.
remote_slotid	This field specifies the SlotID of the slot to import on the remote module. The value of this field must be an integer. The default is 0.

22.2.8. slot_exports

The slot_exports section defines the slots on local modules that the local hardserver should allow network modules to import. Each local slot has an entry for each remote module that can import it, consisting of the following fields:

Field	Description
local_esn	This field specifies the ESN of the local module whose slot can be imported by a network module.
local_slotid	This field specifies the SlotID of the slot that is to be imported. The value must be an integer. The default is 0.
remote_ip	This field specifies the IP address of the module that is allowed to import the slot. Use 0.0.0.0 to allow all machines. The default is 0.0.0.0
remote_esn	This field specifies the ESN of the module allowed to import the slot. Leave the value blank to allow all permitted modules in the Security World. The default is blank.

22.2.9. dynamic_slots

The dynamic_slots section defines the number of Dynamic Slots that each HSM is to support for the Remote Administration Service.

Field	Description
esn	ESN of the HSM to be configured with Dynamic Slots.
slotcount	The number of Dynamic Slots that the HSM is to support. If set to 0 (default) the HSM does not support the Remote Administration Service.

22.2.10. slot_mapping

The `slot_mapping` section defines, for each specified HSM, a slot that is exchanged with slot 0 of the HSM. Slot 0 becomes a Dynamic and/or Remote Slot and the local slot becomes the specified slot number. This enables applications and utilities that only support slot 0 to use Remote Administration and Remote Operator.

Field	Description
`esn`	ESN of the HSM to which the mapping is applied.
`slot`	The slot number to be swapped with slot 0, so that: • Slot 0 refers to a Dynamic and/or Remote Slot • The specified slot number refers to the local slot of the HSM. If `slot` is set to 0 (default) there is no slot mapping.

22.2.11. dynamic_slot_timeouts

The `dynamic_slot_timeouts` section defines timeout values that are used to specify expected smartcard responsiveness for all HSMs associated with the relevant host or client, when using the Remote Administration.

Field	Description
`round_trip_time_limit`	`round_trip_time_limit` > 5s + network latency time Round trip (HSM to smartcard and back) time limit in seconds. The card is regarded as removed, if no response has been received within the allowed time. Expected network delays need to be taken into account when setting this. The default is ten seconds.
`card_remove_detect_-time_limit`	`card_remove_detect_time_limit` >= `round_trip_time_limit` *2 Maximum number of seconds that can pass without a response from the smart card, before it is regarded as removed and all the keys that it protects are unloaded. Lower values increase network traffic. The default is 30 seconds.

22.2.12. audit_logging

The `audit_logging` section defines the settings for for the syslog infrastructure used by the audit logging capability. These values require a restart of the hardserver to be recognized.

Field	Description
auditlog_port	This field specifies the UDP port to which audit log syslog messages should be delivered. The default is 514.
auditlog_addr	This field specifies the IP address of the machine that hosts the syslog server to which audit logging syslog messages shoud be sent.
auditlog_copy_hslog	This field specifies if audit logging sylog entries should be copied into the hard server log as well as being transmitted to the syslog server. The default is off. It can be turned by setting to **yes**, **true**, **on** or **1**. Care should be taken when setting this field as this can cause the hardserver log to grow significantly.

22.3. Sections only in client configuration files

22.3.1. nethsm_imports

The nethsm_imports section defines the network modules that the client imports. It can also be set up by the nethsmenroll utility. Each module is defined by the following fields:

Field	Description
local_module	This field specifies the ModuleID to assign to the imported module. The value must be an integer. A module with this ID must not be already configured on the client computer.
remote_ip	This field specifies the IP address of the module to import.
remote_port	This field specifies the port for connecting to the nShield HSM.
remote_esn	This field specifies the ESN of the imported module.
keyhash	This field specifies the hash of the key that the module should use to authenticate itself.
privileged	The value in this field specifies whether the client can make a privileged connection to the module. The default is 0, which specifies no privileged connections. Any other value specifies privileged connections.
ntoken_esn	This field specifies the ESN of this client's nToken, if an nToken is installed.

The default value for remote_keyhash (40 zeros) specifies that no authentication should occur. We recommend that you set a specific key hash in place of this default.

22.3.2. rfs_sync_client

This section defines which remote file system the client should use to synchronize its key

management data:

Field	Description
remote_ip	The IP address of the RFS against which to synchronize.
remote_port	This field specifies the port for connecting to the RFS.
use_kneti	Setting this option to yes to use a local module KNETI instead of the default hardserver's software KNETI to authenticate this client to the RFS.
local_esn	This is only required if use_kneti is set to yes. It is the ESN of the local module used for authentication.
remote_keyhash	Software or module KNETI hash used to authenticate the RFS, or 40 zeroes to indicate no authentication required (default is 40 zeroes).
remote_esn	ESN of the remote module used to authenticate the RFS, or empty when using software KNETI authentication or no authentication required (default is empty).

22.3.3. remote_file_system

This section is updated automatically when the rfs-setup utility is run. Do not edit it manually.

The remote_file_system section defines a remote file system on the client by listing the modules allowed to access the file system on this client. Each module is defined by an entry consisting of the following fields:

Field	Description
remote_ip	This field specifies the IP address of the remote module that is allowed to access the file system on this client.
remote_esn	This field specifies the ESN of the remote module allowed to access the file system on this client.
keyhash	This field specifies the hash of the key with which the client must authenticate itself to the module. The default is 40 zeros, which means that no key authentication is required.
native_path	This field specifies the local file name for the volume to which this entry corresponds.
volume	This field specifies the volume that the remote host would access to use this entry.
allow_read	If this field is set to yes, it means that a remote server is allowed to read the contents of the file. The default is no.

Field	Description
allow_write	If this field is set to yes, it means that a remote server is allowed to write to the file. The default is no.
allow_list	If this field is set to yes, it means that a remote server is allowed to list the contents of the file. The default is no.
is_directory	If this field is set to yes, it means that this entry represents a directory. The default is no.
is_text	If this field is set to yes, it means that line endings should be converted to and from the Linux convention for transfers.

If you upgrade from an earlier software version to v12 and are using Remote Administration, you need to manually add the following sections to your configuration file.

22.3.4. remote_administration_service_slot_server_startup

The remote_administration_service_slot_server_startup section defines the communication settings that are applied at start-up to the Remote Administration Service.

Field	Description
port	Which port to use to connect to the Dynamic Slot Server. The default is 9005.

23. Cryptographic algorithms

23.1. Symmetric algorithms

Symmetric Algorithms				
Algorithm	**FIPS approved in a v1 or v2 Security World**	**FIPS approved in a v3 Security World**	**Key type**	**Supported by generatekey**
AES	Y	Y	AES or Rijndael	Y
Arcfour	N	N	Arcfour	N
ARIA	N	N	Aria	N
Camellia	N	N	Camellia	N
CAST 256	N	N	CAST256	N
DES	N	N	DES	N
DES2	Y	N	DES2	Y
Triple DES	Y	N [1]	Triple DES	Y
MD5 HMAC	N	N	HMACMD5	N
RIPEMD160 HMAC	N	N	HMACRIPEMD160	N
SEED	N	N	SEED	N
SHA-1 HMAC	Y	Y	HMACSHA1	Y
SHA-224 HMAC	Y	Y	HMACSHA224	N
SHA-256 HMAC	Y	Y	HMACSHA256	Y
SHA-384 HMAC	Y	Y	HMACSHA384	Y
SHA-512 HMAC	Y	Y	HMACSHA512	Y
Tiger HMAC	N	N	HMACTiger	N

[1] Not FIPS 140 approved for encryption operations, but available for decryption operations.

23.2. Asymmetric algorithms

Asymmetric Algorithms

Algorithm	FIPS approved in a v1 or v2 Security World	FIPS approved in a v3 Security World [1]	Key type	Supported by generatekey
Diffie-Hellman	Y	Y	DH or DHEx	Y
DSA	Y	Y	DSA	Y
ECDH	Y [2]	Y [2]	ECDH or EC [3]	Y
ECDSA	Y [4]	Y [4]	ECDSA or EC	Y
ECIES	N	N	ECDH or EC	N
Ed25519	N	N	Ed25519	N
ElGamal	Y	Y	DH	Y
KCDSA	N	N	KCDSA	N
RSA	Y	Y	RSA	Y
X25519	N	N	X25519	N

[1] Some insecure key sizes are non-FIPS 140 approved.

[2] FIPS 140 approval is only for use with ECDH keys, not with EC keys, but see [3] for exception.

[3] FIPS 140 allows an EC key to be used as an ECDH key for a single use-case. In this use case, an ECDH key is allowed to perform a single signing of a Certificate Signing Request (CSR), so that a certificate for the ECDH key can be generated.

[4] FIPS 140 approval is only for use with ECDSA keys, not with EC keys.

23.3. FIPS information

In a FIPS 140 Level 3 Security World, the nShield HSM only supports FIPS-approved algorithms and key sizes.

- If you have a FIPS 140 Level 3 Security World and have any protocols that use algorithms not approved by FIPS, you have the following options:
  - If you need to use these non-approved algorithms, you can migrate to a non-FIPS Security World but continue to use hardware and firmware validated for FIPS 140 Level 3.
  - If you have strict FIPS 140 Level 3 requirements, you must replace your protocols to use approved algorithms.

- If you have a FIPS 140 Level 3 Security World and have existing long-term keys for unapproved algorithms, you have the following options:
  - Migrate to a non-FIPS Security World but continue to use hardware and firmware validated for FIPS 140 Level 3.
  - Replace the keys with approved keys before upgrading to the current firmware. Keys for unapproved algorithms are incompatible with this Security World.

To obtain more details on the specific algorithms that are FIPS approved for use in the HSM, refer to the nShield Security Policy for the particular FIPS CMVP certified nShield product that you are using.

For the FIPS CMVP certificates for nShield products, see https://csrc.nist.gov/projects/cryptographic-module-validation-program/validated-modules/search. The FIPS CMVP certificate links to the Security Policy.

23.4. Compatibility of Security World versions with FIPS

To comply with the latest FIPS cryptographic transitions, Security World v3 was introduced in firmware version 12.50. If an nShield HSM is upgraded to use firmware version 12.50 or later, any v2 Security Worlds using the HSM that were compliant with FIPS 140 Level 3 will no longer be compliant.

You can create a v3 Security World that is compliant with FIPS 140 Level 3 from a host server if you meet the following criteria:

- The host server is running Security World host-side software version 12.50 or later.
- The HSM is running firmware version 12.50 or later.

Your solution is only FIPS 140 compliant if you are running the exact firmware version that has been FIPS 140 certified.

24. Audit Logging

Audit Logging on nShield HSMs provides the following features:

- Logs generated and signed on the nShield HSM
- Tamper detection
- Deletion Detection
- Administrative operations are logged
- Key lifetime events are logged
- Per key usage events are optionally logged
- Optional key usage logging
- Public key verification of audit logs
- Compatibility with syslog and Security Information and Event Management (SIEM).

24.1. Configuring Audit Logging

Audit Logging is enabled per Security World and is configured on creation of the Security World.

Prerequisites

- If the audit logs are to be sent to a non-default location, the configuration file must be set up before the Security World is created.
- The Real Time Clock (RTC) on the HSM must be set and the setting confirmed after cre ating the Security World or indoctrinating an HSM into the Security World. The RTC on the HSM is used to timestamp audit log entries.

24.1.1. Configure audit log transport through syslog

The Audit Logging entries are delivered over syslog using UDP transport. This transport must be configured before Audit Logging is enabled in order to collect the initial messages.

1. Check the syslog transport before creating an Audit Logging enabled Security World.

   Send a log message to the configured host and port using a command, for example **log ger**, that can send messages to a syslog server over UDP.

2. Set the Audit Log entries in the hardserver configuration file.

   ```
   [auditlog_settings]
   # Start of the auditlog_settings section
   ```

```
# Hardserver settings for audit logging.
# Each entry has the following fields:
#
# The port number Auditlogging server listens to .
auditlog_port=514
#
# IP Address of the Auditlogging server
auditlog_addr=WWW.XXX.YYY.ZZZ
#
# Copy auditlog to hardserver log. (default=no)
#  auditlog_copy_hslog=ENUM
```

`auditlog_port`	This is the UDP destination port for Audit Logging syslog messages. The default is 514.
`auditlog_addr`	This is the IP address of the host to which the Audit Logging syslog messages should be delivered. The default is 0.0.0.0.
`auditlog_copy_hslog`	This indicates that the syslog messages from Audit logging should be copied to the hardserver's log file. This provides some degree of redundancy but means that the hardserver's log file may grow significantly. The default is no.

3. Restart the hardserver to load the updated configuration file.

24.1.2. Create a Security World with Audit Logging enabled

For the overall procedure, see Creating a Security World using new-world.

Key considerations:

- A Security World is created with Audit Logging enabled if either the `--audit-logging` or `-G` options are passed to the `new-world` command or the Security World is created in the `common-criteria-cmts` mode. This requires that the HSM is capable of audit logging and Security World creation will fail if the HSM does not support Audit Logging.
- Additional HSMs are indoctrinated into an Audit Logging enabled Security World using the `new-world` command with the `--program` or `-l` options.
- The HSM must be capable of Audit Logging. If it is not capable the indoctrination will fail. Therefore all HSMs in an Audit Logging Security World are set to Audit Logging.

When you configure an Audit Logging Security World:

1. Audit Logging is set as enabled for this Security World and is recorded in the world file.
2. The HSM is initialized and:

- A flag indicating the Audit Logging status is recorded in the HSM's EEPROM
- A 3072bit DSA HSM specific Audit Logging Signing Key (KAL) is created and persisted in the HSM's EEPROM
- A Certifier Block containing the public half of KAL is generated and sent to the log receiver via the hardserver.

When you indoctrinate a new HSM into an Audit Logging Security World:

- The world file specifies that this is an Audit Logging Security World
- The same actions as for initializing an HSM when the Audit Logging Security World was created are performed. This means that:
  - All HSMs in an Audit Logging Security World have Audit Logging enabled
  - Each HSM has a distinct Audit Logging Signing Key.

24.1.3. Confirm the Audit Logging configuration

Check for `AuditLogging` on the `state` line in the output of `nfkminfo`.

Enabled `AuditLogging`

Disabled `!AuditLogging`

```
>nfkminfo
...
 state       0x37270009 Initialised Usable Recovery !PINRecovery !ExistingClient RTC NVRAM FTO
AlwaysUseStrongPrimes !DisablePKCS1Padding !PpStrengthCheck AuditLogging SEEDebug
...
```

Check `AuditLogging` on the `active modes` line in the output of `enquiry`.

Enabled `AuditLogging`

Disabled (AuditLogging is not listed)

```
>enquiry
...
mode                operational
...
active modes        AuditLogging UseFIPSApprovedInternalMechanisms AlwaysUseStrongPrimes
hardware status     OK
```

24.1.4. Disable Audit Logging

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

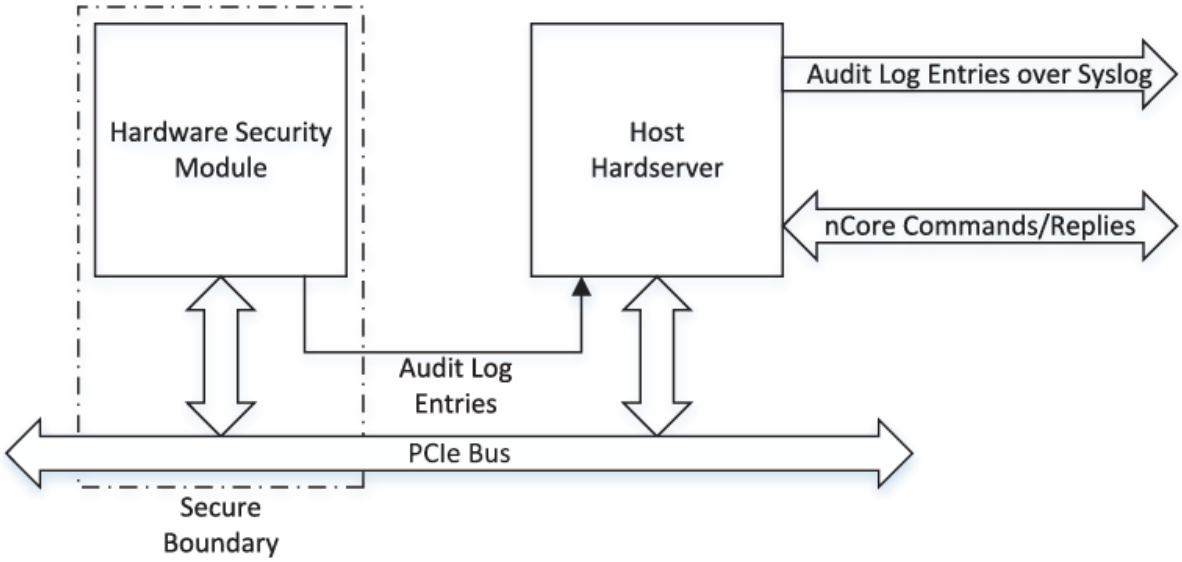
To disable Audit Logging on an HSM:

1. Remove that HSM from the Security World.
2. Reinitialize the HSM using `initunit`.

24.2. Audit Logging architecture

Audit Logging is implemented on the nShield HSM. The following image displays the nShield HSM implementation.

The audit log entries are generated on the module, the hardserver acts as a relay to a syslog infrastructure.



The hardserver layer implements a higher-level abstraction which is presented to application clients. The information used to manage this such as Client Identifiers etc. is not available to the HSM and therefore cannot be logged.

24.2.1. Audit Logging implementation

The Audit logging functionality is based on that described in RFC-5848 (https://tools.ietf.org/html/rfc5848). This describes a mechanism also known as syslog-sign that adds the following capabilities to syslog:

- Origin authentication
- Public verification
- Message integrity

- Replay resistance
- Message sequencing
- Detection of missing messages.

It is implemented on top of a standard syslog data stream and does not use any additional protocol. The syslog-sign logging scheme adds a number of control messages to the log entries that are to be audited. These are also implemented as syslog messages. The following sections outline the audit log entries that are present in the syslog data stream for nShield Audit Logging. The signing mechanism used is DSA with a 3072 bit key and SHA-256 as the hashing mechanism.

Audit log entry

This is the log message from the entity being audited. It is in a standard format and includes operation specific data required to provide an auditing capability. As each log message is generated on the HSM, a digest operation is performed on it and the digest buffered in the HSM.

Signature Block

When sufficient digests have been accumulated (*N*), a Signature Block is generated as a standard log entry containing the following:

- Digests of the previous *N* messages
- Information to allow the digests to be matched with their respective log entries
- A signature across the digests and other information.

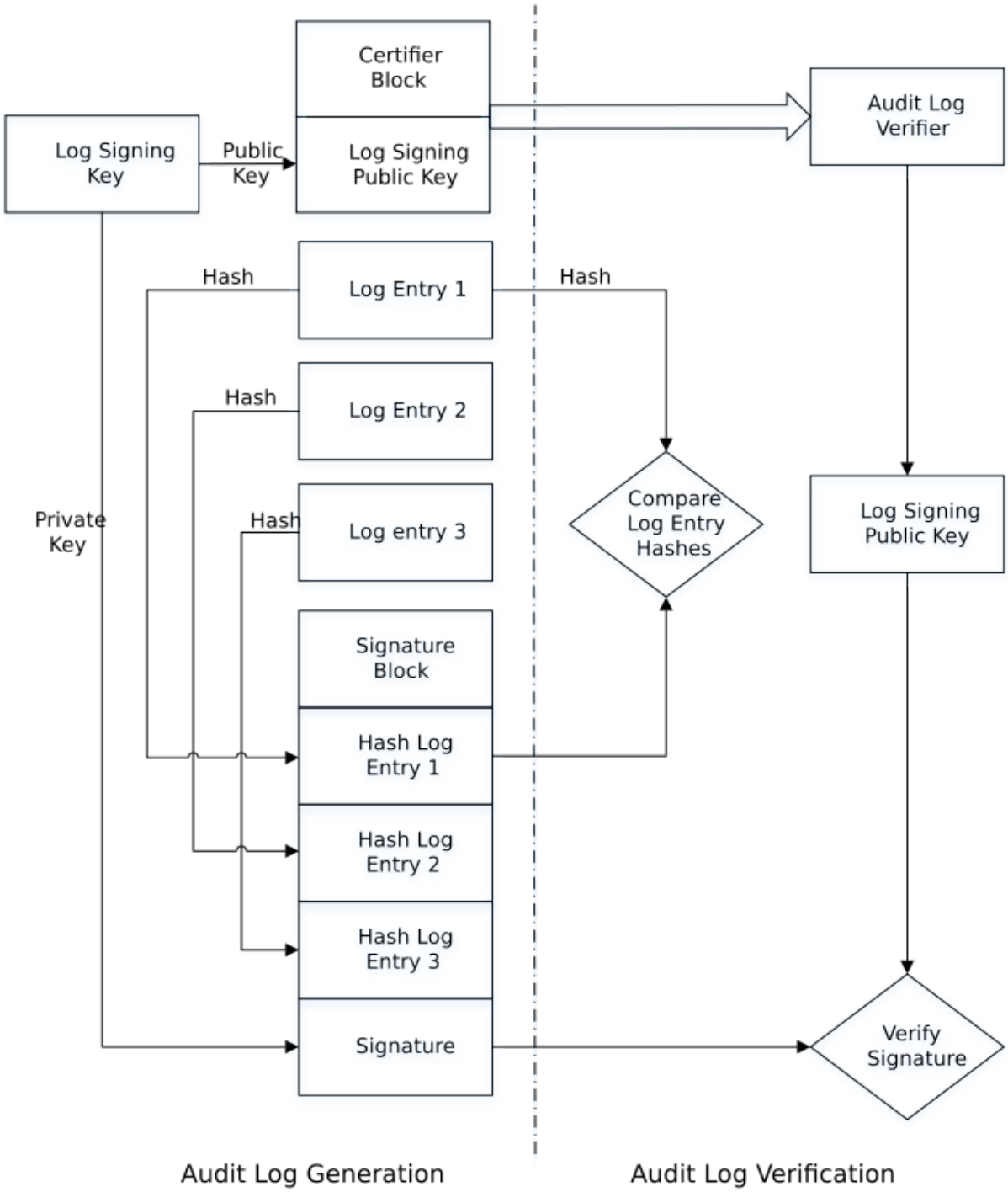
The number of messages is dictated by the transport medium, the size of the digests, the size of the signature and the size of other data contained in the message. There is a limit to the size of messages that can be transported over syslog. The signature is performed using a log signing key. This key is generated and the private half is held securely in the HSM.

Certifier Block

Verification of the Signature Blocks requires that you, or the application performing the ver ification, has access to the public half of the log signing key. The Certifier Block provides a mechanism for the log verifier to get access to this key. The key is packaged in a form allow ing the source of the audit logs to be verified. As the size of this information may be too large for the syslog transport medium it can be broken down into Certifier Block Fragments which are compatible with the syslog transport mechanism. When all of these fragments are received by the log verifier, it can reconstruct the public half of the log signing key and perform any consistency checks and origin verification that is needed.

24.2.2. Audit Log Verification process

Given the public half of the log signing key, a Signature block and its corresponding log entries, the verifier can check the signature on the Signature Block. When this is verified, the log entry digests in the Signature Block are implicitly verified. The integrity of the corresponding log entries can be verified by performing a digest on received log entries and comparing them to the corresponding verified digests in the Signature block. The image below shows the basics of this approach. For more information, see Audit Log Verification.



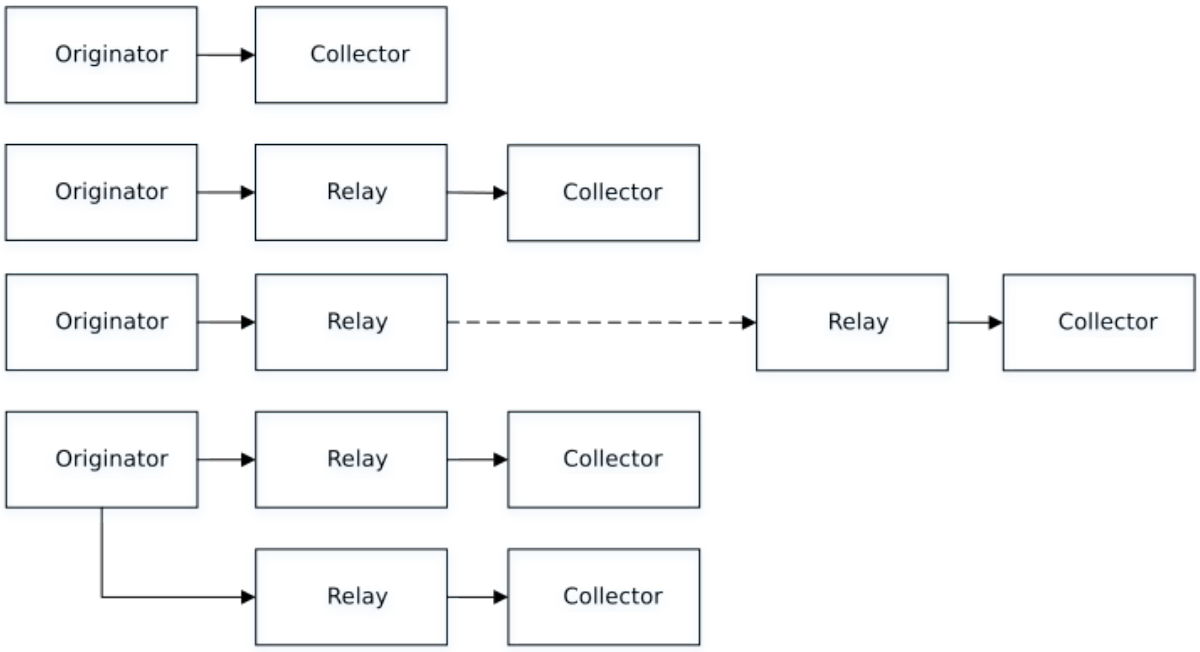
24.2.3. Log distribution

The nShield Audit Logging capability uses the RFC-3164 (https://tools.ietf.org/html/rfc3164) standard for distributing audit log messages. All audit log messages will have the following header prepended. This header is applied by the hardserver before sending the message and does not form part of the signed audit log messages. The signed portion of the audit log message starts at the `CEF:0` CEF identifier and continues to the end of the message.

```
<134>MMM DD HH:MM:SS hostname CEF:0......
```

The PRI value of this header `<134>` indicates the facility `local0` and a severity of `info`.

The syslog infrastructure used for Log distribution is out of the scope of this guide and your responsibility to implement. Log distribution for Audit Logging uses syslog as the transport medium which allows a standard protocol and message format to be used for the Audit Log ging messages. This transport is compatible with SIEM systems and the wider syslog infrastructure in an organization can be used to further distribute or process the log stream. There are many possible configurations of syslog deployment, as shown below.



24.3. Configuring audit log distribution

The actual implementation of the syslog infrastructure is at your discretion. Verification of the log messages requires that the verifying application has access to the audit logs from the HSMs in the Security World. The example verifier for the nShield Audit logging facility described in Audit Log Verification processes a file containing the audit log messages. It

can process audit log messages from a specific HSM identified by its ESN or will use the first ESN found in the file.

It is recommended that logs from the nShield Audit Logging facility are separated from those from other applications. This can be accomplished by using the information in the audit log messages described in the section on Log Format. There are a number of entries that can be used to separate out the messages from the nShield Audit Logging facility. These include:

- Identifying elements in the CEF header:
  - Device Vendor
  - Device Product
- Identifying elements in the syslog header
  - Hostname or IP address of the machine hosting the hardserver which is distributing the audit log messages
- Using a distinct port for nShield Audit Logging see configuring syslog.

The log messages can be further split into those from specific HSMs using the ESN in the audit log messages.

As an example, the following `rsyslog` configuration will direct all messages with the string *nCipher Security* to a specific log file:

```
:msg, contains, "nCipher Security" /var/log/hsmauditlog
```

A similar strategy can be used with `syslog-ng`:

```
destination d_auditlog { file("/var/log/hsmauditlog"); };
filter f_auditlog { match("nCipher Security" value("MESSAGE")); };
log { source(s_log); filter(f_auditlog); destination(d_auditlog); }
```

Adjust the example paths for the platform running your syslog server as appropriate. `s_log` is the source receiving the audit logging messages.

Refer to the documentation for your syslog implementation for information on processing and distributing log messages.

24.4. Configuring the syslog message infrastructure

It is important that the syslog infrastructure does not attempt to rewrite the log messages as this will affect the ability of the Audit Logging process to verify log messages. For example, the `rsyslog` default RFC-3164 parser will rewrite log messages interpreting the **CEF:0**

as a tag and will write **CEF: 0** to the log file. This means that an Audit Logging message persisted by the default RFC-3164 parser cannot be verified as Audit Logging signs the log message starting at **CEF:**. You must configure your syslog infrastructure to preclude the signed part of the audit log message.

24.4.1. rsyslog

rsyslog can be configured to not reformat messages using the following approach:

1. Define a formatting template as shown below in the `/etc/rsyslog.conf` file. This should be added in the `##### MODULES #####` section of the rsyslog configuration file.

   ```
   $template myFormat,"%rawmsg%\n"
   ```

2. Apply this formatting template to the processing of Audit Logging messages. For this example it is assumed that messages containing *nCipher Security* will be persisted in the `/var/log/hsmauditlog` file. You can use any other selection mechanism such as storing messages for a particular HSM as identified by its ESN in separate files.

   ```
   :msg, contains, "nCipher Security"        /var/log/hsmauditlog;myFormat
   ```

3. If the rsyslog server is going to be used as a relay, then the format needs to be applied to any relay statements in the rsyslog configuration file and to any receivers of the syslog message.

24.4.2. syslog-ng

`syslog-ng` does not appear to rewrite messages in the same way as `rsyslog`. Refer to the *syslog-ng* documentation for information on formatting.

24.5. Audit log format

24.5.1. CEF format

The audit log entries are emitted from the HSM in CEF format. This provides both human readable log messages and compatibility with SIEM applications. As indicated in the previous section the audit log entries are distributed using the syslog transport mechanism.

A CEF format log message is shown below:

```
CEF:Version|Device Vendor|Device Product|DeviceVersion|Device Event Class ID|Name|Severity|[Extension]
```

Parameter	Description
`CEF:Version`	This is mandatory and Version is currently **0.**
`Device Vendor`	This is nCipher Security
`Device Product`	This identifies the family of nShield HSMs: • nShield Solo • nShield Solo XC • nShield Edge • nShield 5s
`Device Version`	This is the firmware version, for example *12.80.0*.
`Device Event Class ID`	This is an identifier for the type of message: **Class ID** / **Description** 1 — nCore Commands 2 — Internal HSM events: • Periodic heartbeat • Secure channel establishment 3 — Audit logging control messages: • Signature Blocks • Certifier Blocks 4 — Reserved 5 — Shutdown messages 6 — Reserved
`Name`	This is the event being logged. For Audit Logging, it is either the nCore command that is being logged, `Cmd_Destroy` for example, a description of the event such as `heartbeat` or one of `ssign` or `ssign-cert` which identifies either a Signature Block or a Certifier block.

Parameter	Description
`Severity`	This is an indication of the importance of the message. **Severity** / **Description** 1 — nCore Commands 2 — Internal HSM events: • Reboot events • Secure channel establishment 3 — nCore Commands that force a Signature Block flushing buffered message hashes. 4 — Periodic Heartbeat messages 5 — Audit Logging control messages: • Signature Blocks • Certifier Blocks 6 — Shutdown messages 10 — HSM Error messages

24.5.2. CEF extensions

The rest of the log message is made up of CEF extensions. These are name/value pairs that are used to convey specific information for the log message. The name-value pairs can be processed by SIEM applications such as Arcsight and can be displayed in tabular reports of the messages received. They can be used for filtering and further processing within the SIEM application. The following table specifies the meaning and format of the extensions used by the Audit Logging facility.

Extension Name	Description
`esn`	Electronic Serial Number (ESN) of the HSM in the following format: `XXXX-XXXX-XXXX`
`rtc`	Time-stamp from the HSM's Real Time Clock (RTC) as ms since the epoch (1970 Jan 01 00:00:00 UTC).
`outcome`	Outcome of the operation - **success** or **failure**
`hkey`	Identifying nCore key hash for the main key of the command being logged as a 40 character hex string

Extension Name	Description
hbase	Identifying nCore key hash for the base key of a Cmd_DeriveKey command being logged as a hex string
hwrap	Identifying nCore key hash for the wrap key of a Cmd_DeriveKey command being logged or the logical token hash for key blobbing operations as a hex string
hin3-5	Identifying nCore key hashes for the remaining keys of a Cmd_DeriveKey command being logged as hex strings
hknso	Identifying nCore key hash of Security Officer's key as a hex string
htok	Identifying nCore Logical Token hash as a hex string
shareindex	Index of share being operated on by Logical Token functions as a decimal number
sharesleft	Number of Logical Token shares left to read or write as a decimal number
tokenslot	Slot number for Logical Token operations as a decimal number
sharesneeded	Quorum required to reconstruct a Logical Token as a decimal number
sharestotal	Total number of shares for a Logical Token as a decimal number
timelimit	How many seconds after reassembly the Logical Token is usable for
shorthash	Short Logical Token hash used in Cmd_EraseShare and Cmd_ChangeSharePIN
hkm	Identifying nCore hash of module key KM
mode	Mode that a channel is opened in. One of **encrypt**, **decrypt**, **sign** or **verify**
source	Source of command. One of **host**, **SEE** or **internal**
flags	Flags supplied to Cmd_SetNSOPerms and Cmd_InitialiseUnitEx. Cmd_SetNSOPerms • AlwaysUseStrongPrimes • DisablePKCS1Padding • FIPSLevel3Enforcedv2 • CommonCriteriaCMTSRestrictions Cmd_InitialiseUnitEx • AuditLogging • UseFIPSApprovedInternalMechanisms
slotcount	Count of Dynamic Slots to be configured
slotid	Dynamic Slot to create association for

Extension Name	Description
prevrtc	The previous value of the HSM's RTC as ms since the epoch. Used to indicate previous value of the RTC before a Cmd_SetRTC timestamp or an event occurring before a restart
smartcardesn	ESN of smartcard used for Dynamic Slot operations
kmltype	Type of the Module Per-Initialization Signing Key (KML) set by Cmd_InitialiseUnit(Ex)
sos	Indication of the sos code

24.5.3. Infrastructure extensions

The Audit Logging Implementation requires a number of infrastructure CEF extensions to provide data necessary for the RFC-5848 based signed syslog approach used. Please refer to RFC-5848 for further details on these infrastructure extensions. These CEF extensions replace the RFC-5424 Structured Data used in the original scheme but have the same meaning.

24.5.4. Message and reboot counters

There are two counters that are sent with all Audit Logging command log messages. The Reboot Session ID is also sent with Certifier Block and Signature Block messages.

Counter	Description
seqNo	Log message sequence number as a decimal number. This a counter that has a range of 1 to 9999999999. When seqNo reaches 9999999999 it is reset to 1. It is incremented for every command log message sent. This is not part of RFC-5848 and has been added to provide a direct mechanism for detecting deleted or missing log messages. The sequence number is unique in the context of a reboot session. When rsid is incremented seqNo is reset to **1**.
rsid	Reboot Session ID as a decimal number. This a counter that has a range of 1 to 9999999999. When rsid reaches 9999999999 it is reset to 1. It is incremented every time the HSM is restarted and whenever the Global Block Counter (gbc) reaches its limit and is reset.

24.5.5. Certifier Block extensions

The following extensions are used in the Certifier Block where the CEF header name element is ssign-cert. The Certifier Block is used to distribute the log signing public key. It is

sent by the HSM when logging is enabled and every time the HSM is restarted. This provides for redundancy as any Certifier Block from an HSM that has been configured for Audit Logging will contain the same log signing public key. The Certifier Block can extend over multiple syslog messages. The extensions identified here allow the data of a Certifier Block to be rebuilt from multiple fragments. Sufficient fragments are sent in separate `ssign-cert` messages to rebuild the payload block. See Certifier Block example for the details of the data included in the certifier block.

Name	Description
`tpbl`	Total Payload Block Length. This is the length of all Certifier Block fragments.
`findex`	Index of this fragment (1 based) as a decimal number.
`flen`	Length of the fragment as a decimal number.
`frag`	Base 64 encoded Certifier Block fragment.
`sign`	DSA signature using KAL of the data in each fragment up to the `sign` extension. The DSA signature is DER encoded and then base64 encoded. It is present here to support consistency checking.

24.5.6. Signature Block extensions

The following extensions are used in the Signature Block where the CEF header name element is ssign. The Signature Block supports the verification of Audit Logging messages. The Signature Block is sized to fit within a syslog message which dictates the number of audit log messages it covers. This release supports a maximum of 10 audit log messages per Signature Block. The main data for the Signature Block is the SHA-256 hashes of the log messages covered by the block.

Name	Description
`gbc`	Global Block Counter as a decimal number. Count of signature blocks sent in this Reboot Session prior to this Signature Block. This is a decimal number that has a range of 1 to 9999999999. When `gbc` reaches 9999999999 it is reset to 0. At the same time the `rsid` is incremented and `seqNo` reset to 1. It is incremented after every Signature Block is sent.
`fmn`	First Message Number as a decimal number. First log message `seqNo` in this Signature Block.
`hcnt`	Count of log messages included in this Signature Block as a decimal number.
`hb`	The log message SHA-256 hashes base64 encoded and separated by the & character. The & character does not occur in base64 encoding and avoids SIEM issues with embedded spaces.

Name	Description
sign	DSA signature using KAL of the data in the Signature Block up to the sign extension. The DSA signature is DER encoded and then base64 encoded.

24.5.7. Example Audit Logging messages

This section shows example Certifier Block, Signature Block and Audit Logging messages and shows how the CEF extensions are used together.

24.5.7.1. Certifier Block example

This is an example Certifier Block produced after a reboot of the HSM. The log messages have been reformatted for display as each one can be up to 1024 characters long. The Reboot Session ID (rsid) is 8. There are five fragments in this example. The first four are 450 characters and the final 340 long for a total length of the payload of 2140 characters. The Event Class Id is 3 and the severity is 5 identifying these as infrastructure messages.

```
<14>Apr 17 18:38:29 myhost CEF:0|nCipher Security|nShield5|13.2.2|3|ssign-cert|5|esn=56FF-28CF-56D8 rsid=1
rtc=1650217101121 tpbl=2140 findex=1 flen=450
frag=uwAAAEQAAABDs/xJZqXKSK6WQErGC0r2i73PM/iIHghD+zAe6sGkJkVNDpZvVxbBVv5sEWHgz7xe+GeHp0UR1YwcjiWkuHkdawEAAEQAAAAK
Lly29RAAgN1+YrBlcHDo0qhOalcsuhNOMt4do8hdvLl1uzfC8qHHV/roeNd2VBB4kgA/BTzwYS9YVyUXzCU5nwAAAKwFAAAEAAAAAAAAAAMAAAACA
AAADwAAADU2RkYtMjhDRi01NkQ4AAANAAAAKXs34KGsx758FHFb0tb2JgcrXx4uAAAABgAAAAAAAABEAAAAxay7rP7+ev4sX4364CQ7QCiHYPKCeO
t5A/mZsMFdLkvXWvIgCKYLi/7HUkwOWbAq6svLMljYZaoXvjiYF/rtFO0AAABEAAAAkDQ78/aWuOJebyUg65fRP4v4qH3s8K57uO9u98QvH+TcZ8H
YCY sign=MEUCIQCY3eAJD5FMbhIAYwqrkJEGVzaQK9pkSqmlVI2FqqrrxQIgU4xu9yr6SMbNjN0TkVO/5EpJmMULLdmboMFlCOulJt0=

<14>Apr 17 18:38:29 myhost CEF:0|nCipher Security|nShield5|13.2.2|3|ssign-cert|5|esn=56FF-28CF-56D8 rsid=1
rtc=1650217101122 tpbl=2140 findex=2 flen=450
frag=yLxeQnj+Xn12TtnDJgciGuSbcAptHI89SK3DYAAAC7AAAADgAAAJp3SB8LpmqUrOh1GlOl+dzaSYDEAwAAAIABAADjwRLJlKVAir+HlVAUCW
ojKksMqGyWGhwhMoqYP8ldIy7bb3UVQBp6M+fxVpSFFrz3bfDgJQNh/13YCAY1+r1JYvEner7cnGatDIjnMgNqQPN6alqM787pMz3/eIq0L0xI8rV
y99F/foV6aFcJVCvxsjL9wIQOd4AhjIgTfPTiAEC4UTl5Eg9YkKnjZXizpTxhReSZVMjIM8Fu2sjcvzh1Q8POqYcEuU5sZhQbLVjUvRpou2HpgOTw
hcXW+X4gWpMlXsVkwV7F24j4Ax6eiyaSp1HCx4savMxcyA3cxwp7dTUJnFPVr8npfqp2H3ai3khSIefMc7d8gyajJnOLlJQMzf6O5HrlVeuixd6hB
dwd sign=MEUCIF+kGFpzmaxSBs2mH6kdxxAk4FhKQL75PCXmVt7bXQS0AiEArsEj7VWjlWmdADrbQCzOrOtNBuVcx3C7tQZzPcceY3g=

<14>Apr 17 18:38:29 myhost CEF:0|nCipher Security|nShield5|13.2.2|3|ssign-cert|5|esn=56FF-28CF-56D8 rsid=1
rtc=1650217101123 tpbl=2140 findex=3 flen=450
frag=B1Ku9rirlixkgEd+73tMVJ1FQz85aCWuRqJl04YB1YwFvZgvRXhHvzqLFeJZAUerKlLgIaZwDq1twoXzvHq88QcJdbr0i4+87VorPKkEjKtS
SGHOVkkHhoBC8uNgYXnTBxqcqCqpZl4whuiEBmJQLcwgAAAAg8rgckmo3ArobecQooPxQ9AjYbCmAoKOUTRi7grTzPyAAQAA3Bvuz+tQ1uh5LvuKM
LTtGDTTplG7ks6ZkL8b+F2UW37jfN3lap27oAZq1otU4FOP4EVvoMmNSdI4uzCPi7VgcI3AcIkdjZIwbpYf9XQwvFwMxYvdBPGhPtc/t8Lslgs97r
MkES4ZciNI/NwjKp0fW4kCiSBSUQUAUcp6vgqg2vVL9naqRHhXNRJuweaRtOO60z0mBkTgCnAvscdr2ymErrWDZArHosYXJZrXghjNmXvu+rS8GvT
vTc sign=MEUCIEJSIiI39lan+vXxaIfdG3GiH+JEBTmbxdvxpRCognyrAiEAv6NaGmEDWmwKAScDk3OS6zwp6/c+ywHbzm1LUN6F8Gc=

<14>Apr 17 18:38:29 myhost CEF:0|nCipher Security|nShield5|13.2.2|3|ssign-cert|5|esn=56FF-28CF-56D8 rsid=1
rtc=1650217101125 tpbl=2140 findex=4 flen=450
frag=189gfRjMpL5aBYYAkl1XqWDGHhFcltxTSzCMgWalxMOeOQxaZLbzYDtl2/udXIo0bn/PTgaOkPYTypvzFwsQM4axpDzVYCE064YVoyjUpgWB
U4kMlC4JuH5ytJAM+uA67xu36Iqx3j/mjMozTR1rGJuH+b314zgHRjvfr8AA0juiXn8tdxkFFRQlzYC9Ulw4g6fOSa06ecBIOA5Q0ylvdVjqmcX2+
+J+snC0wTxHV+vLKWh8m7/DDjExTXKpHo5EqyQB24tHCogAEAAPXoVMbwuAYleYDBkzxtkQ/79S5KmvCB8n9yQ2DBLlVf0msdJmP9EB93KYL3bifX
dq4ZhvaNtZ6zv1smErluGqEynASq/1KTz4XCVRCPY1UKBLitSl/izJLPwIiEjyRNrSBH88E+YHjkOvPDhG/e9v+w6Wj/xqR3yUCuvRraeTljUVNw4
RSr sign=MEUCIQCg8ikMnEC18uOdVHUeLpQt1HaGjKjRFzOT0RasPITmlwIgQkTZ+2yAy/DWNU3LKdkQdLYXLYdpKLuo+NICuhb+rS0=

<14>Apr 17 18:38:29 myhost CEF:0|nCipher Security|nShield5|13.2.2|3|ssign-cert|5|esn=56FF-28CF-56D8 rsid=1
rtc=1650217101126 tpbl=2140 findex=5 flen=340
frag=ewoGlc3Gv8gm9dih5IRR3MbsuAHgc9ngtW2zJtAp9V/E5jWi/j696Hlw70myararAqj/h7A1JJUEn3k2tVYwVFhlUQ+E6bgqliwfY7Rzp6M2
5+rwISSHdmbQvBB1KqQ/CtCjxCSrZeWB51jVeV7jqT7d6tNLpFbBlTGsX4y7GNiSokjCEjF8nWm2dHB/PEc5KBnIIKyShJ7X+ZHWfngBHrmAeKVk3
RYB9EjHMRLDc9nPJStMOXCxkWdyzv39KaBcdsUyfbJob6SUPugANJrBt1zEGbPdm4U9XttW7x61d4eTjZpQqxEXgQmny7TQDyJ3gu5slAwQgFZ3f6
```

```
oAAAA= sign=MEUCIQD12o6ZHYiLkHbq3Z2/jsCwWryL3l4SQYsFH5s/7OMbkQIgGX+qmmA+0h1GNaOY0kLjHkn6mnTm8fjTjL2MifY8r/8=
```

24.5.7.2. Log Messages and Signature Block

This example shows a sequence of audit log messages with a Signature Block after 10 messages. These are in the same `rsid` as the previous example. The log sequence number for this excerpt starts at 31 and the last log message before the Signature Block is sequence number 40. The name element identifies the command being executed by the HSM. Each of the example commands operates on an nCore Key and this is identified by the nCore key hash of the relevant key.

The Signature Block has name element `ssign` identifying it as a Signature Block. The `gbc` is 6 meaning this is the 7th Signature Block in this Reboot Session ID. The `fmn` is 31 and `hcnt` is 10 meaning that this Signature Block covers messages 31 to 40. As Audit Logs are generated this sequence will be repeated. Once this Signature Block has been received and with the log signing public key available the signature on this Signature Block can be verified and then the hashes of the individual log messages can be calculated and compared with the hashes recorded in the Signature Block for the corresponding log message to allow the detection of tampering.

```
<14>Apr 17 19:17:27 myhost CEF:0|nCipher Security|nShield5|13.2.2|1|Cmd_GenerateLogicalToken|1|esn=56FF-28CF-56D8
rsid=2 rtc=1650219439624 seqNo=302 source=host outcome=success htok=b70efe1c30dbbe98d70b6fff61ee718e69424620
sharesneeded=1 sharestotal=1 timelimit=0 hkm=72934cd31305e3c15fbbf8bd40a871801043b20b
<14>Apr 17 19:17:27 myhost CEF:0|nCipher Security|nShield5|13.2.2|1|Cmd_WriteShare|1|esn=56FF-28CF-56D8 rsid=2
rtc=1650219439625 seqNo=303 source=host outcome=success htok=b70efe1c30dbbe98d70b6fff61ee718e69424620
shareindex=0 tokenslot=1
<14>Apr 17 19:17:27 myhost CEF:0|nCipher Security|nShield5|13.2.2|1|Cmd_Destroy|1|esn=56FF-28CF-56D8 rsid=2
rtc=1650219439627 seqNo=304 source=host outcome=success hkey=b70efe1c30dbbe98d70b6fff61ee718e69424620
<14>Apr 17 19:17:27 myhost CEF:0|nCipher Security|nShield5|13.2.2|1|Cmd_LoadBlob|1|esn=56FF-28CF-56D8 rsid=2
rtc=1650219439627 seqNo=305 source=host outcome=success hkey=b504b47bc6963327671f44c09997dd124fe4cccd
hwrap=c2be99fe1c77f1b75d48e2fd2df8dffc0c969bcb
<14>Apr 17 19:17:27 myhost CEF:0|nCipher Security|nShield5|13.2.2|1|Cmd_Destroy|1|esn=56FF-28CF-56D8 rsid=2
rtc=1650219439628 seqNo=306 source=host outcome=success hkey=b504b47bc6963327671f44c09997dd124fe4cccd
<14>Apr 17 19:17:29 myhost CEF:0|nCipher Security|nShield5|13.2.2|1|Cmd_LoadBlob|1|esn=56FF-28CF-56D8 rsid=2
rtc=1650219441515 seqNo=307 source=host outcome=success hkey=47f7a0c26f178989951554442292202361 25b770
hwrap=c2be99fe1c77f1b75d48e2fd2df8dffc0c969bcb
<14>Apr 17 19:17:29 myhost CEF:0|nCipher Security|nShield5|13.2.2|1|Cmd_LoadBlob|1|esn=56FF-28CF-56D8 rsid=2
rtc=1650219441515 seqNo=308 source=host outcome=success hkey=1ab476f929918950fd41d29cc5ed4ec2c074dd23
hwrap=c2be99fe1c77f1b75d48e2fd2df8dffc0c969bcb
<14>Apr 17 19:17:29 myhost CEF:0|nCipher Security|nShield5|13.2.2|1|Cmd_LoadBlob|1|esn=56FF-28CF-56D8 rsid=2
rtc=1650219441515 seqNo=309 source=host outcome=success hkey=b504b47bc6963327671f44c09997dd124fe4cccd
hwrap=c2be99fe1c77f1b75d48e2fd2df8dffc0c969bcb
<14>Apr 17 19:17:29 myhost CEF:0|nCipher Security|nShield5|13.2.2|1|Cmd_LoadBlob|1|esn=56FF-28CF-56D8 rsid=2
rtc=1650219441515 seqNo=310 source=host outcome=success hkey=50e87ac66be97e8cfa4c289df56a4c172f4b7ac0
hwrap=c2be99fe1c77f1b75d48e2fd2df8dffc0c969bcb
<14>Apr 17 19:17:29 myhost CEF:0|nCipher Security|nShield5|13.2.2|1|Cmd_LoadBlob|1|esn=56FF-28CF-56D8 rsid=2
rtc=1650219441516 seqNo=311 source=host outcome=success hkey=44617dcb5a2d3a0e3711c334c92f1f69167cb2c2
hwrap=c2be99fe1c77f1b75d48e2fd2df8dffc0c969bcb
<14>Apr 17 19:17:29 myhost CEF:0|nCipher Security|nShield5|13.2.2|3|ssign|5|esn=56FF-28CF-56D8 rsid=2
rtc=1650219441516 gbc=31 fmn=302 hcnt=10
hb=NBBxStSc6L6aLcUknUSSoSeKHd4KEjfzd6z9lg1bu0E=&HYaUyW3IU1QzB6IoWQDByQ3QZufTJo7vkfbH5CZ7BbE=&w6pHxLQ3z4s78pfQEX0w
8T06uxubZ4/hDaCKJqWUr1M=&z3ndxekpyf+VWjKuNFXPIhHb3RhmbNXFFjDxyHc5S30=&ceOJGuIHuLEVi69yPMniwK4suAvkQIrzOmLeDYGCW7A
=&70VqW2Zka+oS6TW8cMGdeaJe4K4R9v7Vkn3h+aVAmr8=&DRDJUaIkdm2a4vUPI9aP5tUPble/9+5V1TIOVPVxGUE=&se+zKy0ll2n9DqoSvx8DU
I2kgoQcxWvnUHmCdArXChc=&9RHV8ohjo0zWJnYVlLeMrWGSQ7ZoVl+HorBPRH8yhIQ=&8MxoPVZCHtjq2rAZjTUU4N5DuhdPH2vWQZjGc3SpXQU=
```

```
sign=MEQCIFAzDFsJd78W4Zwgz7HJD9wZrDvqzLg6wr7yE/A/VTFMAiB9aAeTFsXfvJZRCA/2xLEO2mEkTWk2jt7F+GClxRJb/g==
```

24.6. Commands Audited

The Audit Logging facility generates log entries on the module for a set of nCore commands and module operations. The commands and information logged for each command are described in the following sections.

24.6.1. Key usage logging

By default the nShield Audit Logging Facility does not log usage of keys for cryptographic operations such as sign, verify, encrypt and decrypt or their usage in channels for these pur posed. Creation, Deletion and a number of other key operations are unconditionally logged by default. The Audit Logging feature provides the capability to optionally log these operations. This 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
```

In the following sections, the tables will indicate if this mechanism is required to generate a log message for a specific command or key.

24.6.2. Commands generating Audit Log messages

The following tables list the nCore commands that generate Audit Logging messages. For

each command they identify command specific data that is contained in the log message and the CEF extension used to identify it.

nCore Key and Logical Token hashes are the standard nCore identifying hashes. They are used to identify a key or logical token as it is an invariant for the key or logical token. These hashes are logged as lower-case hex encoding. In some cases a short hash may be presented. This is the first 10 bytes of the hash in a lower-case hex encoding.

For each command logged the command is specified by the `name` element of the CEF header. The other elements of the CEF header are filled as detailed in the previous section. All commands being logged will also include the following CEF extensions:

Extension	Description
`esn`	ESN of the HSM
`rsid`	Reboot Session ID
`rtc`	Timestamp as milliseconds since the epoch derived from the HSM's Real Time Clock
`seqNo`	Sequence number of the Audit Log message
`outcome`	Success or failure

Identifying Log Messages: As Audit Logging will potentially be running for a long time, the identification of a log message from an HSM based on the `rsid` and `seqNo` will not hold if the HSM is not restarted before the `seqNo` is reset when it reaches 9999999999. In this case account can be taken of the `gbc` as this will increment at a slower rate than the `seqNo`. Therefore messages in the same `rsid` with the same `seqNo` will have significantly different values of `gbc` (the mapping between `seqNo` and `gbc` is determined by the Signature Block containing the message in question). When the `gbc` is reset the `rsid` is incremented so counting begins in a new Reboot Session. Account can also be taken of the value of the `rtc`.

24.6.3. Key commands

Commands with **Yes** in the **Requires logkeyusage ACL** column will only be logged if the Key's ACL contains the `LogKeyUsage` Flag in the permission group authorizing the operation.

Command	Command Specific Information Logged	Extension	Requires logkeyusage ACL
Cmd_Sign	nCore key hash	hkey	Yes
Cmd_Encrypt	nCore key hash	hkey	Yes
Cmd_Decrypt	nCore key hash	hkey	Yes
Cmd_Verify	nCore key hash	hkey	Yes
Cmd_ChannelOpen	nCore key hash	hkey	Yes
	channel mode	mode	-
	Mode is one of encrypt, decrypt, sign, verify		
Cmd_Import	nCore key hash	hkey	No
Cmd_Export	nCore key hash	hkey	No
Cmd_Duplicate	nCore key hash	hkey	No
Cmd_GenerateKey	nCore key hash	hkey	No
Cmd_GenerateKeyPair	nCore key hash	hkey	No
	nCore Key Hashes of private and public key halves are identical		
Cmd_SetAppData	nCore key hash	hkey	No
Cmd_SetACL	nCore key hash	hkey	No
Cmd_Destroy	nCore key hash	hkey	No
	Also used for Logical Tokens		
Cmd_DeriveKey	nCore Key Hash of derived key	hkey	No
	nCore Key Hash of base key	hbase	Yes
	nCore Key Hash of wrap key	hwrsp	Yes
	nCore Key Hash of third input key	hin3	Yes
	nCore Key Hash of fourth input key	hin4	Yes
	nCore Key Hash of fifth input key	hin5	Yes
	The nCore Key Hashes for the input keys will only be included in the audit log if the Permission Group for the DeriveKey action has the LogKeyUsage flag.		

Command	Command Specific Information Logged	Extension	Requires logkeyusage ACL
Cmd_MakeBlob	nCore Key Hash nCore LT Hash	hkey hwrap	No
Cmd_LoadBlob	nCore Key Hash nCore LT Hash	hkey hwrap	No
Cmd_SetKM	nCore key hash	hkey	No
Cmd_RemoveKM	nCore key hash	hkey	No

24.6.4. Logical Token and Share Commands

These commands do not use the logkeyusage ACL mechanism and log unconditionally.

Command	Command Specific Information Logged	CEF Extension
Cmd_ChangeSharePIN	KM nCore Key Hash Short LT Hash Share Index Slot	hkm shorthash shareindex tokenslot
Cmd_Destroy	LT Hash	hkey
	Cmd_Destroy is used for Logical Tokens as well as Keys	
Cmd_EraseShare	Short LT Hash Share Index Slot	shorthash shareindex tokenslot
Cmd_GenerateLogicalToken	KM nCore Key Hash nCore LT Hash Token Shares Needed Token Total Shares Token time-limit	hkm htok sharesneeded sharestotal timelimit

Command	Command Specific Information Logged	CEF Extension
Cmd_LoadLogicalToken	KM Hash LT Hash Token Shares Needed Token Total Shares Token time-limit	hkm htok sharesneeded sharestotal timelimit
Cmd_ReadShare	LT Hash Share Index SlotId Share Left	htok shareindex tokenslot sharesleft
	This is the remaining number of shares required to reconstruct the Logical Token. It reduces to 0 when a quorum of the Shares have been read.	
Cmd_WriteShare	LT Hash Share Index SlotId	htok shareindex tokenslot

24.6.5. Administrative Commands

These commands are logged unconditionally.

Command	Command Specific Information Logged	CEF Extension
Cmd_InitialiseUnit	Type of KML key	kmltype
	DSAp3072s256	
Cmd_InitialiseUnitEx	Type of KML key InitialiseUnitEx Flags	kmltype flags
	DSAp3072s256 Combination of AuditLogging and UseFIPSAp-provedInternalMechanisms	

Command	Command Specific Information Logged	CEF Extension
Cmd_SetNSOPerms	nCore Key Hash of Security Officers Key SetNSOPermsFlags	hknso flags
	Combination of AlwaysUseStrongPrimes, DisablePKCS1Padding, FIPSLevel3Enforcedv2 and CommonCriteriaCMTSRestrictions	
Cmd_CreateSeeWorld		
Cmd_SetSEEMachine		
Cmd_SetRTC	Previous RTC	prevrtc
	New RTC value will be shown in the rtc extension	

24.6.6. Dynamic Slot Commands

These commands are logged unconditionally.

Command	Command Specific Information Logged	CEF Extension
Cmd_DynamicSlotsConfigure	Count of Dynamic Slots to be Configured	slotcount
Cmd_DynamicSlotCreateAssociation	Slot Id for Association	slotid
EstablishSecureChannel	ESN of smartcard	smartcardesn
	This internal event is logged with name element EstablishSecureChannel, a Severity of 2 and a Device Event Class Id of 2 in the CEF header and with source=internal in the CEF extensions.	

24.6.7. Heartbeat

The heartbeat is a periodic audit log message sent every 15 minutes. This audit log message indicates that the HSM is still active. After a heartbeat event is logged a Signature Block is generated including the heartbeat log message and any outstanding audit log messages. Waiting until the heartbeat is logged before restarting the HSM will ensure outstanding log messages can be verified.

Command	Command Specific Information Logged	CEF Extension
heartbeat	nCore Key Hash of Security Officers Key	hknso
	 The heartbeat is logged in the CEF header with name element heartbeat, Severity 4, and Device Event Class Id 2. In the CEF extensions it's logged with source=internal.	

24.6.8. Post Reboot Logging

The nShield HSM has a number of commands and errors that cannot be logged directly when they occur. This applies primarily to errors detected during processing or self test and the reboot command Cmd_ClearUnit. The strategy adopted for these is to persist sufficient information and replay them as log entries after a successful reboot of the HSM. These reboot event messages occur after the Certifier Block has been emitted.

Each of these messages are emitted with rsid and seqNo relating to the current session and will have a prevrtc CEF element recording the RTC at the time of the event. The name element will identify the event. If the event is associated with a nCore SOS code this will be indicated by a sos CEF extension and an appropriate code. The Device Event Class Id is set to 5 and Severity will be set to 10 for errors or 6 for shutdown events. The source CEF extension will be internal. The following table lists the events replayed in a post reboot log. The available events depend on the type of HSM.

Event Id	Event	SOS Code
Cmd_ClearUnit	Cmd_ClearUnit	
Cmd_Fail	Cmd_Fail	D
Environment_SensorFail		HV
Temperature_OutofRange		T
RNG_PeriodicTestFail		HRTP
SOS	Starting up crypto offload	HF
SOS	cache keygen failed	HR
Voltage_Tamper		V
Battery_Tamper		B
Unknown_Tamper		TAMPER

Event Id	Event	SOS Code
SelfTestFail	POST test timed out	HC0TTO
	POST test failed: lock failure detected	HC0LC
	POST test failed: TEST_STARTED	HC0TS
	POST test failed: PROCESS_STARTED	HC0PS
	POST test failed: CPUID_CHECK	HC0CC
	POST test failed: SRAM_ALLOC	HC0SA
	POST test failed: SRAM_WRITE	HC0SW
	POST test failed: SRAM_READ	HC0SR
	POST test failed: SRAM_FREE	HC0SF
	POST test failed: CRAM_ALLOC	HC0CA
	POST test failed: CRAM_GETCACHED	HC0CG
	POST test failed: CRAM_WRITE	HC0CW
	POST test failed: CRAM_READ	HC0CR
	POST test failed: CRAM_FREE	HC0CF
	POST test failed: LOCK_CHECK	HC0LC
	POST test failed: RTC_CHECK	HC0RT
	POST test failed: KAT_DSA	HC0KS
	POST test failed: KAT_ECDSA	HC0KC
	POST test failed: KAT_DES	HC0KE
	POST test failed: KAT_DES3	HC0KF
	POST test failed: KAT_DES3CBCMAC	HC0KO
	POST test failed: KAT_AES	HC0KA
	POST test failed: KAT_AESCMAC	HC0KB
	POST test failed: KAT_AESCBCMAC	HC0KD
	POST test failed: KAT_SHA1	HC0KH
	POST test failed: KAT_SHA1HMAC	HC0KM
	POST test failed: KAT_SHA224HMAC	HC0KN
	POST test failed: KAT_SHA256HMAC	HC0KJ
	POST test failed: KAT_SHA384HMAC	HC0KP

Event Id	Event	SOS Code
	POST test failed: KAT_SHA512HMAC	HC0KI
	POST test failed: KAT_RSA	HC0KRH
	POST test failed: KAT_NISTKDF	HC0KDF
	POST test failed: KAT_HASHDRBG	HC0HD
	POST test failed: KAT_RSAOAEP	HC0KZ
	POST test failed: KAT_25519	HC0KX
	POST test failed:unknown	HC0H

As an example, the following shows a post reboot log of `Cmd_ClearUnit`. In this excerpt, it can be seen after the last fragment of the Certifier Block. A Signature Block is generated after the reboot log entries.

```
....
<134>May 16 15:08:45 myhost2 CEF:0|nCipher Security|nShield5|13.2.2|3|ssign-cert|5|esn=1111-2222-4444 rsid=2
rtc=1524140117693 tpbl=2140 findex=5
flen=340frag=3/ITRJT4T/qgd2ZEJufIzCR+nR9lngOrmogj+5JM7VMFLsWGDxUqxmFlpqs52T2zWuYIeFHGQfx9WS9PUhf2eLMyF/7onn+hFUs5
7/GSZlGbCnxWybfPN27oyXjHE7pfyOrWRVKlIw8UULHVezVsxeIsZuuNEsZa5gUQ++DkoTu5M2BoPr4A+6dVL2eDhOF1m2zKATfk2moW93GkA3AO7
lNPV5xU76ujo2tT7Mttvg+vyddiF2UWe6n75U0FMFjlM9WnhpFAhNk9mJPrNZ5smf4i9JuNKZat+5tq5w2b/a8Sy01EVEKtJI5SSjahtp5z77RseQ
8H8ytsw6oAAAA=sign=MEUCIHgrF1m7t9X5xsl/gXwlju0bPfFPjJeIeIiH8TKSN7prAiEAs3lPS62zX3TE940/Dw9/1gVradNi62wrQI+WlSI4IY
U=

<134>May 16 15:08:45 myhost2 CEF:0|nCipher Security|nShield5|13.2.2|5|Cmd_ClearUnit|6|esn=1111-2222-4444 rsid=2
rtc=1524140117693 seqNo=1 source=internal prevrtc=1524140108693

<134>May 16 15:08:45 myhost2  CEF:0|nCipher Security|nShield5|13.2.2|3|ssign|5|esn=1111-2222-4444 rsid=2
rtc=1524140117693 gbc=0 fmn=1 hcnt=1
hb=nwtggjmPYA1TR07KhdOHoyytxLb7RDvg7Wpw6FfAiC4=sign=MEYCIQDxIIJZRfKsXpMMoQ3GDEkTZ/+DTuEdNLKwHQzllflMUQIhAPipdSPrB
SUnarrtjMslYS4k3RPCXcNoO16xEhg/907z
```

24.6.9. Tracing Key Usage

With the information logged as detailed in the preceding sections it is possible to trace back from a Key Command to the loading of the Key, then to loading the Logical Token and reading the Shares that constitute the Logical Token.

The following example shows the notional traceback from a `Cmd_Encrypt` operation. This command logs the nCore Key Hash **KKKKKKK**. Prior to this the Key was loaded onto the HSM using `Cmd_LoadBlob` which correlates the nCore Key Hash with the ncore Hash of the Logical Token that authorized loading the Key. Tracing further back we can identify the shares used to reconstruct that Logical Token. In this example two shares are required identified by share indices S1 and S2. The share index identifies a specific card in an OCS cardset.

Command	Key Hash	Logical Token Hash	Share Index
Cmd_Encrypt	KKKKKKK		
Cmd_LoadBlob	KKKKKKK	LLLLLLL	
Cmd_Read Share		LLLLLLL	S2
Cmd_Read Share		LLLLLLL	S1
Cmd_LoadLogicalToken		LLLLLLL	

24.7. Audit Log Verification

The audit logs produced when AuditLogging feature is active can be verified using the infor mation contained in the audit logging metadata. Every HSM enrolled into a Security World with AuditLogging enabled generates an HSM-specific log signing private key (KAL) that is maintained in the HSM's non-volatile memory until the module is re-initialized. The public key corresponding to this private key is sent as a Certifier Block by the HSM when Audit Logging is configured either by Security World creation or by indoctrination into an existing Audit Logging Security World. Every Signature Block sent by the HSM is generated using the log signing private key. The Audit Log can be verified as follows:

- Extract the KAL public key from the Certifier block
- Verify the Signature Blocks
- Verify the log message hashes in the Signature Block against hashes of the received logs to determine if any messages have been tampered
- Identify any missing log messages.

The basics of the verification approach is shown on the Audit Log Verification diagram.

To support Audit Log verification, Entrust provide an example verification program written in Python to serve as an example for developing a more comprehensive verification solu- tion.

24.7.1. Running the example verification program

The example verification program can be found in the following location:

```
/opt/nfast/python/examples/audit-log-verifier.py
```

This program requires the use of the nShield Python interpreter. This is necessary to provide support for the nShield specific marshalling functions used to export the log signing public

key. The example verification program also requires the presence of an nShield HSM accessible to the machine on which the verification is to be performed. This is required to perform the cryptographic operations necessary to verify the log signing public key and the Signature Blocks. This HSM does not need to be the same HSM on which the logs were generated, nor does it need to be in a Security World.

The Audit Log verifier program is run with a command of the form:

```
python audit-log-verifier.py [-h] [-e ESN] SYSLOG
```

Where:

Parameter	Function
`-h|--help`	Displays the help message
`-e ESN|--esn ESN`	ESN of the logevents to be verified
`SYSLOG`	Location of the syslog file to be verified

Make sure that you use the nShield Python.

24.7.1.1. Results

Results from the Audit Log Verifier are written to several different files and saved in a subdirectory called LogResult. See example below for more detail.

24.7.1.2. Example

Running a command of the form:

```
> python audit-log-verifier.py AuditLogInputFile.txt
```

Should produce a screen output similar to the following:

```
FIRST LOG INSTANCE-1 for ESN:9204-02E0-D947 @ Line:1 rsid:8 ######

Verifying certifier block...
Verification of CERTIFICATE Success
Verifying a cert fragment...Line:1
Verifying a cert fragment...Line:2
Verifying a cert fragment...Line:3
Verifying a cert fragment...Line:4
Verifying a cert fragment...Line:5
Verifying SB....Lineno:7:InstanceNo:1
Verifying SB....Lineno:18:InstanceNo:1
Valid Hash list from SBs written to ValidHashes_fromSB_forInst1.txt
No entry in SB for event @ Line:19 : Seq No:12 --
```

```
No entry in SB for event @ Line:20 : Seq No:13 --

Verified cert blocks written to./LogResult/CBs.txt
Sig blocks written to./LogResult/SBs.txt
Log messages written to./LogResult/AllEvents.txt
Anything that did not match (incl.
invalid cert blockfragments)  written to :./LogResult/Inconsistent.txt
```

The screen output indicates the contents of the main results files which are stored in the *LogResult* sub-directory. The contents of the folder will vary slightly depending on the log contents and whether there were any failures, but should be similar to the following:

```
AllEvents.txt                     # log entries relating to events
CBs.txt                           # certificate blocks
Inconsistent.txt                  # inconsistent log entries - should be empty []
                                   (assuming no inconsistencies)
Instance.txt                      # esn number and other info relating to the log
InvalidHashes_fromSB_forInst1     # only exists if verification failed due to signature
block (forInst1 refers to
                                   first logging instance/world found in the
                                   log file)
SBs.txt                           # signature blocks
Tampered_logs.txt                 # contains log messages that did not verify - e.g. due
                                   to a corrupt signature block.
This file only exists
                                   if verification failed.
Unverified_logs.txt               # unverified log entries - e.g. any trailing entries
                                   from the end of the log file that lack an accompanying
                                   signature block
ValidHashes_fromSB_forInst1.txt   # valid hashes from the signature blocks
                                   (forInst1 refers to first logging
                                   instance/world found in the log file)
Verified_logs.txt                 # verified log messages
```

Use a text editor to examine the files as required to check the verification. Note that `Inst1` in the filenames refers to the first `logging world` instance in the log, see Program Architecture. If the log contains messages relating to more than one `logging world`, files relating to subsequent instances will be tagged with `Inst2`, `Inst3` etc.

If the verification fails, screen output should indicate the source of the failure. For example, output for a log where a log message was missing would look something like this:

```
FIRST LOG INSTANCE-1 for ESN:9204-02E0-D947 @ Line:1 rsid:8 ######

Verifying certifier block...
Verification of CERTIFICATE Success
Verifying a cert fragment...Line:1
Verifying a cert fragment...Line:2
Verifying a cert fragment...Line:3
Verifying a cert fragment...Line:4
Verifying a cert fragment...Line:5
Verifying SB....Lineno:7:InstanceNo:1
Verifying SB....Lineno:17:InstanceNo:1
Verifying SB....Lineno:28:InstanceNo:1
Valid Hash list from SBs written to ValidHashes_fromSB_forInst1.txt
No entry in SB for event @ Line:29 : Seq No:22 --
No entry in SB for event @ Line:30 : Seq No:23 --
```

```
@@@@@@@@ Some Log events present in the SB but missing in Log file                          # indicates
missing log message
Verified cert blocks written to./LogResult/CBs.txt
Sig blocks written to./LogResult/SBs.txt
Log messages written to./LogResult/AllEvents.txt
Anything that did not match (incl.
invalid cert blockfragments)  written to :./LogResult/Inconsistent.txt
```

Output for a log where a log message had been tampered with or is otherwise corrupt might look like this:

```
FIRST LOG INSTANCE-1 for ESN:9204-02E0-D947 @ Line:1 rsid:8 ######

Verifying certifier block...
Verification of CERTIFICATE Success
Verifying a cert fragment...Line:1
Verifying a cert fragment...Line:2
Verifying a cert fragment...Line:3
Verifying a cert fragment...Line:4
Verifying a cert fragment...Line:5
Verifying SB....Lineno:7:InstanceNo:1
Verifying SB....Lineno:18:InstanceNo:1
Verifying SB....Lineno:29:InstanceNo:1
Valid Hash list from SBs written to ValidHashes_fromSB_forInst1.txt
Validating Log @ Line No:10 SeqNo:4 is Failed ----                                        # indicates
tampered log entry
***** Hash Mismatch No entry in SB for event @ Line:30 : Seq No:22 --
No entry in SB for event @ Line:31 : Seq No:23 --
Verified cert blocks written to./LogResult/CBs.txt
Sig blocks written to./LogResult/SBs.txt
Log messages written to./LogResult/AllEvents.txt
Anything that did not match (incl.
invalid cert blockfragments)  written to :./LogResult/Inconsistent.txt
```

The tampered log line(s) will be listed in output file `Tampered_logs.txt`.

Output for a log where the signature block is corrupt will look something like this:

```
FIRST LOG INSTANCE-1 for ESN:9204-02E0-D947 @ Line:1 rsid:8 ######

Verifying certifier block...
Verification of CERTIFICATE Success
Verifying a cert fragment...Line:1
Verifying a cert fragment...Line:2
Verifying a cert fragment...Line:3
Verifying a cert fragment...Line:4
Verifying a cert fragment...Line:5
Verifying SB....Lineno:7:InstanceNo:1
Verifying SB....Lineno:18:InstanceNo:1
Signature Tampered B64 decode

//k=&s0QG1C08QBi34gaTU2+rUzp/dwtAXi9Hv0IjDvDL/yg=&Im5nW+OX0gbdlLnrFLxsZtR4meDSEXG5JXtkMmltTZU=&LGAXS1nvgHElvXhk8R
VT2lCK2NMtXyD9OYTecVOaaBk=&MbJAk706yU2+QykWmtfnCV0lxn/enber8aJK3cZyxLg=&y2qxF5VGm/X/h6ZcZ5iOes7ZAFpqM/6ND8nAXzCM/
bY=&kWjEaGIclJv494A1ZcUgGHJko7AeKvUUqVimhfExioU=  Length: 577
Verifying SB....Lineno:29:InstanceNo:1
Valid Hash list from SBs written to ValidHashes_fromSB_forInst1.txt
In-Valid Hash list from SBs written to InvalidHashes_fromSB_forInst1.txt
        Log entry found in Tampered SB. Line no:8 SeqNo:2
        Log entry found in Tampered SB. Line no:9 SeqNo:3
        Log entry found in Tampered SB. Line no:10 SeqNo:4
        Log entry found in Tampered SB. Line no:11 SeqNo:5
```

```
        Log entry found in Tampered SB. Line no:12 SeqNo:6
        Log entry found in Tampered SB. Line no:13 SeqNo:7
        Log entry found in Tampered SB. Line no:14 SeqNo:8
        Log entry found in Tampered SB. Line no:15 SeqNo:9
        Log entry found in Tampered SB. Line no:16 SeqNo:10
        Log entry found in Tampered SB. Line no:17 SeqNo:11
No entry in SB for event @ Line:30 : Seq No:22 --
No entry in SB for event @ Line:31 : Seq No:23 --
Verified cert blocks written to./LogResult/CBs.txt
Sig blocks written to./LogResult/SBs.txt
Log messages written to./LogResult/AllEvents.txt
Anything that did not match (incl.
invalid cert blockfragments)  written to :./LogResult/Inconsistent.txt
```

The failed log messages should be reported in `Tampered_logs.txt` in the `LogResult` folder.

If the certificate block is corrupt, output will be similar to that shown below. In this case, the `CBs.txt` file may be empty and the cert block fragments will be written to `Inconsistent.txt`.

```
FIRST LOG INSTANCE-1 for ESN:9204-02E0-D947 @ Line:1 rsid:8 ######

Verifying certifier block...
Verification of CERTIFICATE Success
Verifying a cert fragment...Line:1
Verifying a cert fragment...Line:2
Verifying a cert fragment...Line:3
Signature Tampered B64 decode
('Failed fragment:', 3, '<134>May  2 16:10:38 exampleCB1.myexample.com CEF:0|nCipher Security|nShield Solo
XC|12.60.2|3|ssign-cert|5|esn=9204-02E0-D947 rsid=8rtc=4294967386734000 tpbl=2140 findex=3 flen=450
frag=B1Ku9rirlixkgEd+73tMVJ1FQz85aCWuRqJl04YB1YwFvZgvRXhHvzqLFeJZAUerKlLgIaZwDq1twoXzvHq88QcJdbr0i4+87VorPKkEjKtS
SGHOVkkHhoBC8uNgYXnTBxqcqCqpZl4whuiEBmJQLcwgAAAAg8rgckmo3ArobecQooPxQ9AjYbCmAoKOUTRi7grTzPyAAQAA3Bvuz+tQ1uh5LvuKM
LTtGDTTplG7ks6ZkL8b+F2UW37jfN3lap27oAZq1otU4FOP4EVvoMmNSdI4uzCPi7VgcI3AcIkdjZIwbpYf9XQwvFwMxYvdBPGhPtc/t8Lslgs97r
MkES4ZciNI/NwjKp0fW4kCiSBSUQUAUcp6vgqg2vVL9naqRHhXNRJuweaRtOO60z0mBkTgCnAvscdr2ymErrWDZArHosYXJZrXghjNmXvu+rS8GvT
vTc sign=MEYCIQCXhbJeFTv8oR8a51aU3Os9w2Vs9w67mzYk584Gy+MdbgIhAOD0bAwU0Vw1xOmR2oerqWKLFeawZe5rONmDMZFbJoB')
Fragment verification unsuccessful!
Adding all fragments in this CB to Inconsistent.txt
('No Valid CB for this instance:', 1)
In-Valid Hash list from SBs written to InvalidHashes_fromSB_forInst1.txt
        Log entry found in Tampered SB. Line no:6 SeqNo:1
        Log entry found in Tampered SB. Line no:8 SeqNo:2
        Log entry found in Tampered SB. Line no:9 SeqNo:3
        Log entry found in Tampered SB. Line no:10 SeqNo:4
        Log entry found in Tampered SB. Line no:11 SeqNo:5
        Log entry found in Tampered SB. Line no:12 SeqNo:6
        Log entry found in Tampered SB. Line no:13 SeqNo:7
        Log entry found in Tampered SB. Line no:14 SeqNo:8
        Log entry found in Tampered SB. Line no:15 SeqNo:9
        Log entry found in Tampered SB. Line no:16 SeqNo:10
        Log entry found in Tampered SB. Line no:17 SeqNo:11
        Log entry found in Tampered SB. Line no:19 SeqNo:12
        Log entry found in Tampered SB. Line no:20 SeqNo:13
        Log entry found in Tampered SB. Line no:21 SeqNo:14
        Log entry found in Tampered SB. Line no:22 SeqNo:15
        Log entry found in Tampered SB. Line no:23 SeqNo:16
        Log entry found in Tampered SB. Line no:24 SeqNo:17
        Log entry found in Tampered SB. Line no:25 SeqNo:18
        Log entry found in Tampered SB. Line no:26 SeqNo:19
        Log entry found in Tampered SB. Line no:27 SeqNo:20
        Log entry found in Tampered SB. Line no:28 SeqNo:21
No entry in SB for event @ Line:30 : Seq No:22 --
No entry in SB for event @ Line:31 : Seq No:23 --
```

```
Verified cert blocks written to./LogResult/CBs.txt
Sig blocks written to./LogResult/SBs.txt
Log messages written to./LogResult/AllEvents.txt
Anything that did not match (incl.
invalid cert blockfragments)  written to :./LogResult/Inconsistent.txt
```

24.7.2. Program Architecture

The program takes and reads the input syslog file containing the log messages. It optionally sets the ESN of module for which log events are to be validated, if this was passed in. If an ESN is not provided as input then the first ESN found in the syslog will be processed.

The verifier calls its parse function which segregates the messages based on ESN, and creates lists of Certifier fragments, Signature Blocks and Log events, based on matching with regular expressions.

Syslog may have gathered logs from multiple sources. As such, the verifier has a concept of a *logging world*, which represents a set of logs, sigblocks and certblocks that belong together, from a Security World. Based on Reboot Sequence ID, Sequence Number of the Log event, Global block counter of the Signature block and Fragment index of the Certifier block, a logging world is identified and a logging instance is created.

All records are thus given a log-instance number, such that records with the same instance number belong together.

Each event can thus be uniquely identified via a tuple. For the log messages, signature blocks and certifier blocks these are respectively (`rsid` and `sequence number`), (`rsid` and `gbc`) and (`rsid` and `findex`).

The `reconstruct_CBs` function is then called to validate the certifier fragments (using calls to an nShield HSM for crypto functionality). It then reconstructs the certifier blocks from the certifier fragments.

This does not require the HSM to be in the same Security World as the HSM that first generated the logs.

A list of valid and verified Certifier Blocks is created.

For any log instance one valid Certifier Block is enough to validate the events, so further cer tifier blocks are ignored after the first.

Next the `process_sbs` function is called. Signature Blocks for a supplied ESN are validated per log instance (once again via calls to the module for crypto functionality), using the KAL value taken from the Certifier block previously.

The validated Signature block hashes are maintained as a dictionary of hashes with keys as unique ids. These unique ids per instance are generated based on rsid and sequence numbers.

The `process_logs` function is finally called. This generates the hash of each of the log events and matches against hashes from corresponding signature blocks. Verified and Tampered log events are then written to different files in the *LogResult* folder.

24.7.3. Extended Verification

While the example verifier uses an HSM for cryptographic operation, it would be possible to use 3rd party cryptographic libraries to provide this functionality. This is outside the scope of this document.

Currently the log messages are verified against the hash in the signature blocks, and the signature of the signature blocks is verified against the key extracted from the certifier block. The certifier block itself is not verified. A potential extension to the verifier tool would be to verify the certifier block. The certifier block is signed by KLF2. This can be checked against the KLF2 value found within the module's warrant. This would complete the chain of trust.

Additionally, the example verifier does not cope with fields that rotate back around to zero when their max size is exceeded. (for example, `gbk`, `rsid` or `seqno`). Currently logs, SBs and CBs are uniquely identified by (`rsid` and `sequence number`), (`rsid` and `gbc`) and (`rsid` and `findex`). This means that, if any of those values rotate back around to zero, we are no longer able to uniquely identify them. As a potential extension, RTC or line number values could be used to solve this.

The example verifier does not detect missing/deleted log messages in the case where a complete group of log messages are deleted, along with their corresponding `SignatureBlock`. Given that the `SeqNo` field increases for each log message, spotting missing `SeqNos` would reveal missing or deleted log messages. This is a potential extension.

The example verifier expects a static, unchanging log file to be supplied to it. This would be compatible with verifying a batch of log files at the end of each day, for example. A possible extension would be to extend the verifier to cope with a live stream of logs, continuously verifying them as they are generated.

25. Key generation options and parameters

This appendix describes the various options and parameters that you can set when running the `generatekey` utility to control the application type and other properties of a key being generated.

For information about generating keys with the `generatekey` utility, see Generating keys with the command line.

25.1. Key application type (APPNAME)

The `APPNAME` parameter specifies the name of the application for which `generatekey` can generate keys. Specifying an application can restrict your choice of key type. A value for `APPNAME` must follow any `OPTIONS` and must precede any parameters specified for the key:

Parameter	Description
`simple`	Specifying the `simple` application type generates an nShield-native key. No special action is taken after the key is generated.
`custom`	Specifying the `custom` application type generates a key for custom applications that require the key blob to be saved in a separate file. Specifying `custom` also causes the generation of a certificate request and self-signed certificate. However, we recommend that you specify the `simple` (instead of `custom`) application type whenever possible.
`pkcs11`	Specifying the `pkcs11` application type generates keys that are formatted for use with PKCS #11 applications and are given a suitable identifier. The set of possible supported key types is currently limited to: • `DES3` • `DH` • `DSA` • `ECDH` • `ECDSA` • `Ed25519` • `HMACSHA1` • `RSA` • `Rijndael` (`AES`) • `X25519` Some key types are only available if the features that support them have been enabled for the module, if the Security World is not compliant with FIPS 140 Level 3, or if you do not set the `--no-verify` option.

Parameter	Description
embed	Specifying the embed application type generates a PEM-format RSA/DSA key file that points to a key in NFAST_KMDATA so a software application can then use the HSM-protected key. In applications that use Security World software older than v12.60 and would use the legacy OpenSSL CHIL engine with hwcrhk: • The plainname specified in the generatekey command is used as the prefix for all 3 generated files (.key, _req, _selfcert) • .key is appended to all 3 files • The embedsavefile specified in the generatekey command is the destination for all 3 files In applications that use v12.60 or later Security World software : • The plainname specified in the generatekey command is used as the prefix for only the .key file, the prefix for the _req and _selfcert file is embed<hash> • .key is not appended to the _req and _selfcert files • The embedsavefile is the destination only for the .key file, _req and _selfcert are created in the directory from which generatekey was run from
kpm	Specifying the kpm application type generates a key for delivery by an nForce Ultra key server. The generatekey utility automatically creates a special ACL entry that permits a kpm to be delivered to an nForce Ultra's enrolled internal hardware security module.
seeinteg	Specifying the seeinteg application type generates an SEE integrity key. The DSA, RSA, ECDSA and KCDSA algorithms are supported. SEE integrity keys are always protected by an OCS and cannot be imported. You cannot retarget an existing key as an SEE integrity key.
seeconf	Specifying the seeconf application type generates an SEE confidentiality key. Both the Triple DES and AES algorithms are supported for this key type. SEE confidentiality keys are module-protected by default and cannot be imported. You cannot retarget an existing key as an SEE confidentiality key.

25.2. Key properties (NAME=VALUE)

The NAME=VALUE syntax is used to specify the properties of the key being generated.

If a parameter's argument contains spaces, you must enclose the argument within quotation marks (" ").

You can supply an appropriate *VALUE* for the following *NAME* options:

Option	Description
`alias`	The *VALUE* for `alias` specifies an alias to assign to the key.
`blobsavefile`	When using the `custom` application type, the *VALUE* for `blobsavefile` specifies a file name of the form *FILENAME*`_req.`*ext* to which the key blob is saved. Additionally, a text file containing information about the key is saved to a file whose name has the form *ROOT*`_inf.txt`; for asymmetric key types, the public key blob is also saved to a file whose name has the form *ROOT*`_pub.`*EXT*.
`cardset`	The *VALUE* for `cardset` specifies an OCS that is to protect the key (if `protect` is set to `token`). In interactive mode, if you do not specify an OCS, you are prompted to select one at card-loading time. The default is the OCS to which the card currently inserted in the slot belongs (or the first one returned by `nfkminfo`).
`certreq`	Setting `certreq` enables you to generate a certificate request when generating a PKCS #11 key (RSA keys only). The default behavior is to not generate a certificate request. To generate a certificate request you must set the *VALUE* for `certreq` to `yes`, which makes `generatekey` prompt you to fill in the extra fields required to generate a key with a certificate request. The resultant certificate request is saved to the current working directory with a file name of the form *FILENAME*`req.ext` (where *FILENAME* is a name of your choice). An extra file with a name of the form *FILENAME*`.ext` is also generated for use as a pseudo-key-header. This file can be removed after the certificate request has been generated. You can use `certreq` with the `--retarget` option to generate a self-signed certificate for an existing key.
`checks`	For RSA key generation only, this specifies the number of checks to be performed. Normally, you should leave *VALUE* empty to let the module pick an appropriate default.
`curve`	For ECDH and ECDSA key generation only, the *VALUE* for `curve` specifies which curves from the supported range to use. Supported curves are: ANSIB163v1, ANSIB191v1,BrainpoolP160r1, BrainpoolP160t1, BrainpoolP192r1, BrainpoolP192t1, BrainpoolP224r1, BrainpoolP224t1, BrainpoolP256r1, BrainpoolP256t1, BrainpoolP320r1, BrainpoolP320t1, BrainpoolP384r1, BrainpoolP384t1, BrainpoolP512r1, BrainpoolP512t1, NISTP192, NISTP224, NISTP256, NISTP384, NISTP521, NISTB163, NISTB233, NISTB283, NISTB409, NISTB571, NISTK163, NISTK233, NISTK283, NISTK409, NISTK571, SECP160r1 and SECP256k1
`embedconvfile`	The *VALUE* for `embedconvfile` specifies the name of the PEM file that contains the RSA key to be converted.

Option	Description
`embedsavefile`	When using the `embed` application type, the `VALUE` for `embedsavefile` specifies the name for the file where the fake RSA private key is to be saved. The file has the same syntax as an RSA private key file, but actually contains the key identifier rather than the key itself, which remains protected. A certificate request and a self-signed certificate are also written. If the file-name is *ROOT*`.EXT` then the request is saved to *ROOT*`_req.EXT` and the self-signed certificate is saved to *ROOT*`_selfcert.EXT`.
`from-application`	When retargeting a key, the *VALUE* for `from-application` specifies the application name of the key to be retargeted. Only applications for which at least one key exists are acceptable.
`from-ident`	When retargeting a key, the *VALUE* for `from-ident` specifies the identifier of the key to be retargeted (as displayed by the `nfkminfo` command-line utility).
`hexdata`	The *VALUE* for `hexdata` specifies the hex value of DES or Triple DES key to import. The hex digits are echoed to the screen and can appear in process listings if this parameter is specified in the command line.
`ident`	The *VALUE* for `ident` specifies a unique identifier for the key in the Security World. For applications of types `simple`, this is the key identifier to use. For other application types, keys are assigned an automatically generated identifier and accessed by means of some application-specific name. The following characters are allowed in key IDs: • digits 0-9 • lower-case letters a-z • hyphen (-)
`keystore`	The *VALUE* for `keystore` specifies the file name of the key store to use. This must be an nShield key store.
`keystorepass`	The *VALUE* for `keystorepass` specifies the password to the key store to use.
`logkeyusage`	The *VALUE* for `logkeyusage` specifies if usage of the generated key in cryptographic operations is subject to audit logging. If set to **yes** the ACL of the generated key will predicate audit-logging entries to be made for cryptographic usages of the key. The default is **no**.
`module`	The *VALUE* for `module` specifies a module to use when generating the key. If there is more than one usable module, you are prompted to supply a value for one of them. The default is the first usable module (one in the current Security World and in the operational state).  You can also specify a module by setting the `--module` option.

Option	Description
paramsreadfile	The *VALUE* for paramsreadfile specifies the name of the group parameters file that contains the discrete log group parameters for Diffie-Hellman keys only. This should be a PEM-formatted PKCS#3 file. If a *VALUE* for paramsreadfile is not specified, the module uses a default file.
pemreadfile	The *VALUE* for pemreadfile specifies the name of the PEM file that contains the key to be imported. When importing an RSA key, this is the name of the PEM-encoded PKCS #1 file to read it from. Password-protected PEM files are not supported.
plainname	The *VALUE* for plainname specifies the key name within the Security World. For some applications, the key identifier is derived from the name, but for others the name is just recorded in *kmdata* and not used otherwise.
protect	The *VALUE* for protect specifies the protection method, which can be module for security-world protection, softcard for softcard protection or token for Operator Card Set protection. The default is token, except for seeconf keys, where the default is module. seeinteg keys are always token-protected. The softcard option is only available when your system has at least one softcard present.
pubexp	For RSA key generation only, the *VALUE* for pubexp specifies (in hexadecimal format) the public exponent to use when generating RSA keys. We recommend leaving this parameter blank unless advised to supply a particular value by Support.
recovery	The *VALUE* for recovery enables recovery for this key and is only available for card-set protected keys in a recovery-enabled world. If set to yes, the key is recoverable. If set to no, key is not recoverable. The default is yes. Non-recoverable module-protected keys are not supported.
seeintegname	If present, the *VALUE* for seeintegname identifies a seeinteg key. The ACL of the newly generated private key is modified to require a certificate from the seeinteg key for its main operational permissions, such Decrypt and Sign (DuplicateHandle, ReduceACL, and GetACL are still permitted without certification.) If you use seeintegname to specify a key that has been recovered with the rocs utility, you must also use the -N option with generatekey.
selfcert	The *VALUE* for selfcert enables you to generate a self-signed certificate when generating a PKCS #11 key (RSA keys only). To generate a self-signed certificate request you must set selfcert to yes, which makes generatekey prompt you to fill in the extra fields required to generate a key with a self-signed certificate. The resultant certificate is saved to the current working directory with a file name of the form *FILENAME*.ext. You can use this parameter with the --retarget option to generated a self-signed certificate for an existing key.

Option	Description
`size`	For key types with variable-sized keys, the *VALUE* for `size` specifies the key size in bits. The range of allowable sizes depends on the key type and whether the `--no-verify` option is used. The default depends on the key type; for infor mation on available key types and sizes, see Cryptographic algorithms. This parameter does not exist for fixed-size keys, nor for ECDH and ECDSA keys which are specified using `curve`.
`strict`	For DSA key generation only, setting the *VALUE* for `strict` to `yes` enables strict verification, which also limits the size to 2048 or 3072 bits. The default is `no`.
`type`	The *VALUE* for `type` specifies the type of key. You must usually specify the key type for generation and import (though some applications only support one key type, in which case you are not asked to choose). Sometimes the type must also be specified for retargeting; for information on available key types and sizes, see Cryptographic algorithms. The `--verify` option limits the avail- able key types.
`x509country`	The *VALUE* for `x509country` specifies a country code, which must be a valid 2-letter code, for the certificate request.
`x509dnscommon`	The *VALUE* for `x509dnscommon` specifies a site domain name, which can be any valid domain name, for the certificate request.
`x509email`	The *VALUE* for `x509email` specifies an email address for the certificate request.
`x509locality`	The *VALUE* for `x509locality` specifies a city or locality for the certificate request.
`x509org`	The *VALUE* for `x509org` specifies an organization for the certificate request.
`x509orgunit`	The *VALUE* for `x509orgunit` specifies an organizational unit for the certificate request.
`x509province`	The *VALUE* for `x509province` specifies a province for the certificate request.
`xsize`	The *VALUE* for `xsize` specifies the private key size in bits when generating Diffie-Hellman keys. The defaults are 256 bits for a key size of 1500 bits or more or 160 bits for other key sizes.

25.3. Available key properties by action/application

The following table shows which actions (generate, import, and retarget) are applicable to the different *NAME* options:

Property	generate	import	retarget
`alias`	X	X	X

Property	generate	import	retarget
blobsavefile	X	X	X
cardset	X	X	
certreq			
checks	X		
curve	X		
embedconvfile		X	
embedsavefile	X	X	X
from-application			X
from-ident			X
hexdata		X	
ident	X	X	
keystore	X	X	X
keystorepass	X	X	X
module	X	X	
nvram	X	X	
paramsreadfile	X		
pemreadfile		X	
plainname	X	X	X
protect	X	X	
pubexp	X		
qsize	X		
recovery	X	X	
seeintegname			
selfcert			
size	X		
strict	X		
type	X		
x509country	X	X	X
x509dnscommon	X	X	X

Property	generate	import	retarget
x509email	X	X	X
x509locality	X	X	X
x509org	X	X	X
x509orgunit	X	X	X
x509province	X	X	X
xsize	X		

The following table shows which applications are applicable to the different *NAME* options:

Property	custom	embed	hwcrhk	pkcs 11	seeconf	seeinteg	seessl	simple	kpm
alias									
blobsavefile	X								
cardset	X	X	X	X				X	X
certreq				X					
checks	X	X	X	X				X	X
curve	X	X	X	X	X	X		X	
embedconvfile		X							
embedsavefile		X		X					
from-application	X	X	X	X				X	X
from-ident	X	X	X	X				X	X
hexdata	X	X	X	X				X	
ident			X					X	X
keystore									
keystorepass									
module	X	X	X	X			X	X	X
nvram	X	X	X	X				X	
paramsreadfile	X	X	X	X	X	X		X	
pemreadfile	X		X					X	X
plainname	X	X		X	X	X	X	X	X
protect	X	X	X	X	X	X	X	X	X

Property	custom	embed	hwcrhk	pkcs 11	seeconf	seeinteg	seessl	simple	kpm
pubexp	X	X	X	X				X	X
qsize	X	X	X	X				X	X
recovery	X	X	X	X	X	X		X	X
seeintegname	X						X	X	
selfcert				X					
size	X	X	X	X	X	X	X	X	X
strict	X	X	X	X				X	
type	X	X	X	X	X	X	X	X	X
x509country		X							X
x509dnscommon		X							X
x509email		X							X
x509locality		X							X
x509org		X							X
x509orgunit		X							X
x509province		X							X
xsize	X	X	X	X				X	

26. Checking and changing the mode on an nShield 5s module

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

Use the following commands to change the mode of a module:

Command	Resulting mode
`nopclearfail --maintenance |-M`	Pre-maintenance
`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.

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

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

27. Maintenance of nShield Hardware

This chapter describes maintenance steps for your nShield hardware installation.

After installing your nShield HSM by following the *Installation Guide*, Entrust recommend that you use some of the provided software utilities to monitor your installation. Specifically, the `stattree` command allows reporting of voltages and temperatures from your mod ule.

For more information regarding `stattree`, see stattree: information utility.

27.1. Voltage Monitoring for Battery Replacement

All of the voltage rails in the nShield HSM are monitored to protect against potential over- or under-voltage attacks. You can view the most recent measurement of the voltages using the `stattree` command.

These modules also contain a user-replaceable battery. The battery powers security functions on the module when the main module power is removed, for example when the host is turned off, so it is expected that the battery voltage will drop over time as the battery drains. To avoid module downtime due to battery replacement we recommend that the bat tery voltage is monitored regularly, especially if a module has had its main power removed for considerable time.

`CPUVoltage10` reported by `stattree` under the `ModuleEnvStats` node tag displays the current battery voltage:

```
+PerModule:
   +#1:
      +ModuleEnvStats:
         ...
         -CPUVoltage10         3.16
         ...
```

The battery supplied with the nShield HSM has a nominal voltage of 3.0V. In the above example the battery is fully charged and has been measured at 3.16V, which is within the acceptable range of 2.46V - 3.55V. If the battery voltage is measured to be lower than 2.46V, the module will report an `SOS-B1` error. See Error codes for more information regarding error reporting.

Contact Support to request information regarding a replacement battery if `stattree` reports the battery voltage to be below 2.70V.

Consult the *Installation Guide* for instructions on replacing the battery in your module.

27.2. Temperature Monitoring for Airflow Validation

Temperatures within a module are monitored to protect against potential attacks, and to prevent overheating. The temperatures of the processors within an nShield 5s are reported as `CurrentCPUTemp1` and `CurrentCPUTemp2` under the `ModuleEnvStats` node tag of `stattree`.

As an nShield 5s has a passively-cooled heatsink, care must be taken to install it in an environment with forced airflow. Refer to the *Installation Guide* for airflow guidance.

The table below documents the expected normal operating ranges for the temperatures of your module. Module temperatures would be expected to be within these values when installed with sufficient cooling in an approximately 20-30°C ambient air temperature environment. Calculated `stattree` statistics such as minima and maxima are reset on module reboot.

The temperatures in this table do not cover operation of the product across the full temperature range specified in the *Warnings & Cautions* and *Installation Guide*. This is because these values are recommendations to ensure a long product lifetime, thus are specified for 20-30°C ambient air operation.

`stattree` Statistic	Description	Minimum expected in optimum environment	Maximum expected in optimum environment
`CurrentCPUTemp1`	First processor temperature	10°C	75°C
`CurrentCPUTemp2`	Second processor temperature	10°C	78°C
`MaxTempC`	Maximum temperature measured on either processor	-	78°C
`MinTempC`	Minimum temperature measured on either processor	10°C	-

If any of the above temperatures are reporting higher than their specified maximum it is likely your nShield hardware does not have sufficient cooling. Please refer to the *Installation Guide* to confirm your cooling setup.

28. Upgrading firmware

This appendix describes how to load firmware onto your nShield HSM hardware security module.

28.1. Version Security Number (VSN)

The firmware includes a Version Security Number (VSN). This number is increased whenever Entrust improve the security of the firmware.

Entrust supply several versions of the module firmware. Every HSM records the minimum firmware VSN that it will accept. You can always upgrade to firmware with an equal or higher VSN than the minimum VSN set on your module, even if the firmware currently installed on the module has a higher VSN than the firmware to which you are upgrading. firmware currently installed on the module has a higher VSN than the firmware to which you are upgrading.

You can never load firmware with a lower VSN than the target HSM's minimum VSN requirement.

For example, if the HSM has a minimum VSN requirement of 3 and the currently installed firmware has a VSN of 4, you can install firmware with a VSN of 3 or above to the HSM. You cannot install firmware with a VSN of 1 or 2 to this HSM.

To increase the HSM'as minimum VSN requirement, use the `hsmadmin setminvsn` command. The new VSN must be greater than or equal to the HSM's current minimum required VSN, and cannot be greater than the VSN of the firmware currently installed on the HSM.

Therefore it is possible to upgrade to a firmware version with a higher VSN that the HSM's current firmware without committing yourself to the upgrade by installing the newer firmware without using the `hsmadmin setminvsn` command. The older firmware can be reinstalled at any time provided the `hsmadmin setminvsn` command has not been run.

Ensuring you use firmware with the highest available VSN allows you to benefit from security improvements and enhanced functionality. It also prevents future downgrades of the firmware that could potentially weaken security. It is therefore recommended that the `hsmadmin setminvsn` command always be used as soon as the decision has been made not to return to the older version of the firmware.

However, you may choose to install an associated firmware that does not have the highest available VSN. For example, if you have a regulatory requirement to use FIPS-approved firmware, you should install the latest available FIPS-validated firmware, which may not

have the highest VSN.

28.2. Firmware on the installation media

Your Firmware installation media contains several sets of firmware for each supplied product. These can include the latest available:

- FIPS-approved firmware with the base VSN
- FIPS-approved firmware with a higher VSN
- Firmware awaiting FIPS approval with the base VSN
- Firmware awaiting FIPS approval with a higher VSN.

You should ensure you are using the latest firmware, unless you have a regulatory requirement to use firmware that has been FIPS validated. In the latter case, you should ensure that you are using the latest available FIPS validated firmware.

28.2.1. Primary and recovery firmware

Upgrade packages may contain updates for either the primary firmware or the recovery firmware. The same upgrade method is used in both cases. The system will automatically detect whether the firmware is primary or recovery and will load it to the correct location.

If upgrade packages are available for both primary and recovery firmware it is not recommended to upgrade them both at the same time. The recommended procedure is to always upgrade the primary firmware first. Test that the system the performs as expected and then upgrade the recovery firmware at a later date.

28.2.2. Recognising firmware files

The firmware files are stored in subdirectories within the `firmware` directory on the installation media. The subdirectories are named by product and then certification status, which can be `latest`, `fips-pending`, `fips`, or `cc`.

Firmware files for nShield HSM modules have a `.npkg` filename suffix.

The VSN of a firmware file is incorporated into its filename and is denoted by a dash and the letters "vsn" followed by the digits of the VSN. For example, `-vsn24` means the VSN is 24.

To display information about a firmware file on the installation media, enter the following command:

```
hsmadmin npkginfo /disc-name/firmware/nShield5s/status/firmware_file.npkg
```

In this command, *disc-name* is the directory on which you mounted the installation media, *status* is the certification status, and *firmware_file* is the file name.

28.3. Firmware installation overview

Normal procedure is to install firmware when the HSM is running in primary mode. If the HSM is running in recovery mode, as described in Recovery mode the procedure is identical except that the reboot caused by `hsmadmin upgrade` will cause the module to factory state and it will be necessary to run `hsmadmin enroll` before continuing with the rest of the instal lation.

If you are upgrading a module which has SEE program data or NVRAM-stored keys in its nonvolatile memory, use the `nvram-backup` utility to backup your data first.

1. Put the module in Maintenance mode.

   See Checking and changing the mode on an nShield 5s module for more about chang-ing the mode.

2. Run the `hsmadmin status` command-line utility and check the version of the firmware currently loaded.

   By default the command only displays the version of the current primary firmware. If you wish to also see the current recovery firmware version use the command line option `--json`

3. Run the `hsmadmin npkginfo` command-line utility to view information about the firmware in the upgrade file including the version and the VSN.
4. Run the `hsmadmin upgrade` command-line utility to upgrade the firmware.

   The current firmware version and the firmware version being loaded will be displayed automatically.

   The module will be programmed with the new firmware and will be automatically rebooted.

If the installation is being run from recovery mode this reboot will factory state the HSM and `hsmadmin enroll` must be run before continuing.

The module will report which internal components of the firmware have been updated. These components are pre-determined by the individual upgrade file and the internal names are intended for use by Entrust support staff only.

5. Run the `hsmadmin status` command-line utility and check the version of the firmware now loaded.
6. Put the module in initialization mode.

   See Checking and changing the mode on an nShield 5s module for more about changing the mode.

7. Initialize the module by running the command `initunit`
8. Put the module in Operational mode.

   See Checking and changing the mode on an nShield 5s module for more about changing the mode.

9. Run the `enquiry` command to verify the module is in operational state and has the correct firmware version.

   In Operational mode, the `enquiry` command shows the version number of the firmware.

28.4. After firmware installation

After you have installed new firmware and initialized the HSM, you can create a new Security World with the HSM or reinitialize the HSM into an existing Security World.

If you are initializing the HSM into a new Security World, see Creating a Security World.

If you are re-initializing the HSM into an existing Security World, see Adding or restoring an HSM to the Security World

29. SNMP monitoring agent

This appendix describes the Simple Network Management Protocol (SNMP) monitoring agent. The SNMP monitoring agent provides you with components that you can add to your (third-party) SNMP manager application.

SNMP was developed in 1988 and revised in 1996. It is currently regarded as the standard method of network management. It is widely supported and offers greater interoperability than traditional network management tools (for example, `rsh` or `netstat`). This makes it ideal for use for the large array of platforms that we support and also avoids the overhead of remote login and execution, helping to reduce network congestion and improve performance.

SNMP defines a collection of network management functions allowing management stations to gather information from, and transmit commands to, remote machines on the network. Agents running on the remote machines can take information gathered from the system and relay this information to the manager application. Such information is either requested from the underlying operating system or gained by interrogating the hardware.

Every SNMP manager adds monitor components differently. Consult the documentation supplied with your SNMP Manager application for details on how to add the MIB files.

SNMP defines the following SNMP messages:

Message	Description
`get`	This message is sent by a manager to retrieve the value of an object at the agent.
`set`	This message is sent by a manager to set the value of an object at the agent.
`trap`	This message is sent by an agent to notify a management station of significant events.

The SNMP monitoring agent is based on the open-source Net-SNMP project, version 5.7.3. More information on SNMP in general, and the data structures used to support SNMP instal lations, is available from the NET-SNMP project Web site: https://net-snmp.sourceforge.io/.

This site includes some support information and offers access to discussion e-mail lists. You can use the discussion lists to monitor subjects that might affect the operation or security of the SNMP agent or command-line utilities.

Discuss any enquiries arising from information on the NET-SNMP Web site with Support before posting potentially sensitive information to the

> NET-SNMP Web site.

29.1. Installing the SNMP agent

The SNMP agent is installed with the installation of the Security World Software and starts automatically.

29.1.1. Default installation settings

When installing Security World Software, you may be prompted to select Security World Software components from a list. If you select `all` components, then the SNMP agent is installed as part of a full Security World Software installation. The default installation directory for the nShield Management Information Base (MIB) and the SNMP configuration files (`snmp.conf` and `snmpd.conf`) is `/opt/nfast/etc/snmp/`.

29.1.2. Do you already have an SNMP agent running?

If you already have another SNMP agent running, you must configure the ports used by the agents in order to avoid conflicts before enabling the SNMP agent. A port is assigned by editing the `agentaddress` entry in the `snmpd.conf` file or by editing the `defaultPort` entry in `snmpd.conf` file. If both files have been edited, the `agentaddress` entry is `snmpd.conf` file takes priority for snmpd, and the `defaultPort` entry in `snmpd.conf` is ignored.

If no existing SNMP agent is found, the SNMP agent runs on the default port 161. If an existing SNMP agent is detected, and no SNMP agent configuration files are found (implying a fresh installation), the installer automatically configures the SNMP agent to use the first unused port above 161 by creating a new `snmpd.conf` configuration file with the appropriate directive. It then displays a message indicating the number of the port that is has selected.

If an existing SNMP agent is found and an existing SNMP agent installation exists, the installer checks the existing configuration files for an appropriate directive and warns you if one does not exist. If you need to edit these configuration files yourself, a port is assigned by editing the `agentaddress` entry in `snmpd.conf` file or editing the `defaultPort` entry in `snmpd.conf` file. If both files have been edited, the `agentaddress` entry in `snmpd.conf` file takes priority for snmpd, and the `defaultPort` entry in `snmpd.conf` is ignored.

29.1.3. Starting the SNMP agent

The SNMP agent is started automatically however it can be stopped and started manually.

To stop, start, or restart (stop and immediately start again) the SNMP daemon:

```
/opt/nfast/scripts/init.d/ncsnmpd stop|start|restart
```

See The SNMP configuration file: snmp.conf for more information on additional parameters accepted by snmpd.

29.2. Basic configuration

29.2.1. Protecting the SNMP installation

The SNMP agent allows other computers on the network to connect to it and make requests for information. The SNMP agent is based on the NET-SNMP code base, which has been tested but not fully reviewed by Entrust. We strongly recommend that you deploy the SNMP agent only on a private network or a network protected from the global Internet by appropriate network protection systems (e.g. a firewall, a network Intrusion Detection/Prevention System, etc.).

The default nShield SNMP installation allows read-only access to the Management Information Base (MIB). There is no default write access to any part of the MIB.

Every effort has been taken to ensure the confidentiality of cryptographic keys even when the SNMP agent is enabled. In particular, the nShield module is designed to prevent the theft of keys even if the security of the host system is compromised, provided that the Administrator Cards are used only with trusted hosts. Care must be used when changing the configuration of the SNMP agent.

We strongly advise that you use the SNMP User-based Security Model (USM) with Authentication and Privacy protocols selected, to ensure only authorised users can obtain information from the SNMP agent and the confidentiality and data integrity of the transferred information is protected.

Care has also been taken to ensure that malicious attackers are unable to inundate your module with requests by flooding your SNMP agent. Command results from administration or statistics commands are cached, and thus the maximum rate at which the SNMP agent sends commands to the module is throttled. For more information on setting the cache time-outs. see The SNMP configuration file: snmp.conf.

29.2.2. Configuring the SNMP agent

The Security World Software package uses various configuration files to configure its applications. This section describes the overall nature of the configuration files for the SNMP agent.

If you are installing the SNMP agent to a host that has an existing SNMP agent installation, you may need to edit the SNMP configuration files (`snmpd.conf` and `snmp.conf`) associated with the SNMP agent to change the port on which the agent listens for SNMP requests. For more information, see Do you already have an SNMP agent running?.

Make sure you protect access to the configuration files, since these contain information that defines the security parameters of the SNMP system.

By default, the SNMP configuration files are located in the `/opt/nfast/etc/snmp/` directory.

29.2.2.1. Re-reading SNMP configuration files

The SNMP agent reads its configuration files on startup, and any changes made after this point will have no effect. If new directives are added and need to be applied, the SNMP agent can be forced to re-read its configuration files with:

- An snmp `set` of integer(1) to `enterprises.nCipher.reloadConfig.0(.1.3.6.1.4.1.7682.999.0)`
- kill `-HUP` signal sent to the `snmpd` agent process
- stop then restart the SNMP agent.

29.2.2.2. The SNMP configuration file: snmp.conf

The `snmp.conf` configuration file contains directives that apply to all SNMP applications. These directives can be configured to apply to specific applications. The `snmp.conf` configuration file is not required for the agent to operate and report MIB entries.

29.2.2.3. The SNMP agent configuration file: snmpd.conf

The `snmpd.conf` configuration file defines how the SNMP agent operates. It is required only if an agent is running.

The `snmpd.conf` file can contain any of the directives available for use in the `snmp.conf` file and may also contain the following Security World Software-specific directives:

Directive	Description
`statstimeout`	This directive specifies the length of time for which statistics commands are cached. The default is 5 seconds.
`admintimeout`	This directive specifies the length of time for which administrative commands are cached. The default is 60 seconds.
`keytable`	This directive sets the initial state of the key table to `none`, `all`, or `query`. See `listKeys` in Administration sub-tree overview.
`enable_trap_zero_suffix`	This directive appends the '.0' suffix to object identifiers (OIDs) for backward compatibility. The default is `0` (disabled): the directive can be set to `1` to restore the suffix. Valid values are 0 and 1.
`memoryUsageOkThreshold`	This directive specifies the threshold (as a percentage) below which HSM memory usage is considered to be ok. The default is 0. See Memory usage monitoring for more details.
`memoryUsageHighThreshold`	This directive specifies the threshold (as a percentage) at which HSM memory usage is considered to be too high. The default is 0. See Memory usage monitoring for more details.

There may be a tolerance gap between the `memoryUsageOkThreshold` and the `memoryUsageHighThreshold` values.

The timeouts should be set to values that achieve a balance between recieving up to date information whilst preventing excessive load.

29.2.3. The SNMP agent persistent configuration file

On running the SNMP agent for the first time, the `persist` directory will be created. This contains configuration files that are maintained by the SNMP agent. This directory will be created in the following location:

```
/opt/nfast/etc/snmp/persist
```

Modifications should only be made to the persist folder's `snmp.conf` file in order to create users. The files within this directory should otherwise only be managed by the SNMP agent itself.

User creation can be performed with the `createUser` directive. See USM users. On initialization of the agent the information is read from the file and the lines are removed (eliminating the storage of the master password for that user) and replaced with the key that is derived from it. This key is a localised key, so that unlike the password, if it is stolen it can not be used to access other agents.

Do not modify the persistent `snmpd.conf` file while the agent is running. The file is only read on initialization of the agent and it is overwritten when the SNMP agent terminates. Any changes made to this file while the SNMP agent directives is running will be lost. The SNMP agent should be stopped prior to adding `createUser` directories to the configu ration file.

29.2.4. Agent Behaviour

There are a small number of directives that control the behaviour of the SNMP Agent when considering it as a daemon providing a network service.

29.2.5. agentaddress directive

The listening address(es) that the SNMP Agent will use are defined by the `agentaddress` directive. It takes a comma separated list of address specifiers where an address specifier consists of one or more of:

- a transport specifier udp: or tcp
- a hostname or IPv4 address
- a port number (e.g. :161 or :1161).

The default behaviour is to listen on UDP port 161 on all IPv4 interfaces (i.e. equivalent to udp:161).

```
agentaddress localhost : 161,tcp:1161
```

`agentaddress` will listen on UDP port 161, but only on the loopback interface (the port specifi cation ":161" is not strictly necessary as this is the default port). It will also listen on TCP port 1161 on all IPv4 interfaces.

29.2.6. agentgroup and agentuser directives

The user and group that the SNMP Agent changes to after opening the listening port(s) are defined using the `agentgroup` and `agentuser` directives. The following must be used:

```
agentgroup ncsnmpd
agentuser ncsnmpd
```

29.2.7. System information

Most of the scalar objects in the .iso.org.dod.internet.mgmt.mib-2.system sub-tree can be configured.

```
sysLocation STRING
sysContact STRING
sysName STRING
```

The three directives above set the system location, contact or name for the SNMP Agent respectively. Ordinarily these objects are writable via a suitably authorised SNMP SET request, however, specifying one of these directives in the configuration file makes the cor responding object read-only.

```
sysServices INTEGER
```

Sets the value of the sysService.0 object. RFC1213 defines how the integer value is calculated.

```
sysDescr STRING
sysObjectID OID
```

The two directives above set the system description and object ID for the agent. These objects are not SNMP-writable, but these directives can be used by a network administrator to configure suitable values for them.

29.3. USM users

The SNMPv3 protocol supports a User based Security Model as defined in RFC-3414. USM provides authentication and privacy (encryption) functions and operates at the message level allowing for the following security level to be used with SNMPv3:

- Communication without authentication and privacy (`noauth`)
- Communication with authentication and without privacy (`auth`)
- Communication with authentication and privacy (`priv`).

Within this document the three possible security levels are referred to as `noauth`, `auth` and `priv`. However, other forms are sometimes used within the NET-SNMP and the equivalents are:

Security level	Equivalents
`noauth`	`noauthnopriv`

Security level	Equivalents
auth	authnopriv
priv	authpriv

Users can be added to the SNMP configuration with the createUser directive, defining the security mechanisms to be used.

```
createUser [-e ENGINEID] username [SHA authpassphrase] [AES privpassphrase]
```

It would not normally be necessary to specify the engine ID, but if it is specified, ENGINEID is defined as a hexadecimal string of octets starting with the 0x prefix. The encoding of the engine ID is defined in the description of SnmpEngineID from RFC3411. The following recommendations should be followed when defining the security parameters for SNMPv3:

- Select a 'Security Level' of Priv, (authpriv) or auth (authNoPriv).
  - Priv is the preferred 'Security Level', since this will provide both data source authentication and confidentially protection for the SNMP messages.
  - auth is the minimum 'Security Level' that should be selected, since this will ensure that SNMP data sent/received has not been tampered with and has been sent from an authorised entity.
- Define separate authpassphrase and privpassphrase.
  - It is good security practice to have key separation.
- Use randomly generated passphrases which contain upper and lower case characters, numbers and symbols (e.g. ASCII characters 0x20 - 0x7E).
  - This should give an entropy per character of 6.57bits,
- Use either 15 char for 96 bits of security strength keys and 20 char for 128 bits security strength keys.
  - The minimum length of both auth and priv passphrases is eight characters.
  - If a random passphrase is not used, consult NIST SP800-63-2 - Appendix A to determine the security strength of the password and the resultant keys. See https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-63-2.pdf.

MD5 and DES are not supported or enabled in the nShield distribution of SNMP. Only SHA may be used for authentication, and only AES may be used for privacy (encryption).

It is strongly recommended that createUser directives be added to the persist/snmpd.conf file, so that the passphrases are not available after the SNMP agent is installed. See USM users. The user can then be referenced in access control directives(s) after which it can be

used.

29.4. Traditional access control

Most simple access control requirements can be specified using the directives `rouser` /`rwuser` (for SNMPv3) or `rocommunity`/`rwcommunity` (for SNMPv1 or SNMPv2c).

```
rouser [-s usm] USERNAME [noauth | auth | priv [OID | -V VIEW [CONTEXT]]
rwuser [-s usm] USERNAME [noath | auth | priv [OID | -V VIEW [CONTEXT]]
```

These directives specify that an SNMPv3 user (USERNAME) will be allowed read-only or read-write access respectively. The default (unspecified) security level is `auth`, which is the recommended minimum security level (see above). It is not recommended to use the usm security level `noauth`, where all SNMP messages are unauthenticated and any tampering of the message cannot be detected. Using `noauth` will reduce the security of the SNMP messages to the level of SNMPv1 or SNMPv2c.

OID restricts access for that user to the subtree rooted at the given OID.

`VIEW` restricts access for that user to the specified View-based Access Control Model (VACM) view name. An optional context can also be specified, or `context` to denote a context prefix. If no context field is specified (or the token * is used), the directive will match all possible contexts. (Contexts are a mechanism within SNMPv3 whereby an agent can support parallel versions of the same MIB objects, referring to different underlying data sets.)

A security model can be specified with `-s SECMODEL` however the default security model `usm` is the only security model which is supported in the nShield distribution of SNMP.

Example:

- Read-only user with access to the full OID tree requiring authentication as a minimum:

  ```
  rouser userl
  ```

- Or

  ```
  rouser -s usm user1 auth .1
  ```

- Read-only user with access to the nShield MIB allowing unauthenticated requests:

  ```
  rouser user2 noauth .1.3.6.1.4.1.7682
  ```

- Read-write user with access to the full OID tree requiring authentication as a minimum:

```
rwuser user3
```

Or

```
rwuser user3 auth .iso
```

- Read-write user with access to the snmpVacmMIB subtree requiring authentication and encryption:

```
rwuser user4 priv snmpVacmMIB
```

Or

```
rwuser user4 priv .1.3.6.1.6.3.16
```

```
rocommunity COMMUNITY [SOURCE [ OID | -V VIEW [CONTEXT]]
rwcommunity COMMUNITY [SOURCE [ OID | -V VIEW [CONTEXT]]
```

Specifies an SNMPv1 or SNMPv2c community that will be allowed read-only (`GET` and `GET-NEXT`) or read-write (`GET`, `GETNEXT` and `SET`) access respectively. By default, this will provide access to the full OID tree for such requests, regardless of where they were sent from. `SOURCE` allows access either from a particular range of source addresses, or globally (`default`). A restricted source can either be a specific hostname or address (e.g. `localhost` or 127.0.01), or a subnet - represented as IP/MASK (e.g. 10.10.10.0/255.255.255.0), or IP/BITS (e.g. 10.10.10.0/24).

OID `VIEW` and `CONTEXT` are as defined for `rouser` and `rwuser`.

Example:

- Setting up a read-only community named `public` that can be accessed by any user with the community name:

```
rocommmunity public
```

- Setting up a read/write community named `private` that can only be accessed from the machine on which the agent is running:

```
rocommmunity private localhost
```

In each case, only one directive should be specified for a given SNMPv3 user, or community string. It is not appropriate to specify both `rouser` and `rwuser` directives referring to the same SNMPv3 user (or equivalent community settings). The `rwuser` directive provides all the access of `rouser` (as well as allowing `SET` support). The same applies to `rwcommunity` and `rocommunity`.

More complex access requirements (such as access to two or more distinct OID subtrees, or different views for `GET` and `SET` requests) should use VACM configuration directives.

29.5. VACM configuration

The full flexibility of the VACM, for example allowing access to two or more distinct OID sub trees, or different access requirements for reading and writing, is available using four config uration directives - `com2sec`, `group`, `view` and `access`. The directives essentially define who has access and what they have access to using four directives. The first two directives (`com sec2sec` and `group`) define the who, while the last two (`view` and `access`) define the what.

```
Com2sec [-Cn CONTEXT] SECNAME SOURCE COMMUNITY
```

Maps an SNMPv1 or SNMPv2c community string to a security name. As it defines the com- munity and maps it to a security name, `rocommunity`/`rwcommunity` directives are not required when using the directive.

`SECNAME` is the security name to be defined.

`SOURCE` is as defined for the `rocommunity`/`rwcommunity` directives above.

`COMMUNITY` defines the community name to be mapped to the security name. The same com munity string can be specified in several separate directives with different source tokens, and the first source/community combination that matches the incoming request will be selected. Various source/community combinations can also map to the same security name.

`CONTEXT` if defined (using `-Cn`), means that the community string will be mapped to a secu- rity name in the named SNMPv3 context. Otherwise the default context ("") will be used.

Example:

Creating three SNMPv1/v2c community names (`private`, `public` and `ltd`), where `private` and `ltd` only allow requests from the machine on which the SNMP Agent is running (note lines beginning with a # in `snmpd.conf` are treated as comments):

```
#        [-Cn CONTEXT] SECNAME     SOURCE    COMMUNITY
```

```
com2sec   ""            sec_private localhost private
com2sec                 sec_public  default   public
com2sec                 sec_limited localhost ltd
```

```
group GROUP v1 | v2c | usm SECNAME
```

Maps a security name (in the specified security model) into a named group. Several group directives can specify the same group name, allowing a single access setting to apply to several users and /or community strings. Note that groups must be set up for the two community-based models separately - a single `com2sec` directive will typically be accompanied by two `group` directives.

- `GROUP` is the group name being defined/added to.
- `v1`, `v2c` or `usm` defines the security model to which the definition relates.
- `SECNAME` is the security (USM) user name or security name defined by `com2sec` to be added to the group.

Example:

Creating three groups (`grp_private`, `grp_public`, `grp_limited`) for three USM users (`user1`, `user2` and `user3`) and the three communities shown in the `com2sec` example above:

```
# GROUP v1|v2c|usm SECNAME
Ggroup grp_private v1 sec_private
group grp_private v2c sec_private
group grp_private usm user1

group grp_public v1 sec_public
group grp_public v2c sec_public
group grp_public usm user2

group grp_limited v1 sec_limited
group grp_limited v2c sec_limited
group grp_limited usm user3
```

```
view VNAME included | excluded OID [MASK]
```

Defines a named `view` - a subset of the overall OID tree. This is most commonly a single sub tree, but several `view` directives can be given with the same view name (`VNAME`), to build up a more complex collection of OIDs. An optional mask can also be specified, providing a means of indicating which parts of the OID must be matched.

`VNAME` is the view being modified.

`included` | `excluded` allows you to define whether the view includes or excludes the subtree, allowing the definition of a more complex view (e.g. by excluding certain sensitive objects from an otherwise accessible subtree).

MASK is an optional list of hex octets (separated by '.' or ':') whose bits indicate which OID sub-identifiers to match against. So for example if we assume we have on OID with 11 sub-identifiers (.1.3.6.1.x.y.z.table.entry.column.1) where the last four relate to a table, an entry, a column and index 1, specifying a MASK value of " FF.A0 " (i.e. 1111111110100000) maps to this OID as follows:

```
.3.6.1.x.y.z.table.entry.column.1
1 1 1 1 1 1 1   1     1     0    1
```

i.e. this mask means all parts of the OID except the column must match, therefore defining a view to any column of the first row of the table.

By including and excluding various aspects of the full OID tree, it is possible to define fine grained visibility within a view's definition.

Example:

Creating five views where vw_sysContact only has access to the system.sysContact.0 OID, vw_nCipher only has access to the MIB, vw_global has access to the full OID tree, vw_nCipher_stats only has access to nCipher.nC-series.statistics and vw_nCipher_admin only has access to nCipher.nC-series.administration:

```
#     VNAME             included|excluded   OID                                        [MASK]
view  vw_sysContact     excluded            .1
view  vw_sysContact     included            system.sysContact.0                        FF.80

view  vw_nCipher        excluded            .iso
view  vw_nCipher        included            .1.3.6.1.4.1.7682

view  vw_global         included            .1

view  vw_nCipher_stats  excluded            .1
view  vw_nCipher_stats  included            enterprises.nCipher.nC-series.statistics

view  vw_nCipher_admin  excluded            .1
view  vw_nCipher_admin  included            enterprises.nCipher.nC-series.administration
```

```
access GROUP CONTEXT any | v1 | v2c | usm noauth | auth | priv exact | prefix READ WRITE NOTIFY
```

Maps from a group of users/communities (with a particular security model and minimum security level, and specific context) to one of three views, depending on the request being processed.

GROUP is a group name defined by the group directive and specifies the group that access is being defined for.

CONTEXT specifies the context for the access (the default context is the empty string ""). The context of incoming requests must match against the context either exactly or by prefix, as

specified by the choice of `exact` | `prefix` made in this directive.

`any`, `v1`, `v2c`, or `usm` define the security model to which this definition relates.

`noauth` | `auth` | `priv` define the security level to which this definition relates. For `v1` or `v2c` access, this will need to be `noauth` as these protocols do not support authentication.

`exact` | `prefix` specify how `CONTEXT` should be matched against the context of the incoming request, either an exact match to `CONTEXT`, or prefixed by `CONTEXT`.

`READ`, `WRITE` and `NOTIFY` specifies the view to be used for `GET*`, `SET` and `TRAP`/`INFORM` requests (although the `NOTIFY` view is not currently used). The keyword `none` is used if there is to be no access for that type of request.

Example:

Specifying that:

- SNMPv1 requests using the public community only have read access to the enterprises.nCipher.nC-series.statistics subtree,
- SNMPv2c requests using the public community only have read access to the enterprises.nCipher.nC-series.administration.subtree,
- SNMPv3 requests using the user2 USM user, must as a minimum be authenticated, and have read, notify access to the nCipher MIB (i.e. enterprises nCipher)
- SNMPv3 requests using the user1 USM user, must as a minimum be authenticated and encrypted, and have read, write and notify access to the full OID tree. Note that since requests must be authenticated and encrypted as a minimum, SNMPv1 and v2c requests using the private community cannot be made even though the community is included in grp_private.
- SNMPv1 and SNMPv2 requests using the ltd community and SNMPv3 requests using the user3 USM user, do not require to be authenticated or encrypted, and have read, write access to the system.sysContact.0 OID.

```
#      GROUP       CONTEXT  SECMODEL  LEVEL     PREFIX  READ             WRITE     NOTIFY
access grp_public  ""       v1        noauth    exact   vw_nCipher_stats  none      none
access grp_public  ""       v2c       noauth    exact   vw_nCipher_admin  none      none
access grp_public  ""       usm       auth      exact   vw_nCipher        none      vw_nCipher
access grp_private ""       any       priv      exact   vw_global         vw_global
vw_global
access grp_limited ""       any       noauth    exact   vw_sysContact     vw_sysContact
none
```

29.6. Trap Configuration

The distribution of SNMP supports SNMPv1, SNMPv2 and SNMPv3 traps. Control over

these traps is defined with a number of directives:

29.6.1. SNMPv1 and SNMPv2 traps

```
trapcommunity COMMUNITY
```

Defines the default community to be used when sending SNMPv1 or SNMPv2 traps. Note that this directive must be used prior to a `trapsink` or `trap2sink` directive that wishes to use this community.

`COMMUNITY` the community name to be used.

Example:

```
trapcommunity traps
```

```
trapsink HOST [COMMUNITY [PORT]]
trap2sink HOST [COMMUNITY [PORT]]
```

Defines a destination for SNMPv1 or SNMPv2 traps generated by the agent.

`HOST` is an address specifier defining the network target that traps will be sent to. It consists of an optional transport specifier (`udp` (default if not specified) or `tcp`) followed by a host-name or IPv4 address followed by an optional port number, deliminated by colons ":". (e.g. `localhost` or tcp:192.168.137.2:163).

`COMMUNITY` if specified will be the community name used for the traps. If it is not specified, the most recently specified `trapcommunity` will be used.

`PORT` allows for port-number to be defined if it is not present as part of the `HOST` specification. If no port is defined, the default port number of 162 will be used.

When a TCP transport specifier is used the SNMP agent establishes the TCP connection with the trap manager at start-up. Therefore the trap manger must be started before the SNMP agent otherwise an error is reported for the line in the `snmpd.conf` file which defines the trap manager.

Likewise when the TCP connection between the SNMP agent and the trap manager is dropped, traps are lost. Therefore it is inadvisable to use TCP instead of UDP for the transport specifier of trap managers.

If TCP is used for the connection between the SNMP agent and the trap manager and the connection is lost, to re-establish the connection the SNMP agent must be restarted, with

the trap manager running and able to accept a TCP connection from the SNMP agent.

For issues with the trap manager accepting TCP connections from a SNMP agent refer to trap manager documentation.

Example:

```
trap2sink udp:192.168.137.220:162 traps
```

29.6.2. SNMPv3 traps

```
trapsess [SNMPCMD_ARGS]  HOST
```

Defines the configuration for a trap. This is the only way to define SNMPv3 traps and it is an alternative method for defining SNMPv1 and SNMPv2 traps.

`SNMPCMD_ARGS` are arguments that would be used for an equivalent `snmptrap` command. So for example to send an SNMPv3 trap as USM user `user1` with authentication and encryption, the value `-v3 -u user1 -1 priv` would be used.

`HOST` see host definition for `trap2sink` above.

Example:

```
trapsess -v3 -u user1 -1 priv udp:192.168.137.220:162
trapsess -v2c -c public 192.168.137.221:162
```

29.7. Using the SNMP agent with a manager application

The nShield SNMP monitoring agent provides MIB files that can be added to your (third-party) SNMP manager application.

29.7.1. Manager configuration

The manager application is the interface through which the user is able to perform network management functions. A manager communicates with agents using SNMP primitives (`get`, `set`, `trap`) and is unaware of how data is retrieved from, and sent to, managed devices. This form of encapsulation creates the following:

- The manager is hidden from all platform specific details
- The manager can communicate with agents running on any IP-addressable machine.

As a consequence, manager applications are generic and can be bought off the shelf. You may already be running SNMP managers, and if so, you can use them to query the SNMP agent.

The manager is initially unaware of the MIB tree structure at a particular node. Managed objects can be retrieved or modified, but only if their location in the tree is known.

It is more useful if the manager can *see* the MIB tree present at each managed node. The MIB module descriptions for a particular node must be parsed by a manager-specific MIB compiler and converted to configuration files. These files are read by the manager application at run time.

The SNMP agent is designed to monitor all current nShield modules, working with all supported versions of nShield firmware (contact Support for details of supported firmware).

29.7.2. MIB module overview

A large proportion of the SNMP system is fully specified by the structure of the MIB; the behavior of the agent depends on relaying information according to the layout of the MIB.

The MIB module resides at a registered location in the MIB tree determined by the Internet Assigned Numbers Authority (IANA). The private enterprise number of 7682 designated by the IANA corresponds to the root of the branch, and by convention this (internal) node is the company name.

The MIB module groups logically related data together, organizing itself into a classification tree, with managed objects present at leaf nodes. The nC-series node (`enterprises.nCipher.nC-series`) is placed as a sub-tree of the root (`enterprises.nCipher`); this allows future product lines to be added as additional sub-trees. The structure of the tree underneath the registered location is vendor-defined, and this specification defines the structure chosen to represent Security World Software-specific data.

The MIB file can be found in the following location:

```
/opt/nfast/etc/snmp/mibs/ncipher-mib.txt
```

29.7.3. MIB functionality

The MIB module separates module information into the following categories:

- Retrieval of status and information about installed nC-series modules

- Retrieval of live statistics of performance of installed nC-series modules

These categories form the top-level nodes of the sub-tree; the functionality of the first category is in the administration sub-tree, and the second category is in the statistics sub-tree. The top-level tree also contains three items that it would be useful to check at-a-glance:

Node name	R/W	Type	Remarks
`hardserverFailed`	R	TruthValue	`True` if the remote hardserver is not running. If the hardserver is not running, then most of the rest of the information is unreliable or missing.
`modulesFailed`	R	TruthValue	`True` if any modules have failed.
`load`	R	Unsigned32	Percentage of total available capacity currently utilized.

29.7.3.1. Traps

The traps sub-tree (`enterprises.nCipher.nC-series.nC-traps`) contains traps that the SNMP agent sends when certain events occur. For details on configuring traps, see USM users.

The following table gives details of the individual traps:

Node name	Description
`hardserverAlert`	This trap is sent when the hardserver fails or is shut down.
`hardserverUnAlert`	This trap is sent when the hardserver restarts.
`moduleAlert`	This trap is sent when a module fails.
`moduleUnAlert`	This trap is sent when a module is restarted after a failure.
`psuAlert`	This trap is sent when a PSU fails.
`psuUnAlert`	This trap is sent when a previously-failed PSU is working again.
`fanfailureAlert`	This trap is sent when a fan fails.
`fanfailureUnAlert`	This trap is sent when a previously-failed fan is working again.
`memoryUsageHighAlert`	This trap is sent when the HSM memory usage high threshold has been reached or exceeded by an HSM. See section on Memory usage monitoring below for more details.
`memoryUsageOkAlert`	This trap is sent when the memory usage for an HSM falls below the HSM memory usage ok threshold. See section on Memory usage monitoring below for more details.

Some traps can take up to five minutes to be received.

Other generic Net-SNMP traps may also be received. These include the two below, see Net-SNMP project website for more details.

Net-SNMP trap name	Description
SNMPv2-MIB::coldStart	This trap is sent when the SNMP agent is started
NET-SNMP-AGENT-MIB::nsNotifyShutdown	This trap is sent when the SNMP agent is stopped

29.7.4. Memory usage monitoring

The HSM memory usage thresholds and memory usage traps provide a mechanism to moni tor HSM memory usage for HSMs in which the SNMP agent's client computer are enrolled.

With memory usage monitoring enabled, there will be a `memoryUsageHighAlert` trap sent each time the currently in-use `memoryUsageHighThreshold` is reached or exceeded by an HSM.

With memory usage monitoring enabled, a `memoryUsageHighAlert` trap is also sent:

- If the SNMP agent starts up and recognises that there are HSMs in a high memory usage state or,
- If HSMs in a high memory usage state are enrolled or,
- If the SNMP agent loses and then re-gains contact with the local hardserver which is connected to HSMs in a high memory usage state or,
- If failed HSMs in a high memory usage state then recover.

For each of the four scenarios above, one `memoryUsageHighAlert` trap will be sent for each HSM in a high memory usage state.

With memory usage monitoring enabled, there will be a `memoryUsageOkAlert` trap sent each time the memory usage for an HSM falls below the currently in-use `memoryUsageOkThresh-old`.

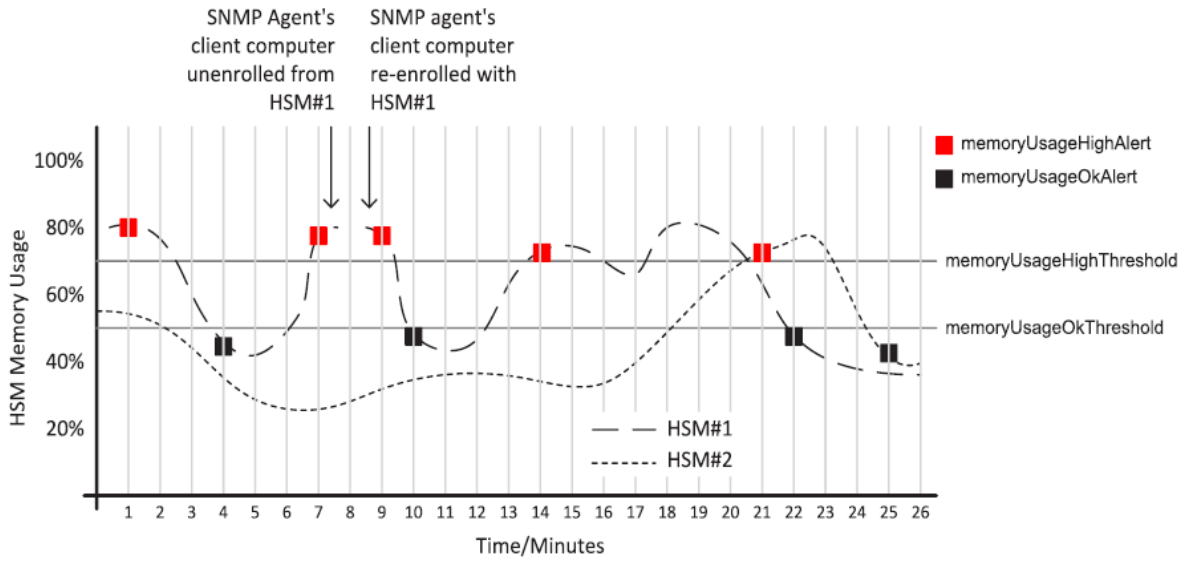
The value for `memoryUsageOkThreshold` is read from the snmpd.conf file on starting the SNMP agent and is used provided it contains an integer value in the range 0 to 100 (inclu-sive); otherwise, the default value of 0 is used. The value for `memoryUsageHighThreshold` is processed in the same way.

Memory usage monitoring is enabled unless the in-use values for `memoryUsageOkThreshold` and `memoryUsageHighThreshold` are both 0 or the in-use values are such that `memoryUsageOk`

`Threshold` > `memoryUsageHighThreshold`.

For example, in `snmpd.conf`, if `memoryUsageOkThreshold` is assigned an invalid value and `memoryUsageHighThreshold` is assigned a valid value of say 75%, then memory usage monitoring will be enabled and the values 0% and 75% will be used respectively.

An example of memory usage monitoring by an SNMP agent on a client computer enrolled with 2 HSMs is given below:



29.7.5. Administration sub-tree overview

The administration sub-tree (`enterprises.nCipher.nC-series.administration`) contains information about the permanent state of the hardserver and the connected modules. It is likely that most of the information in this branch rarely changes over time, unlike the `statistics` branch. The information given in the administration sub-tree is mostly acquired by the `NewEnquiry` command and is supplied both per-module and (where appropriate) aggregated over all modules.

The following table gives details of the individual nodes in the administration sub-tree:

Node name	R/W	Type	Remarks
`hardserverRunning`	R	Enum 1: Running 2: NotRunning	This variable reflects the current state of the hardserver (`Running` or `NotRunning`).
`noOfModules`	R	Gauge32	Number of nC-series modules.
`hsVersion`	R	DisplayString	Hardserver version string.

Node name	R/W	Type	Remarks
globalSpeedIndex	R	Gauge32	Number of 1024-bit signatures each second.
globalminQ	R	Gauge32	Minimum recommended queue.
globalmaxQ	R	Gauge32	Maximum recommended queue.
SecurityWorld	R	TruthValue	True if a Security World is installed and operational.
swState	R	DisplayString	Security World display flags, as reported by nfkminfo.
listKeys	R/W	Integer 1: none 2: all 3: query 4: resetquery	Controls the behavior of the key table (switch off, display all keys, enable individual attribute queries, clear the query fields). Displaying all keys can result in a very long list.
serverFlags	R	DisplayString	Supported hardserver facilities (the NewEnquiry level 4 flags).
remoteServerPort	R	Gauge32	TCP port on which the hardserver is listening.
swGenTime	R	DisplayString	Security World's generation time.
swGeneratingESN	R	DisplayString	ESN of the module that generated the Security World.

listKeys can be preset using the keytable config directive in snmpd.conf file (see The SNMP configuration file: snmp.conf).

29.7.5.1. Security World hash sub-tree

The following table gives details of the nodes in the Security World hash sub-tree (enterprises.nCipher.nC-series.administration.swHashes):

Node name	R/W	Type	Remarks
hashKNSO	R	MHash	Hash of the Security Officer's key.
hashKM	R	MHash	Hash of the Security World key.
hashKRA	R	MHash	Hash of the recovery authorization key.

Node name	R/W	Type	Remarks
hashKRE	R	MHash	Hash of the recovery key pair.
hashKFIPS	R	MHash	Hash of the FIPS authorization key.
hashKMC	R	MHash	Hash of the module certification key.
hashKP	R	MHash	Hash of the passphrase replacement key.
hashKNV	R	MHash	Hash of the nonvolatile memory (NVRAM) authorization key.
hashKRTC	R	MHash	Hash of the Real Time Clock authorization key.
hashKDSEE	R	MHash	Hash of the SEE Debugging authorization key.
hashKFTO	R	MHash	Hash of the Foreign Token Open authorization key.

29.7.5.2. Security World quorums sub-tree

The following table gives details of the nodes in the Security World quorums sub-tree (`enterprises.nCipher.nC-series.administration.swQuorums`):

Node name	R/W	Type	Remarks
adminQuorumK	R	Gauge32	The default quorum of Administrator cards.
adminQuorumN	R	Gauge32	The total number of cards in the ACS.
adminQuorumM	R	Gauge32	The quorum required for module reprogramming.
adminQuorumR	R	Gauge32	The quorum required to transfer keys for OCS replacement.
adminQuorumP	R	Gauge32	The quorum required to recover the passphrase for an Operator card.
adminQuorumNV	R	Gauge32	The quorum required to access non volatile memory (NVRAM).
adminQuorumRTC	R	Gauge32	The quorum required to update the Real Time Clock.

Node name	R/W	Type	Remarks
adminQuorumDSEE	R	Gauge32	The quorum required to view full SEE debug information.
adminQuorumFTO	R	Gauge32	The quorum required to use a Foreign Token Open Delegate Key.

29.7.5.3. Module administration table

The following table gives details of the nodes in the module administration table (enterprises.nCipher.nC-series.administration.moduleAdminTable):

Node name	R/W	Type	Remarks
moduleAdminIndex	R	Gauge32	Module number of this row in the table.
mode	R	Integer 1: Operational 2: Pre-init 3: Init 4: Pre-maint 5: Maint 6: AccelOnly 7: Failed 8: Unknown	Current module state.
fwVersion	R	DisplayString	Firmware version string.
speedIndex	R	Gauge32	Speed index (approximate number of 1024-bit modulo exponentiation operations possible per second) of module
minQ	R	Gauge32	Module minimum recommended queue length
maxQ	R	Gauge32	Module maximum recommended queue length
serialNumber	R	DisplayString	Module Electronic Serial Number (ESN).
productName	R	DisplayString	

Node name	R/W	Type	Remarks
hwPosInfo	R	DisplayString	Hardware bus/slot info (such as PCI slot number).
moduleSecurityWorld	R	TruthValue	Indicates whether or not the module is in the current SW.
smartcardState	R	DisplayString	Description of smart card in slot (empty, unknown card, admin/operator card from current SW, failed). N/A for acceleration only modules.
moduleSWState	R	Integer 1: Unknown 2: Usable 3: MaintMode 4: Uninitialized 5: Factory 6: Foreign 7: AccelOnly 8: Failed 9: Unchecked 10: InitMode 11: PreInitMode 12: Unverified 13: UnusedTableEntry	Current module and Security World state.
moduleSWFlags	R	DisplayString	Security World flags for this module.
hashKML	R	MHash	Hash of the module's secret key.
moduleFeatures	R	DisplayString	Features enabled on this module.
moduleFlags	R	DisplayString	Like serverFlags, but for each module.
versionSerial	R	Gauge32	Firmware Version Security Number (VSN); see Version Security Number (VSN).
hashKNETI	R	MHash	K_{NETI} hash, if present.

Node name	R/W	Type	Remarks
longQ	R	Gauge32	Max. rec. long queue.
connectionStatus	R	DisplayString	Connection status (for imported modules).
connectionInfo	R	DisplayString	Connection information (for imported modules).
machineTypeSEE	R	DisplayString	SEE machine type.

29.7.5.4. Slot administration table

The following table gives details of the nodes in the slot administration table (enterprises.nCipher.nC-series.administration.slotAdminTable):

Node name	R/W	Type	Remarks
slotAdminModuleIndex	R	Integer32	Module number of the module containing the slot.
slotAdminSlotIndex	R	Integer32	Slot number (1-based, unlike nCore which is 0-based).
slotType	R	Integer 1: Datakey 2: Smart card 3: Emulated 4: Soft token 5: Unconnected 6: Out of range 7: Unknown	Slot type.
slotFlags	R	DisplayString	Flags referring to the contents of the slot (from slotinfo).

Node name	R/W	Type	Remarks
slotState	R	Integer 1: Unused 2: Empty 3: Blank 4: Administrator 5: Operator 6: Unidentified 7: Read error 8: Partial 9: Out of range	Partial refers to cards in a partially-created card set.
slotListFlags	R	DisplayString	Flags referring to attributes of the slot (from getslotlist).
slotShareNumber	R	Gauge32	Share number of card currently in slot, if present.
slotSharesPresent	R	DisplayString	Names of shares present in card currently in slot.

29.7.5.5. Card set administration table

The following table gives details of the nodes in the card set administration table (enterprises.nCipher.nC-series.administration.cardsetAdminTable):

Node name	R/W	Type	Remarks
hashKLTU	R	MHash	Hash of the token protected by the card set.
cardsetName	R	DisplayString	
cardsetK	R	Gauge32	Required number of cards in the card set.
cardsetN	R	Gauge32	Total number of cards in the card set.
cardsetFlags	R	DisplayString	Other attributes of the card set.
cardsetNames	R	DisplayString	Names of individual cards, if set.

Node name	R/W	Type	Remarks
cardsetTimeout	R	Gauge32	Token time-out period, in seconds, or 0 if none.
cardsetGenTime	R	DisplayString	Generation time of card set.

29.7.5.6. Key administration table

The key administration table is visible as long as the listKeys node in the administration sub-tree is set to a value other than none.

The following table gives details of the nodes in the key administration table (enterprises.nCipher.nC-series.administration.keyAdminTable):

Node name	R/W	Type	Remarks
keyAppname	R	DisplayString	Application name.
keyIdent	R	DisplayString	Name of key, as generated by the application.
keyHash	R	MHash	
keyRecovery	R	Integer 1: Enabled 2: Disabled 3: No key 4: Unknown 5: Invalid 6: Unset	The value unset is never returned by the key table. If you set the value unset, the keys are not filtered based on any of the attributes.
keyProtection	R	Integer 1: Module 2: Cardset 3: No key 4: Unknown 5: Invalid 6: Unset	The value unset is never returned by the key table. If you set the value unset, the keys are not filtered based on any of the attributes.

Node name	R/W	Type	Remarks
keyCardsetHash	R	MHash	Hash of the card set protecting the key, if applicable.
keyFlags	R	DisplayString	Certificate and public key flags.
keyExtraEntries	R	Gauge32	Number of extra key attributes.
keySEEInteg	R	DisplayString	SEE integrity key, if present.
keyGeneratingESN	R	DisplayString	ESN of the module that generated the key, if present.
keyTimeLimit	R	Gauge32	Time limit for the key, if set.
keyPerAuthUseLimit	R	Gauge32	Per-authentication use limit for the key.

29.7.5.7. Key query sub-tree

The key query sub-tree is used if the `listKeys` node in the administration sub-tree is set to `query`.

If these values are set, they are taken as required attributes for filtering the list of available keys; if multiple attributes are set, the filters are combined (AND rather than OR).

The following table gives details of the nodes in the key query sub-tree (`enterprises.nCipher.nC-series.administration.keyQuery`):

Node name	R/W	Type	Remarks
keyQueryAppname	R/W	DisplayString	Application name.
keyQueryIdent	R/W	DisplayString	Name of key, as generated by the application.
keyQueryHash	R/W	DisplayString	
keyQueryRecovery	R/W	Integer 1: Enabled 2: Disabled 3: No key 4: Unknown 5: Invalid 6: Unset	The value `unset` is never returned by the key table. If you set the value `unset`, the keys are not filtered based on any of the attributes.

Node name	R/W	Type	Remarks
keyQueryProtection	R/W	Integer 1: Module 2: Cardset 3: No key 4: Unknown 5: Invalid 6: Unset	The value unset is never returned by the key table. If you set the value unset, the keys are not filtered based on any of the attributes.
keyQueryCardsetHash	R/W	DisplayString	Hash of the card set protecting the key, if applicable.
keyQueryFlags	R/W	DisplayString	Certificate and public key flags.
keyQueryExtraEntries	R/W	Gauge32	Number of extra key attributes.
keyQuerySEEInteg	R/W	DisplayString	SEE integrity key, if present.
keyQueryGeneratingESN	R/W	DisplayString	ESN of the module that generated the key, if present.
keyQueryTimeLimit	R/W	Gauge32	Time limit for the key, if set (0 for no limit).
keyQueryPerAuthUseLimit	R/W	Gauge32	Per-authentication use limit for the key (0 for no limit).

29.7.6. Statistics sub-tree overview

The statistics sub-tree (enterprises.nCipher.nC-series.statistics) contains rapidly changing information about such topics as the state of the nShield modules, the work they are doing, and the commands being submitted.

Do not rely on information returned from the agent to change instantaneously on re-reading the value. To avoid loading the nShield module with multiple time-consuming statistics commands, the agent can choose to cache the values over a specified period. You can configure this period in the agent configuration file see The SNMP configuration file: snmp.conf.

29.7.6.1. Statistics sub-tree

The following table gives details of the nodes in the statistics sub-tree, and the module statistics table (`enterprises.nCipher.nC-series.statistics.moduleStatsTable`):

Node name	R/W	Type	Remarks
`moduleStatsIndex`	R	Integer	Module number of this row (for `moduleStatsTable`).
`hsuptime`	R	Counter32	Uptime of the hardserver.
`cmdCountAll`	R	Counter32	Returned aggregated for all modules and all commands.
`cmdBytesAll`	R	Counter32	
`cmdErrorsAll`	R	Counter32	Returned as for `cmdCount`, returned value is combined module errors added to hardserver marshalling/unmarshalling errors.
`replyCountAll`	R	Counter32	
`replyBytesAll`	R	Counter32	
`replyErrorsAll`	R	Counter32	See notes above for `cmdErrors`.
`clientCount`	R	Gauge32	
`maxClients`	R	Counter32	
`deviceFails`	R	Counter32	
`deviceRestarts`	R	Counter32	
`outstandingCmds`	R	Counter32	Total number of outstanding commands over all modules.
`load[All]`	R	Counter32	

29.7.6.2. Module statistics table

The following table gives details of the nodes in the module statistics table (`enterprises.nCipher.nC-series.statistics.moduleStatsTable`):

Node name	R/W	Type	Remarks
`moduleStatsIndex`	R	Integer	Module number of this row (for `moduleStatsTable`).
`uptime`	R	Counter32	Uptime of the module.
`cmdCount`	R	Counter32	Returned aggregated for all commands.

Node name	R/W	Type	Remarks
cmdBytes	R	Counter32	
cmdErrors	R	Counter32	Returned as for cmdCount all the different error states aggregated into one counter.
replyCount	R	Counter32	
replyBytes	R	Counter32	
replyErrors	R	Counter32	See notes above for cmdErrors.
loadModule	R	Counter32	
loadModule	R	Counter32	
objectCount	R	Gauge32	
clock	R	DisplayString	Depending on the module settings, this can require K_{NSO} permissions to read (and therefore depend on the installation parameters of the agent).
nvRAMInUse	R	Gauge32	
volatileRAMInUse	R	Gauge32	
tempSP	R	DisplayString	
currentCPUTemp1	R	DisplayString	
currentCPUTemp2	R	DisplayString	
currentFanSpeed	R	DisplayString	
currentFanDuty	R	DisplayString	
CPUVoltage1	R	DisplayString	
CPUVoltage2	R	DisplayString	
CPUVoltage3	R	DisplayString	
CPUVoltage4	R	DisplayString	
CPUVoltage5	R	DisplayString	
CPUVoltage6	R	DisplayString	
CPUVoltage7	R	DisplayString	
CPUVoltage8	R	DisplayString	
CPUVoltage8	R	DisplayString	

Node name	R/W	Type	Remarks
CPUVoltage9	R	DisplayString	
CPUVoltage10	R	DisplayString	
CPUVoltage11	R	DisplayString	
nvmFreeSpace	R	Counter32	Free space available on the HSM's NVRAM, in bytes
nvmWearLevel	R	DisplayString	Wear level of the HSM's NVRAM
nvmWornBlocks	R	DisplayString	Worn blocks in the HSM's NVRAM

29.7.6.3. Per connection statistics table

The following table gives details of the nodes in the per connection statistics table (enterprises.nCipher.nC-series.statistics.connStatsTable):

Node name	R/W	Type	Remarks
connStatsIndex	R	Integer32	Index of this entry.
connNumber	R	Integer32	Hardserver connection number.
connUptime	R	Counter32	Uptime of the connection.
connCmdCount	R	Counter32	Number of commands submitted through this connection.
connCmdBytes	R	Counter32	Number of bytes submitted through this connection.
connCmdErrors	R	Counter32	Number of marshalling errors on commands through this connection.
connReplyCount	R	Counter32	Number of replies received by this connection.
connReplyBytes	R	Counter32	Number of bytes received by this connection.
connReplyErrors	R	Counter32	Number of marshalling errors on replies through this connection.
connDevOutstanding	R	Gauge32	Number of commands outstanding on this connection.
connQOutstanding	R	Gauge32	Number of commands outstanding in the hardserver queue.

Node name	R/W	Type	Remarks
connLongOutstanding	R	Gauge32	Number of long jobs outstanding for this connection.
connRemoteIPAddress	R	IpAddress	IP Address of connection client.
connProcessID	R	Integer32	Process identifier reported by connection client.
connProcessName	R	DisplayString	Process name reported by connection client.
connObjectTotal	R	Gauge32	The total object count for a connection

29.7.6.4. Module/connection statistics table

The following table gives details of the nodes in the per module, per connection statistics table (enterprises.nCipher.nC-series.statistics.connModuleStatsTable).

Node name	R/W	Type	Remarks
connModuleStatsConnId	R	Integer	Identity of this connection
connModuleStatsModuleIndex	R	Integer	Index of the module entry
connModuleStatsObjectCount	R	Gauge32	The object count on this module for this connection

29.7.6.5. Fan table

The fan table provides the speeds of each fan on the remote module (HSM). The following table gives details of the nodes in the fan table (enterprises.nCipher.softwareVersions.netHSMFanTable):

Node name	R/W	Type	Remarks
netHSMModuleIndex	R	Integer32	Module number
netHSMFanIndex	R	Integer32	Fan number
netHSMFanSpeed	R	Gauge32	Fan speed (RPM)

29.7.6.6. Software versions table

The following table gives details of the nodes in the software versions table (enterprises.nCipher.softwareVersions.softwareVersionsTable):

Node name	R/W	Type	Remarks
compIndex	R	Integer	Table index.
compName	R	DisplayString	Component name.
compOutput	R	Component output name	Component name.
compMajorVersion	R	Gauge	
compMinorVersion	R	Gauge	
compPatchVersion	R	Gauge	
compRepository	R	DisplayString	Repository name.
compBuildNumber	R	Gauge	

29.8. SNMP agent command-line

29.8.1. SNMP agent (snmpd) switches

The SNMP agent that binds to a port and awaits requests from SNMP management software is snmpd. Upon receiving a request, snmpd processes the request, collects the requested information and/or performs the requested operation(s) and returns the information to the sender.

The SNMP agent supports a limited subset of command line switches that can be specified when starting the agent.

Usage

```
snmpd [-h] [-v] [-f] [-a] [-d] [-V] [-P PIDFILE):] [-q] [-D] [-p NUM] [-L] [-l LOGFILE] [-r]
```

This command can take the following options:

Option	Description
-h	This option displays a usage message.
-H	This option displays the configuration file directives that the agent understands.
-v	This option displays version information.
-f	This option specifies not forking from the calling shell.
-a	This option specifies logging addresses.

Option	Description
-A	This option specifies that warnings and messages should be appended to the log file rather than truncating it.
-d	This option specifies the dumping of sent and received UDP SNMP packets.
-V	This option specifies verbose display.
-P	PIDFILE This option specifies the use of a file (PIDFILE) to store the process ID.
-q	This option specifies that information be printed in a more easily parsed format (quick print).
-D	This option turns on debugging output.
-p	NUM This option specifies running on port NUM instead of the default: 161.
-c	CONFFILE This option specifies reading CONFFILE as a configuration file.
-C	This option specifies that the default configuration files not be read.
-L	This option prints warnings and messages to stdout and err.
-s	This option logs warnings/messages to syslog.
-r	This option specifies not exiting if root-only accessible files cannot be opened.
-I	[-]INITLIST This option specifies a list of MIB modules to initialize (or not). Run snmpd with the -Dmib_init option for a list.
-l	LOGFILE This option prints warnings/messages to a file LOGFILE (by default, LOG-FILE=log/snmpd.log).

29.8.2. Using the SNMP command-line utilities

As an alternative to using an SNMP manager application, we supply several command-line utilities to test your SNMP installation and enable you to obtain information about your nShield module from the SNMP agent. These utilities support the -h (display a usage message) as described in the table above.

Utility	Description
snmptest	This utility monitors and manages SNMP information.
snmpget	This utility runs a single GET request to query for SNMP information on a network entity.
snmpset	This utility runs a single SET request to set SNMP information on a network entity.
snmpgetnext	This utility runs a single GET NEXT request to query for SNMP information on a network entity.

Utility	Description
`snmptable`	This utility obtains and prints an SNMP table.
`snmptranslate`	This utility translates SNMP object specifications into human-readable descriptions.
`snmpwalk`	This utility communicates with a network entity using repeated GET NEXT requests.
`snmpbulkwalk`	This utility communicates with a network entity using BULK requests.

These tools are general purpose SNMP utilities and are configurable for use with other SNMP agents. For more information on configuring and using these tools, refer to the NET-SNMP project Web site: `http://net-snmp.sourceforge.net/`.

30. Error codes

If a Hardware Security Module (HSM) encounters an unrecoverable error, it enters the error state. In the error state, the module does not respond to commands and does not write data to the bus.

In some cases you can reset a unit in an error state by powering down the unit and then reapplying power, or with `hsmadmin reset`. Not all errors can be reset in this way.

Errors are a rare occurrence. If any module goes into the error state, except as a result of you issuing the `nopclearfail --fail` command, contact Entrust Support, and give full details of your set up and the error code.

Contact Entrust Support even if you successfully recover from the error by taking the recommended action. For troubleshooting information, see the relevant *Installation Guide* for your module.

30.1. Error codes shown on the LED

The nShield HSM is fitted with a tri-color LED on the back panel. This LED shows information about the status of the HSM, see the *Installation Guide*.

If an error occurs, the LED flashes with a pattern corresponding to one of the error codes listed below. The LED will flash red unless marked as 'Blue LED' in the table below.

30.1.1. Reading LED codes

All the LED error codes are a 3 digit code. The first digit is given by the number of dots, the second digit by the number of dashes, and the third digit by a subsequent number of dots. There is then a longer gap and the error code repeats. The Morse code equivalent is shown in the table below.

The following guidelines are useful when reading LED code messages from the module:

- The duration of a dash (`-`) is 3 times the duration of a dot (`.`).
- The gap between components of a letter has the same duration as a dot.
- The gap between letters has the same duration as a dash.
- The duration of the gap between the code repeating is 7 times the duration of a dot.

Code	Morse code	Dots and dashes	Meaning
1-1-1	E T E	. - .	Battery voltage out of spec

Code	Morse code	Dots and dashes	Meaning
1-2-1	E M E	. - - .	Crypto SerDes core voltage out of spec
1-2-2	E M I	. - - . .	Main processor SerDes core voltage out of spec
1-2-3	E M S	. - - . . .	Main processor core voltage out of spec
1-2-4	E M H	. - -	Main processor SerDes core IO voltage out of spec
1-2-5	E M 5	. - -	Crypto SerDes IO voltage out of spec
1-3-1	E O E	. - - - .	Main processor IFC IO voltage out of spec
1-3-2	E O I	. - - - . .	DDR access voltage out of spec
1-3-3	E O S	. - - - . . .	DDR IO voltage out of spec
1-3-4	E O H	. - - -	V12 voltage out of spec
1-3-5	E O 5	. - - -	Security processor voltage out of spec
1-5-1	E 0 E	. - - - - - .	Security processor temperature out of spec
1-5-2	E 0 I	. - - - - - . .	Main processor temperature out of spec
1-5-3	E 0 S	. - - - - - . . .	Crypto temperature out of spec
1-5-4	E 0 H	. - - - - -	Security processor app blank
1-5-5	E 0 5	. - - - - -	Security processor app invalid
2-1-1	I T E	. . - .	Security processor secure state corrupted
2-1-2	I T I	. . - . .	No bootloader heartbeat
2-1-3	I T S	. . - . . .	Board-ID PROM failed
2-1-5	I T 5	. . -	Firmware signature auth failure (Blue LED)
2-2-2	I M I	. . - - . .	Crypto known-answer tests failed
2-2-3	I M S	. . – . . .	RNG driver failed
2-2-4	I M H	. . - -	FIPS DRBG failed
2-2-5	I M 5	. . - -	OpenSSL failed
2-3-1	I O E	. . - - - .	OpenSSH failed
2-3-2	I O I	. . - - - . .	Library signature verification failed
2-3-3	I O S	. . - - - . . .	FPGA initialisation failed
2-3-4	I O H	. . - - -	Init script failed

30.2. Error codes available remotely

The error codes listed in this chapter are reported by the `enquiry` utility in the `hardware status field` of the `Module` and are included in the hardserver log.

30.2.1. Runtime library errors

The runtime library error codes could be caused by firmware bugs or by faulty hardware. If any of these errors occur, reset the module.

Code	Meaning
O L C	SIGABRT: assertion failure and/or `abort()` called.
O L D	Interrupt occurred when disabled.
O L E	SIGSEGV: access violation.
O L J	SIGFPE: unsupported arithmetic exception (such as division by 0).
O L K	SIGOSERROR: runtime library internal error.
O L L	SIGUNKNOWN: invalid signal raised.

Codes `OLD` and `OLE` are more likely to indicate a hardware problem than a firmware problem.

30.2.2. Hardware driver errors

In general, the hardware driver error codes described in the following table indicate that some form of automatic hardware detection has failed. As well as indicating simple hardware failure, one of these error codes could indicate that there is a bug in the firmware or that the wrong firmware has been loaded.

If any of these errors is indicated, contact Entrust support.

Code	Meaning
H L	M48T37 NVRAM (or battery) failed
H C V	CPLD wrong version for PCI policing firmware.
H C X	No crypto offload hardware detected.
H P P	PCI Interface Policing failure.
H V	Environment sensors failed (for example, temperature sensor)
H D	Failure reading unique serial number.
H R	Random number generator failed.

Code	Meaning
H R F O	FIPS continuous RNG failed.
H R A O	Periodic RNG test failed.
H R S	RNG startup failed.
H R T	RNG selftest failed.
H R T P	Periodic (scheduled daily) RNG selftest failed.
H R M	RNG data matched.
H R Z	Impossible RNG Failure (match after PRNG)
H S S	Security processor internal semaphore error
H O	Token interface initialization failed.
H E	EEPROM failed on initialization.
H C	Processing thread initialization failed.
H C P	Card poll thread initialization failed.
H F	Starting up crypto offload.
H C V	CPLD version number incorrect.
H J V	IPC-watcher failed
H J U	IPC-EPD failed
H J R	Module reset notification failed.
K R	RSA selftest failed.
H H D	Unique serial number detection failed.
H H P	PCI bus hardware detection failed.
H H R	RTC hardware detection failed or random number generator detection failed.
H S C	Error writing correct SOS message.

30.2.3. Operational mode errors

The following runtime library error codes could be caused either by bugs in the firmware or by faulty hardware.

Code	Meaning	Action
T	Temperature of the module has exceeded the maximum allowable.	Restart your host computer, and improve module cooling.

Code	Meaning	Action
D	Fail command received.	Reset module by turning it off and then on again.
G G G	Failure when performing `ClearUnit` or `Fail` command.	Contact Entrust Support.
I J A	Audit logging: failed to send audit log message.	Contact Entrust Support.
I J B	Audit logging: no module memory (therefore failed to send audit log message).	Contact Entrust Support.
I J C	Audit logging: key problem or FIPS incompatibility (therefore failed to sign audit log message).	Contact Entrust Support.
I J D	Audit logging: NVRAM problem (therefore failed to configure or send audit log message).	Contact Entrust Support.

SOS IJA can occur for any type of log message (i.e. a log message, signature block or certifier block).

For the cooling requirements for your module, see the *Installation Guide*.

31. Uninstalling Security World Software

This appendix describes how to uninstall Security World Software.

Do not uninstall the Security World Software unless either you are certain it is no longer required or you are going to upgrade it.

The uninstaller removes only those files that were created during the installation. To remove key data or Security World data, navigate to the installation directory and delete the files in the *%NFAST_KMDATA%* folder.

If you intend to remove your Security World before uninstalling the Security World Software, Entrust recommends that you erase the OCS before you erase the Security World or uninstall the Security World Software. Except where Remote Administration cards are used, after you have erased a Security World, you can no longer erase any cards that belonged to it.

1. Log in to the host computer as Administrator or as a user with local administrator rights.
2. Run the following command to erase the OCS:

   ```
   createocs -m# -s0 --erase
   ```

   where # is the module number.
3. Navigate to the Windows Control Panel, and double-click **Programs and Features**.
4. Select the Security World Software entry, then click **Uninstall** to remove the software.

If required, you can safely remove the nShield module after shutting down all connected hardware.

32. Application Performance Tuning

32.1. 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 increas ing the thread count above the `rec. queue` range or the number that gives the best through put 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 tar- get 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.

32.2. 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 syn- chronously in each. You can use the `--threads` parameter to the `ncperftest` utility to experi ment 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 out- standing in that connection.

When using higher-level APIs such as PKCS#11, all cryptographic operations are synchro- nous, 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.

32.3. 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`.

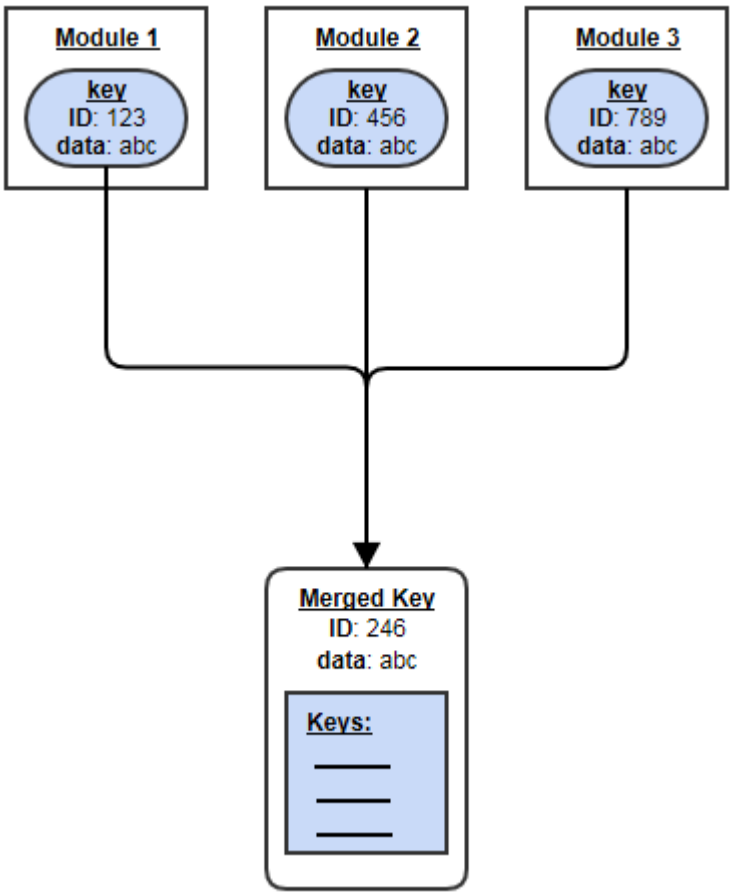
On some systems the default operating system scheduling algorithm is also not optimized for highly multi-threaded applications. A real-time scheduling algorithm such as the POSIX round-robin scheduler may yield noticeable performance improvements for your application.

32.4. File Descriptor Limits

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.

33. Merged Keys Concept

A merged key is a level of abstraction higher than normal keys. It holds an internal list of nor mal key IDs, each associated with its corresponding module. When a command to the hard-server specifies a MergedKey ID instead of a normal (single) key ID, the hardserver chooses an HSM and corresponding single key ID from the list in the Merged Key. See diagram below. Which module is chosen may depend on multiple factors, including load sharing set-tings in the hardserver config.



Benefits of MergedKeys:

- A client need hold only a single M_KeyID reference instead of one for each HSM.
- That ID remains usable even while the key's actual IDs on HSMs can fluctuate.
- The hardserver can use heuristics to choose the most appropriate HSM (e.g. the least heavily loaded one).
- If some HSMs become unavailable, the hardserver uses the remaining ones automati-cally.
    - A MergedKey can be updated, removing its entry for a defunct HSM and corre-

sponding single-key ID.

- New HSMs can be added: if a new HSM is made operational and added to the relevant security world, then
  - the key can be loaded onto that HSM, thus creating a new single-key ID for that key on that HSM, and then
  - the new (Key ID, HSM) pair can be added to the existing Merged Key.

34. Product returns

If you need to return your nShield HSM, contact Entrust nShield Support for instructions: https://nshieldsupport.entrust.com

Before returning your nShield HSM you should return it to factory state by following the pro cedure described in Returning a module to factory state.

35. Returning a module to factory state

35.1. Factory state

nShield HSMs that are delivered from the factory contain no data relating to the `ncoreapi` service. A small amount of 'lifetime' data, which is used by the platform services, is pre-installed. This data is for personalisation and identification of the individual HSM, such as its ESN.

You can perform a reset operation that returns the data stored in an HSM to the state it was in when it left the factory. This erases user credentials and information, leaving only the 'lifetime' data.

When an HSM is in this state it will not support any user commands other than `hsmadmin enroll` and it will be necessary to follow the process described in Set up communication between host and module before any further actions can be taken.

35.2. Purpose of factory state

The main reason for returning an HSM to factory state is to securely erase all user secrets if, for instance, the HSM is being taken out of service or being moved from one domain to another where it is important to ensure that there is no possibility of secrets being leaked between domains.

You should also return your unit to factory state when returning the unit to Entrust for servicing or warranty.

Returning a unit to factory state will also be necessary if you have lost possession of the SSH keys used to communicate with the HSM and you have not previously made a backup of those keys with `hsmadmin keys backup` (or `hsmadmin keys backup --passphrase` if the HSM is being re-installed in a different machine). If this happens returning the HSM to factory state will allow `hsmadmin enroll` to successfully create new keys and re-establish communication with the HSM.

35.3. Entering and exiting factory state

The HSM can be returned to factory state in one of two ways. Either by use of `hsmadmin factorystate` or by placing the HSM in recovery mode as described in Recovery mode.

If the SSH keys used to communicate with the HSM have been lost, only the recovery

mode option is possible. Both of the above methods include a reboot of the HSM.

The HSM is taken out of factory state by use of `hsmadmin enroll`

36. Remote File System Volumes

The hardserver service restricts the paths that can be shared as RFS (Remote File System) volumes using the `[remote_file_system]` section of the `config` file or using the `rserverperm --accessfiles` command-line configuration.

By default, the following paths are permitted:

- `/opt/nfast/kmdata`.
- Any path that was created by the `rfs-setup` utility and associated with RFS volumes to prepare an RFS for an nShield HSM or for use with the `rfs-sync` utility.
- Subdirectories of permitted paths.

If you want to add custom paths not included in this list as RFS volumes, you must add them to the list of permitted paths before starting the hardserver service. If you make these changes after starting the service, you need to restart it for the changes to take effect.

You can update the list of permitted paths by either setting the NFSERV_RFS_ALLOWED_-PATHS environment variable (see Allow custom RFS paths with an environment variable) or by creating an additional `config.secure` configuration file (see Allow custom RFS paths with a configuration file.)

36.1. Allow custom RFS paths with an environment variable

1. If the `/etc/nfast.conf` file does not already exist, create it.

   This file must only be writable by root. This is enforced by nShield start-up scripts.

2. Add the `NFSERV_RFS_ALLOWED_PATHS` environment variable to the `nfast.conf` file with a colon-separated list of paths (`/<path>/share`).

   For example, to share `path1` and `path 2` (spaces are permitted):

   ```
   export NFSERV_RFS_ALLOWED_PATHS=/path1/share:/path 2/share
   ```

36.2. Allow custom RFS paths with a configuration file

1. Create the `config.secure` in the `/opt/nfast/hardserver.d` directory.
2. Add the paths as values in an `rfs_allowed_paths` JSON array. The JSON must be valid.

   For example, to share `path1` and `path 2` (spaces are permitted):

```
{
  "rfs_allowed_paths" : ["/path1/share", "/path 2/share"]
}
```

37. SSH Client Key Protection

37.1. SSH Services

This table explains what the different services are used for, to help inform what protection settings are appropriate for the client keys for those services in a deployment.

Service	Service Description
`sshadmin`	Main authority for administration of the SSH services on the HSM. This key should have the strongest protection.
`ncoreapi`	nCore API service. Used by the hardserver for routine communication with the HSM.
`setup`	Setup service. Used for some administration options such as factory state.
`updater`	Updater service. Used for installing signed firmware upgrade packages.
`monitor`	Monitor service. Used for diagnostic operations such as retrieving logs with `hsmadmin logs`.

37.2. SSH Client Key Encryption

SSH client keys are protected by a passphrase derived from one or more inputs including machine IDs and user-supplied passphrases.

These passphrases are derived automatically by applications which use the SSH keys, and with the exception of the option of user-supplied passphrases do not prompt the user.

37.2.1. Available SSH Key Protection Options

The following are the supported protection options for SSH keys. Multiple options can be combined in any order.

Although the hardserver runs as `nfast` user, it starts as root and then drops privileges. The hardserver derives any SSH key passphrases before dropping privileges, and so can use pro tections that are available only to root.

Option	Description
`K`	Per-key nonce. Ensures that the derived passphrase is unique to the key.
`F`	Fixed nonce present in the nShield client software.

Option	Description
S	System UUID. Ties the key to the local machine. On Linux, this is only readable by root. This is available on most systems.
B	Baseboard UUID. Ties the key to the baseboard (motherboard) of the local machine. On Linux, this is only readable by root. This may not be available on all systems.
G	Global nonce. Ties the key to the local OS install. The global nonce is readable only by root or Administrators and is generated the first time the option is used to protect a key.
P	User passphrase. The key will require a user-supplied passphrase (in addition to any other protection options specified).

37.2.1.1. User-supplied SSH Key passphrases

If a key is generated with protection by a user-supplied SSH key passphrase, there will be an interactive prompt on the console to enter and confirm the passphrase. When a key is loaded with passphrase protection, there will be an interactive prompt on the console to enter the passphrase.

To avoid interactive prompts for automation purposes, the user passphrase can be supplied using the environment variable `NFAST_KEYPROT_PASS`. If a user passphrase is specified for the `ncoreapi` service SSH key, then the environment variable is the only way to supply the passphrase as interactive prompts are disabled for the hardserver service.

37.3. Setting Protections on SSH keys

Protections are set on SSH keys when they are generated, either during `hsmadmin enroll` (if the keys are not already present) or during `hsmadmin keys roll` (if switching to a freshly generated set of SSH client keys).

Protections can be overridden using environment variables that are set in the environment of the above commands when the keys are generated. The protections for all SSH service keys can be overridden using the `NFAST_KEYPROT` environment variable. Individual SSH service keys can have their protections set directly using per-service environment variables as specified in the table below. If both `NFAST_KEYPROT` and the per-service environment variable are set, the per-service environment variable takes precedence.

Service	Default Protection	Environment Variable
sshadmin	KFSG	NFAST_SSHADMIN_KEYPROT

Service	Default Protection	Environment Variable
ncoreapi	KFSG	NFAST_NCOREAPI_KEYPROT
setup	KFSG	NFAST_SETUP_KEYPROT
updater	KF	NFAST_UPDATER_KEYPROT
monitor	KF	NFAST_MONITOR_KEYPROT

37.4. Permissions on SSH keys

Access to SSH keys is controlled by permissions on the directories they are created in. The key directories are created with permissions during installation of the nShield software.

All keys are owned by user `root`, except for `ncoreapi` which is owned by user `nfast`. The table below shows what group can read each of the service keys by default. The group that can read each key can also be overridden with the environment variables listed below if set in the environment when running the install script `/opt/nfast/sbin/install`. The install script will always set the owner and group on the key, so if custom groups are used, they must be specified every time the install script is run. If the group specified in the environment variable does not exist, it will be created automatically by the install script.

Service	Default Group	Environment Variable
sshadmin	root	NFAST_SSHADMIN_GROUP
ncoreapi	root	NFAST_NCOREAPI_GROUP
setup	root	NFAST_SETUP_GROUP
updater	nfastadmin	NFAST_UPDATER_GROUP
monitor	nfastadmin	NFAST_MONITOR_GROUP