nShield Security World

nShield Solo, Solo XC and Edge v12.40 User Guide for Windows

04 March 2024

Contents

Chapter 1: Introduction

Read this guide if ...

Read this guide if you need to configure or manage:

- An nShield® Solo or nShield Edge (Linux platforms only) Hardware Security Module (HSM) .
- An associated *Security World*. nCipher hardware security modules use the Security World paradigm to provide a secure environment for all your HSM and key management operations.

All nCipher HSMs support standard cryptography frameworks and integrate with many standards 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 Solo or nShield Edge.

Note: Throughout this guide, the term Installation Guide refers to the particular Installation Guide for your product.

Terminology

When nCipher hardware security products are referred to in general, the term *hardware security module* or *HSM* is used. When specific products are referred to they are referred to by name.

Conventions

Typographical conventions

Note: The word **Note** indicates important supplemental information.

If there is a danger of loss or exposure of key material (or any other security risk), this is indicated by a security triangle in the margin.

If there is a danger of damage to the hardware, this is indicated by a caution triangle in the margin. If you see this symbol on the product itself, see the *nShield Solo Installation Guide*.

Si une détérioration du matériel est possible, un triangle d'avertissement l'indique dans la marge. Si ce symbole apparaît sur le produit lui-même, reportez-vous à la partie correspondante du *nShield Solo Installation Guide*.

Besteht die Gefahr eines Hardware-Schadens, wird dies am Rand durch ein Warndreieck angezeigt. Falls Sie dieses Symbol auf dem Produkt selbst bemerken, schlagen Sie im zutreffenden Abschnitt des Installationshandbuchs (*nShield Solo Installation Guide*) nach.

Keyboard keys that you must press are represented like this: `Enter`, `Ctrl-C`.

Examples of onscreen text from graphical user interfaces are represented by **boldface** text. Names of files, command-line utilities, and other system items are represented in **monospace** text. Variable text that you either see onscreen or that you must enter is represented in *italic*.

Examples of onscreen terminal display, both of data returned and of your input, are represented in a form similar to the following:

```
install
```

CLI command conventions

The basic syntax for a `CLI` command is:

```
command object <object_name> [parameter] [option] [modifier]
```

In this syntax, user-defined values are shown in *italics* and enclosed within the **< >** characters. Optional elements are shown enclosed within the **[ ]** characters. Mutually exclusive elements are separated by the **|** character.

Many system objects require the inclusion of a user-defined keyword value. For example, the **user** object is executed against a user-supplied *user_name*. Throughout this guide, all user-defined keyword values are shown in *italics*.

Each `CLI` command that you run performs an operation against the internal configuration of the appliance. The specific type of operation is specified by the first user-defined keyword value in the command string.

Model numbers

Model numbering conventions are used to distinguish different nCipher hardware security devices. In the table below, *n* represents any single-digit integer.

Model number	Used for
nC3*nnn*E-*nnn*, nC4*nnn*E-*nnn*	nShield Solo HSM with a PCI Express (PCIe) interface
nC30n5E-*nnn*, nC40n5E-*nnn*	nShield Solo XC HSM with a PCIe interface.
nC30*nn*U-10, nC40*nn*U-10.	An nShield Edge module

Security World Software

The hardserver software controls communication between applications and nCipher nShield product line HSMs, which may be installed locally or remotely. It runs as a service on the host computer.

The Security World for nShield is a collection of programs and utilities, including the hardserver, supplied by nCipher to install and maintain your nCipher security system. For more information about the supplied utilities, see *Supplied utilities* on page 197.

nCipher provides the firmware that runs on the nShield Solo or nShield Edge, and software to run on each client computer. For more information about:

- Upgrading the firmware, see *Upgrading firmware* on page 277
- Installing and configuring the software on each client computer, see the Installation Guide and *Software and module configuration* on page 37.

Software architecture

nCipher supply the software to control communication between the unit and applications on the network.

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	Environment variable	Windows Server 2008 or later
nShield Installation	`NFAST_HOME`	32 bit: `C:\Program Files\nCipher\nfast` 64 bit: `C:\Program Files (x86)\nCipher\nfast`
Key Management Data	`NFAST_KMDATA`	`C:\ProgramData\nCipher\Key Management Data`
Dynamic Feature Certificates	`NFAST_CERTDIR`	`C:\ProgramData\nCipher\Feature Certificates`
Static Feature Certificates		`C:\ProgramData\nCipher\Key Management Data`
Log Files	`NFAST_LOGDIR`	`C:\ProgramData\nCipher\Log Files`
Remote Static Feature Certificates		`%NFAST_KMDATA%\hsm-ESN\features`
Remote Dynamic Feature Certificates		`%NFAST_KMDATA%\hsm-ESN\features`

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

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

The instructions in this guide refer to the locations of the software installation and program data directories by their names (for example, Key Management Data) or nShield environment variable names enclosed with percent signs (for example, *%NFAST_KMDATA%*).

If the software has been installed into a non-default location, ensure that the associated nShield environment variables are re-set with the correct paths for your installation.

Note: With previous versions of Security World Software, the Key Management Data directory was located by default in **`C:\nfast\kmdata`**, the Feature Certificates directory was located by default in **`C:\nfast\fem`**, and the Log Files directory was located by default in **`C:\nfast\log`**.

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.

Document version numbers

The version number of this document is shown on the copyright page of this guide. Quote the version number and the date on the copyright page if you need to contact Support about this document.

Further information

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

If you have installed the CodeSafe or CipherTools developer products, 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.

Release notes containing the latest information about your product are available in the release directory of your installation media.

Note: We strongly recommend familiarizing yourself with the information provided in the release notes before using any hardware and software related to your product.

Security advisories

If nCipher Security becomes aware of a security issue affecting nShield HSMs, nCipher Security will publish a security advisory to customers. The security advisory will describe the issue and provide recommended actions. In some circumstances the advisory may recommend you upgrade the nShield firmware and or nShield Connect 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.

Note: The Remote Administration feature supports remote firmware upgrade of nShield Solo and nShield Connects and remote ACS card presentation.

We recommend that you monitor the "nShield Announcements & Security Notices" section of the nCipher Security Support Portal, where any announcement of nShield Security Advisories will be made.

Contacting nCipher Support

To obtain support for your product, visit: https://help.ncipher.com.

Recycling and disposal information

In compliance with the WEEE (Waste Electrical and Electronic Equipment) directive for the recycling of electronic equipment, nCipher provides a Takeback and Recycle program.

The program enables you to ship an obsolete or excess nShield product line hardware security device to nCipher, who then dispose of the product in an environmentally safe manner. For further information or to arrange the safe disposal of your hardware security device, contact nCipher Support.

Chapter 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

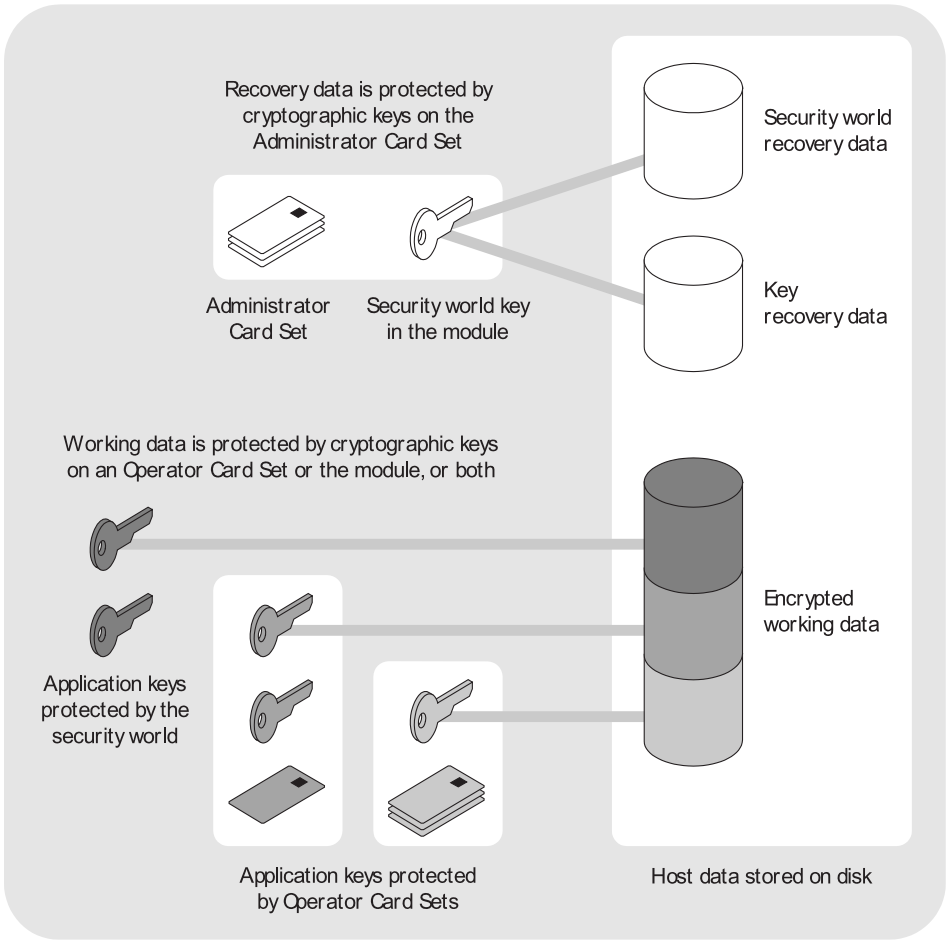
A Security World comprises:

- One or more nCipher nShield HSMs
- An *Administrator Card Set* (ACS)
  A set of Administrator smart cards used to control access to the Security World configuration, 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 Figure 1.

Distributing the keys used for different tasks within the Security World over different storage 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 components before gaining any information.

Figure 1. Key protection in a Security World



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 nCipher 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.

Smart cards

The Security World uses:

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

Note: In FIPS 140-2 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*.

Note: 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.

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.

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 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* on page 27 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.

Remote Administration

The Remote Administration feature enables:

- Card holders to present smart cards to an HSM that is in a different location
  For example, the card holder may be in an office, while the HSM is in a data center.
- All smart card operations to be carried out (apart from loading feature certificates)
  Unlike the Remote Operator feature, Remote Administration supports the ACS and non-persistent OCS cards.
- Security World programs and utilities to be run and authorized remotely, without accessing an HSM card slot, when used in combination with a standard remote access solution

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.

Security World programs and utilities

As has always been the case, it is possible to run the Security World programs and utilities remotely using your preferred remote access solution, e.g. SSH or Remote Desktop. This means you can now 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 also adds the ability to change the mode of HSMs remotely using the nopclearfail utility. This means it is now possible to create a Security World remotely and perform future firmware upgrades.

HSM configuration

To support Remote Administration, HSMs have to be configured to support between zero (default) and 16 Dynamic Slots. These 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 card slot that may be available. See *Configuring Dynamic Slots* on page 44 for more information.

Note: To use Remote Administration, you need to:

- Upgrade your HSM firmware to version 2.61.2 or later
- Upgrade the HSM warrant to a KLF2 warrant

The firmware is available on the installation media. See *Appendix M: Upgrading firmware* on page 277.

Note: Dynamic Slots cannot be imported or exported as Remote Operator slots.

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 the nShield Remote Administration Client *User Guide* 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 *nShield Solo Installation Guide* for more about where to install the Remote Administration Service software.

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.

nShield Remote Administration 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 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* on page 46 for more information.

Card readers

nCipher supplies and recommends the use of the nShield Trusted Verification Device. This card reader allows the card holder to securely confirm the Electronic Serial Number (ESN) of the HSM to which they want to connect, using the nShield Trusted Verification Device display. As an alternative, and if it permitted by your security policy, card holders may use a standard smart card reader for ISO/IEC 7816 compliant smart cards.

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* on page 37.

NIST SP800-131A

You can create a Security World compliant with NIST (National Institute of Standards and Technology) SP800-131A (Special Publication 800-131A Revision 1, November 2015); see *Cipher suite* on page 59. For details about NIST SP800-131A, see http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-131Ar1.pdf.

There is a migration tool available for transferring existing Security World data into an SP800-131A Security World; see *Migrating keys to an SP800-131A Security World* on page 84.

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

FIPS 140-2 compliance

All Security Worlds are compliant with the Federal Information Processing Standards (FIPS) 140-2 specification. The default setting for Security Worlds complies with level 2 of FIPS 140-2.

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

FIPS 140-2 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 140-2 at level 2
- FIPS 140-2 at level 3

The FIPS 140-2 level 3 option is included for those customers who have a regulatory requirement for compliance with FIPS 140-2 at level 3.

If you choose to create a Security World that complies with FIPS 140-2 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 140-2 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 140-2, see http://csrc.nist.gov/publications/fips/fips140-2/fips1402.pdf.

Common Criteria compliance

nShield Solo, Solo+, Connect and Connect+ are certified to Common Criteria v3.1 to EAL4+ AVA_VAN.5. To configure and operate the module in its evaluated configuration, the separate Common Criteria guides should be followed. Please contact nCipher Support for these guides.

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 Unix-based platforms.
- Include hosts running different operating systems in the same Security World.

Note: 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).

Application independence

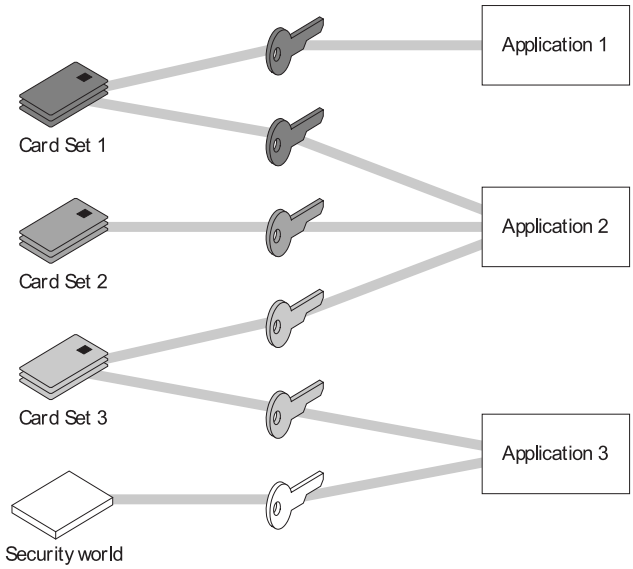
A Security World can protect keys for any applications correctly integrated with the Security World Software. Each key belongs to a specific application and is only ever used by that application. Keys are stored along with any additional data that is required by the application.

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 (see Figure 2).

Figure 2. Operator Card Sets, keys, and applications



In Figure 2:

- **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**.

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 application keys and the OCSs in the Security World. This *Security World key* can be an AES (Advanced Encryption Standard) key which is the default and recommended option, or a Triple DES (Data Encryption Standard) key which is no longer recommended. The options are as follows:

Security World key / Cipher suite	Uses
Triple DES - not recommended	1024-bit DSA key with SHA-1 hash
AES 256 - original	1024-bit DSA key with SHA-1 hash
AES 256 (SP800-131A compliant) - default	3072-bit DSA 2 key with SHA-256 hash

Note: To create a Triple DES Security World, you must use the `new-world` command-line utility.

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

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 pass phrase
- 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.

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.

Note: Some applications restrict *K* to 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.

Note: 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.

Note: 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* on page 61.

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* on page 32).

Note: 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* on page 61.

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* on page 28 for more information
- If there is a power loss to the module, for example, due to power outage.

Note: 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 re-insert the card, and enter its pass phrase 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.

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`**
- Press the Clear button of the hardware security module

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.

Using pass phrases for extra security

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

You can change the pass phrase for a card at any time provided that you have access to the card, the existing pass phrase, and a hardware security module that belongs to the Security World to which the card belongs. For more information, see *Changing card and softcard pass phrase* on page 108.

Note: Some applications do not support the use of pass phrases.

Maximum pass phrase length

Note: The maximum pass phrase length limitation is not applicable to software versions before Security World Software v11.72.

Pass phrases 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 pass phrases that are longer than 254 characters.

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

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

Using softcard-protected keys

If you want to use pass phrases 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 pass phrase. 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 %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 pass phrase 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 softcard is loaded, it remains valid for loading its keys until its **KeyID** is destroyed.

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, the Microsoft CSP wizard 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 nShield Connects. See the *nShield Connect User Guide* for more about the RFS.
- Share Security World and key data stored in the %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* on page 37.

Note: We provide the **`rfs-sync`** command-line utility to synchronize the *%NFAST_KMDATA%* 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).

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.

Note: 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 restarting the application (i.e. failback support). The pool of HSMs can be viewed as a single resource using the command **`enquiry --pool`**.

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

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.

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.

Note: In FIPS 140-2 Level 3 Security Worlds, the ACS or an OCS is needed to control many operations, including the creation of keys and OCSs.

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* on page 77.

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* on page 121).

Note: 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.

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 pass phrases.

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.

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.

Note: The ability to replace an OCS is an option that is enabled by default during Security World creation (see *OCS and softcard replacement* on page 61). 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.

Note: 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.

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.

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* on page 186.

Note: 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 a Security World and its ACS, as either FIPS 140-2 level 2, or level 3
  Note: 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 regaulatory requirement for compliance.
- Add a hardware device to a Security World
- Remove a hardware security module from a Security World
- Replace an ACS
- Create OCSs
- List the OCSs in the current Security World
- Change the pass phrase 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* on page 197. Current versions of these tools can be used interchangeably with the current version of KeySafe.

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://www.ncipher.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 nCipher hardware security modules and Security Worlds.

For more information about the nCipher range of Integration Guides:

- Visit https://www.ncipher.com
- Contact Support.

The nCipher 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 nCipher 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 configuring the application to use the nCipher PKCS #11 library.

The nCipher 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 nCipher PKCS #11 library has vendor specific extensions that support *K/N* card sets, see *nCipher PKCS #11 library with the preload utility* on page 138.

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

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 authorized 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 nCipher or directly from the smart card manufacturer.

  Note: nShield Remote Administration Cards can only be supplied by nCipher.
- Never insert a smart card used with nCipher 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 pass phrase.
- Never write down your pass phrase.
- Never use a pass phrase that is easy to guess.

Note: 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.

Chapter 3: 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.

After software installation and testing

After you have successfully installed and tested the Security World Software, as described in the Installation 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 necessary, perform additional software and HSM configuration tasks, as described in *Software and module configuration* on page 37 :
   - Set up client configuration, as described in *Setting up client cooperation* on page 37.
   - Set nShield specific environment variables, as described in *Setting environment variables* on page 40.
   - Configure logging and debugging parameters, as described in *Logging and debugging* on page 41.
   - Configure Java support for KeySafe, as described in *Configuring Java support for KeySafe* on page 41
   - Configure the hardserver, as described in *Configuring the hardserver* on page 41.
3. Create and configure a Security World, as described in *Creating a Security World* on page 57.
4. Create an OCS, as described in *Creating Operator Card Sets (OCSs)* on page 93.

Chapter 4: 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* on page 57) and an OCS (as described in *Creating Operator Card Sets (OCSs)* on page 93).

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.

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 Connect HSMs:

1. Configure the RFS used by your nShield Connect to accept access by cooperating clients.
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 that use a local K_{NETI} for authorization (which is generated when the HSM is first initialized from factory state) and which are to be given write access to the RFS, run the command:

     ```
     rfs-sync --setup rfs_IP_address
     ```

   - for clients that do not have a local K_{NETI} and require write access, run the command:

     ```
     rfs-sync --setup --no-authenticate rfs_IP_address
     ```

     Note: 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.
     Note: 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.
     Note: For more information about the **rfs-sync** utility, see *rfs-sync* on page 39.
3. 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=%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=%NFAST_KMDATA%\localsync-store
volume=kmdata-backup
allow_read=yes
allow_write=yes
allow_list=yes
is_directory=yes
is_text=no
```

Useful utilities

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.

rfs-sync

This utility synchronises the **kmdata** 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.

Usage

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

Options

-U, **--update**

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

Note: 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 they are until manually removed.

-c, **--commit**

These options commit local key-management data changes to 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

These set-up options specify use of KNETI authentication. This is the default.

--no-authenticate

This set-up option specifies that KNETI authentication should not be used.

-m, --module=module

These options select which HSM to use for KNETI authorisation. The default is HSM 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.

ip_address

This option specifies the IP address of the remote file system.

--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.

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

Setting environment 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* on page 215.

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

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

8. Open the **Services** console by clicking the **Services** icon.
9. From the displayed list of services, select the **nFast Server** icon, and select **Restart the service**.

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`

Note: 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* on page 219.

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.

Configuring Java support for KeySafe

To use KeySafe, follow the instructions in *Using KeySafe* on page 186.

Configuring the hardserver

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

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

The hardserver can handle transactions for multiple HSMs. This does not require configuration of the hardserver. For more information, see *Using multiple modules* on page 54.

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

The hardserver configuration file defines the configuration of the hardserver. By default, it is stored in the *%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* on page 42 for an overview of the hardserver configuration file, and see *Hardserver configuration files* on page 247 for detailed information about the various options available through it.

Note: 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 *%NFAST_KMDATA%***config** so that the configuration can be restored if necessary.
2. Edit the configuration file *%NFAST_KMDATA%***config****config** to contain the required configuration. (See *Hardserver configuration files* on page 247 for descriptions of the options in the configuration file.)
3. Run the **cfg-reread** command-line utility to load the new configuration.
   Note: If you changed the server_startup section of the hardserver configuration file, you must restart the hardserver instead of running **cfg-reread**. For more information, see *Stopping and restarting the hardserver* on page 55.
4. Test that the hardserver is configured correctly by running the **enquiry** command-line utility. Check that an HSM with the correct characteristics appears in the output.
5. Test that the client has access to the Security World data by running the **nfkminfo** command-line utility.
   Check that an HSM with the correct ESN appears in the output and has the state **0x2 Usable**.

Overview of hardserver configuration file sections

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 nCipher 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_remotecomms** section of the hardserver configuration file. This section defines the port on which the hardserver listens for communications from remote HSMs. You need to edit this section only if the default port (9004) is not available.

For detailed descriptions of the options in this section, see *server_remotecomms* on page 251.

For information about configuring the Remote Operator feature (Remote Operator slots), as opposed to remote HSMs, see *Remote Operator* on page 171.

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* on page 248.

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* on page 251.

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* on page 251.

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.

Note: You should use the **nt_privpipe_users** option to define the name of the user who is allowed to carry out privileged operations, for example, using the **nopclearfail** utility. See *nt_privpipe_users* on page 252 for more information.

For detailed descriptions of the options in this section, see *server_startup* on page 252.

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 on the module* on page 46.

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* on page 252

For information about SEE machines, see *CodeSafe applications* on page 150.

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* on page 253 and *slot_exports* on page 254.

The Remote Operator feature must be enabled on the HSM, as described in *Enabling optional features on the module* on page 46.

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* on page 37.

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* on page 256.

Configuring Remote Administration

Hardserver configuration

You can configure Remote Administration by editing sections of the hardserver configuration file.

Configuring Dynamic Slots

To use Remote Administration, you need to configure the number of Dynamic Slots that are available on an HSM. The maximum number of Dynamic Slots that an HSM can have is 16.

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

Do the following:

1. Use the **dynamic_slots** section in the hardserver configuration file to define the number of Dynamic Slots for each relevant HSM.
   See *dynamic_slots* on page 254 for more about the **dynamic_slots** section.
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.

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

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.

Do the following:

- Use The **slot_mapping** section in the hardserver configuration file to define a Dynamic Slot to exchange with slot 0 for each relevant HSM.
  See *slot_mapping* on page 254 for more about the **slot_mapping** section.

You can check the mapping by 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.

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 hardserver configuration file to define the round trip (HSM to smartcard and back) time limit, and the card removal detection timeout.
  See *dynamic_slot_timeouts* on page 255 for more information.

Authorized Card List

The use of nShield Remote Administration 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 Dynamic Slot of the HSM, through the nShield Remote Administration Client

By default, the Authorized Card List is empty following software installation. The serial numbers of Remote Administration Cards must be added to the list before they can be used.

The Authorized Card List is a text file *%NFAST_KMDATA%***config****cardlist**. The list applies to all HSMs that are inside or associated with the host computer where the file is stored, regardless of the Security World to which an HSM may belong. This remains true for an HSM that is being used to create a Security World.

Write access to the Authorized Card List is controlled by the permissions set for your operating system.

Adding cards to the Authorized Card List

Add the serial numbers (16 digits optionally with separators) of all nShield Remote Administration 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.

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

Enabling optional features on the module

nShield modules support a range of optional features. Optional features must be enabled with a certificate that you order from nCipher. You can order the features when you purchase a module, or you can obtain the certificate at a later date and use the Feature Enable Tool to enable the features.

For more information about:

- Ordering optional features, see *Ordering additional features* on page 51
- Feature-enabling procedures, see *Enabling features* on page 52.

The firmware 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.

Most features are *static*; that is, they are enabled by means of a switch in the **EEPROM** of the module. A static feature remains enabled when the module is reinitialized.

Some optional features are *dynamic*; that is, they are enabled by means of a software switch in the volatile memory of the module. A dynamic feature must be enabled again if the module is reinitialized.

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** or by pressing the module's **Clear** switch.

Note: If you are enabling the Remote Operator feature, you must enable it on the module that is to be used as the unattended module. For information about Remote Operator, see *Remote Operator* on page 171.

Available optional features

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.

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

Elliptic Curve activation

All nShield HSMs require specific activation to utilize the elliptic curve features. nCipher uses an activator smart card to enable this feature. Refer to *Enabling features* on page 52 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* on page 52. Contact Sales if you require an EC activation.

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

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	Custom curves[1, 5]

HSM module type	Elliptic Curve support		Elliptic Curve offload acceleration[3]	
nShield Edge[2].	Yes.	Yes.	No.	No.
nShield PCIe 500 and 6000; nShield Connect 500, 1500 and 6000[2].	Yes.	Yes.	No.	No.
nShield PCIe 500+ and 6000+, nShield Connect 6000+[2].	Yes.	Yes.	Yes, Prime curves are accelerated[2, 4].	Yes.

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

[2]Both Prime and Binary named curves are supported. Refer to *Named Curves* on page 48, 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]Includes support for Brainpool curves. See *Brainpool curves* on page 49, below, for a list of Brainpool curves.

nShield software / API support required to use elliptic curve functions

	Security World Software for nShield	CipherTools	CodeSafe
Elliptic curve supported / API	Microsoft CNG, PKCS#11, Java Cryptographic Engine (JCE)[1].	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 nCipher security provider, nCipherKM. There is also the option to use the Sun/IBM PKCS #11 Provider with nCipherKM configured to use the nCipher PKCS#11 library.

To demonstrate the accelerated performance of elliptic signing and verify operations, run the `perfcheck` utility. See *perfcheck: performance measurement checking tool* on page 237.

Named Curves

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

Curve	Offload acceleration: nShield 500+, nShield 6000+ and nShield Connect 6000+
NISTP192	Yes
NISTP224	Yes
NISTP256	Yes
NISTP384	Yes
NISTP521	Yes
NISTB163	—
NISTB233	—
NISTB283	—
NISTB409	—
NISTB571	—
NISTK163	—
NISTK233	—
NISTK283	—
NISTK409	—
NISTK571	—
ANSIB163v1	—
ANSIB191v1	—
SECP160r1	Yes

Custom curves

nCipher 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* on page 48, above.

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

Brainpool curves

Brainpool curves are supported as a custom curve type. The following Brainpool curves are supported.

Curve size in bits	Curve-ID	Curve-ID (twisted representation)
160	brainpoolP160r1	brainpoolP160t1
192	brainpoolP192r1	brainpoolP192t1
224	brainpoolP224r1	brainpoolP224t1
256	brainpoolP256r1	brainpoolP256t1

Curve size in bits	Curve-ID	Curve-ID (twisted representation)
320	brainpoolP320r1	brainpoolP320t1
384	brainpoolP384r1	brainpoolP384t1
512	brainpoolP512r1	brainpoolP512t1

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* on page 135
- CNG:
  - Supported algorithms for CNG: *Supported algorithms for CNG* on page 158
  - Key exchange for CNG: *Key exchange* on page 160
- Symmetric and asymmetric algorithms: *Cryptographic algorithms* on page 258
- Using **generatekey** options and parameters to generate ECDH and ECDSA keys: *Key generation options and parameters* on page 261

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

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 nCipher 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.

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

Remote Operator support

Many nCipher 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 nCipher 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* on page 171.

For v12 and later, nCipher recommends that you use Remote Administration, which is more flexible than the Remote Operator functionality.

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.

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.

Ordering additional features

When you have decided that you require a new feature, you can order it from Sales. Before you call Sales, you collect information about your module as follows:

- If possible, make a note of the serial number. This can be found on the circuit board of the nShield modules.
- 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* on page 52.

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

- nCipher e-mails you the Feature Enabling Certificate.
- nCipher 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 the nCipher web site (https://www.ncipher.com).

Enabling features

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.

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 nCipher:

1. Insert the Feature Enabling card from nCipher 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* on page 52.

Enabling features without a smart card

The **Feature Enable Tool** can also obtain the Feature Enabling Certificate information supplied by nCipher 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.

Feature Enable Tool using PCIe modules

In this example, only one module (**`ESN E51C-D135-83F8`** is attached to the host.

```
                          Feature Enable Tool
                          ===================
                     payShield Activation
                      |  ISO Smart Card Support
                      |   |  Remote Operator
                      |   |   |  Korean Algorithms
                      |   |   |   |  SEE Activation (EU+10)
                      |   |   |   |   |  SEE Activation (Restricted)
                      |   |   |   |   |   |  CodeSafe SSL
                      |   |   |   |   |   |   |  Elliptic Curve algorithms
                      |   |   |   |   |   |   |   |  Elliptic Curve MQV
                      |   |   |   |   |   |   |   |   |  Accelerated ECC
Mod   Electronic      |   |   |   |   |   |   |   |   |   |
No.  Serial Number
1  E51C-D135-83F8 --  YES YES NO  YES YES YES YES YES NO  NO
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 :
```

Feature Enable Tool using Solo XC module

In this example, only one module (**ESN 4748-494A-4B4C**) is attached to the host.

```
                                    Feature Enable Tool
                                    ===================
                               payShield Activation
                               |  ISO Smart Card Support
                               |  |  Remote Operator
                               |  |  |  Korean Algorithms
                               |  |  |  |  SEE Activation (EU+10)
                               |  |  |  |  |  SEE Activation (Restricted)
                               |  |  |  |  |  |  CodeSafe SSL
                               |  |  |  |  |  |  |  Elliptic Curve algorithms
                               |  |  |  |  |  |  |  |  Elliptic Curve MQV
                               |  |  |  |  |  |  |  |  |  Accelerated ECC
                               |  |  |  |  |  |  |  |  |  |  HSM Base Speed
                               |  |  |  |  |  |  |  |  |  |  |  HSM Mid Speed
                               |  |  |  |  |  |  |  |  |  |  |  |  HSM High Speed
                               |  |  |  |  |  |  |  |  |  |  |  |  |  Loaded Objects Base Capacity
                               |  |  |  |  |  |  |  |  |  |  |  |  |  |  Loaded Objects Mid Capacity
                               |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  Loaded Objects High Capacity
Mod Electronic                 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
No.  Serial Number
1 4748-494A-4B4C -- N  N  N  N  N  N  N  N  N  N  N  N  N  N  N  N

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 :
```

If you select option 1, you are prompted to insert a Feature Enabling smart card into a module slot and then press `Enter`. The **Feature Enable Tool** then behaves in exactly the same way as if it were run with the Feature Enabling cards already in the slots.

If you select option 2, you are prompted for the location and file name of the Feature Enabling certificate. When you supply these, you are prompted for the module number and the feature.

Note: Remote Administration users should use their normal remote access solution to run the **Feature Enable Tool** and select option 2.

If you select option 3, you are prompted for the module number and the feature, and are then asked to enter the certificate one line at a time, followed by a period (“.” on a line by itself. You can also cut and paste a Feature Enabling certificate from an e-mail.

If you select option 4, the table of enabled features is written to a file. You can include this file when you request new features from nCipher. See *Ordering additional features* on page 51.

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.

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.

Electronic Serial Number (ESN)

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

ModuleID

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

The order in which buses are searched and the order of modules on a bus depends on the exact configuration of the host. If you add or remove a module, this can change the allocation of `ModuleID`s 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.

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.

Note: Before you install new hardware, check the release notes on the installation media supplied with your new module for information about specific compatibility issues, new features, and bug fixes.

1. Install the module hardware. Refer to the *Installation Guide* for information on installing nCipher hardware.
2. Add the module to the Security World. Refer to *Adding or restoring an HSM to the Security World* on page 77.

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.

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 in a command window with administrative privileges:

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

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

Similarly, you can restart the hardserver on the client, and where applicable the Remote Administration Service, by running the following commands in a command window with administrative privileges:

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

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

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

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

Chapter 5: 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.
- *Software and module configuration* on page 37

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* on page 77.

Note: 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* on page 37.

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. See *Hardserver start-up settings* on page 43 for more about configuring users for Remote Administration.

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* on page 63 .
- KeySafe
  See *Creating a Security World using KeySafe* on page 68.
- The CSP or CNG wizard
  See *Creating a Security World using the CSP or CNG wizard* on page 71.

You can use KeySafe, the nShield CSP wizard, and the **new-world** utility to create a Security World for which you can specify different **k** values for the ACS. However, if you want to create a Security World with advanced features, such as SEE-compatibility or the ability to replace an OCS, you cannot use the nShield CSP wizard.

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

Any Operator Cards created in a previous Security World, cannot be used in a new Security World. If you are replacing a 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* on page 102.

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.

Location of Security World files

Security World files are created or updated in the directory specified by the environment variable **NFAST_KMLOCAL** on the host. By default, this is %NFAST_KMDATA%\local.

Note: 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 Administrator rights that allow read and write operations, and the starting and stopping of applications.

Files and operations

Security World operations create or modify files in the %NFAST_KMDATA%\local directory as follows:

Operation	creates/modifies	the 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)

In this table:

- *ESN* is the electronic serial number of the module on which the Security World is created
- *IDENT* is the identifier given to the card set or key when it is created
- *NUMBER* is the number of the card in the card set
- *APPNAME* is the name of the application by which the key was created.

The *IDENT* of a card set is a 40-character string that represents the hash of the card set's logical token. The *IDENT* of a key is either user supplied or a hash of the key's logical token, depending on the application that created the key.

Required files

The following files must be present and up to date in the %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.

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
- Recovery and replacement options, which must be configured if the Security World, keys, or pass phrases 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.

Security World basic options

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

Cipher suite

You must decide whether to use a cipher suite that uses Triple DES, AES (standard), or AES (SP800-131A compliant) Security World keys. The Security World keys are generated during Security World creation, and protect the application keys and OCSs in the created Security World.

Note: Due to the additional primality checking required by SP800-131A, Security World generation and key generation operations will take longer in SP800-131A compliant Security Worlds.

Note: To create a Triple DES Security World, you must use the `new-world` command-line utility.

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

K and N

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.

Note: 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 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.

FIPS 140-2 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 140-2 standard at level 2. However, you can choose to enable compliance with the FIPS 140-2 standard at level 3.

Note: 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 140-2 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.

UseStrongPrimes Security World setting

When creating a Security World, the default setting for `UseStrongPrimes` depends on the FIPS level:

- *FIPS 140-2 level 3:* `UseStrongPrimes` is *on*, meaning that the Security World always generates RSA keys in a manner compliant with FIPS 186-3.
- *FIPS 140-2 level 2:* `UseStrongPrimes` is *off*, meaning that the Security World leaves the choice of RSA key generation algorithm to individual clients.

Enabling UseStrongPrimes increases the RSA key generation time by approximately 10 times. If you want to use a different `UseStrongPrimes` setting from its default setting, you must use the `new-world` command-line utility to create the Security World.

The **nfkminfo** utility shows the status of the Security World. The **enquiry** utility displays information related to an HSM.

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 on the module* on page 46.

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.

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.

Note: 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* on page 32. For details about performing OCS and softcard replacement operations, see *Replacing Operator Card Sets* on page 112 and *Replacing the Administrator Card Set* on page 121.

Pass phrase replacement

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

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

For details about performing pass phrase replacement operations, see *Changing unknown or lost pass phrase* on page 111.

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.

Note: 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.

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.

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 useful 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.

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.

Security World replacement options

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

If you replace an existing Security World, its %NFAST_KMDATA%\local directory is not overwritten but renamed %NFAST_KMDATA%\local_*N* (where *N* is an integer assigned depending 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 %NFAST_KMDATA%\local_*N* directory from the old Security World, you must delete it manually.

Creating a Security World using new-world, KeySafe, the CSP wizard, or the CNG wizard

Before you start

Before you start to create a Security World:

- The HSM must be in pre-initialization mode. See *Appendix K: Checking and changing the mode on an nShield Solo module* on page 268 or *Appendix L: Checking and changing the mode on an*

nShield Edge on page 275 for more about changing the mode.

- You must be logged in to the host computer as a user who is permitted to create privileged connections. For more information, see *Hardserver start-up settings* on page 43 and *server_startup* on page 252.
- You must have set the **NFAST_HOME** environment variable.

  Note: This variable is set by default during product software installation.
- 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* on page 59.
- You must have enough smart cards to form the Security World's card set.

When you have finished creating a Security World, you must restart the HSM in operational mode.

Creating a Security World using new-world

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

Note: With Security World Software v11.72 and later, pass phrases are limited to a maximum length of 254 characters, when using **new-world**. See *Maximum pass phrase length* on page 28 for more information.

Running the new-world command-line utility

Open a command prompt window and type the command:

```
new-world --initialize [--factory] [--no-remoteshare-cert] [--strict-fips-140-2-level-3]
[--no-strict-rsa-keygen|--strict-rsa-keygen] [--no-recovery] --cipher-suite=CIPHER-SUITE
[--nso-timeout=TIMEOUT] [--module MODULE] --acs-quorum=K/N [--reduced-features] [--pp-
min=LENGTH] FEATURES [--disablepkcs1pad]
```

In this command:

Option	Description
--initialize	This option tells **new-world** to create a new Security World, replacing any existing %NFAST_KMDATA%\local directory. **Note:** 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 %NFAST_KMDATA%\local directory in which these reside as %NFAST_KMDATA%\local*N* (where *N* is an integer assigned depending on how many Security Worlds have been previously saved during overwrites).
--factory	This option tells **new-world** to erase an HSM, restoring it to factory state.
--no-remoteshare-cert	This option tells **new-world** not to make the HSM a target for remote shares.

Option	Description
`--strict-fips-140-2-level-3`	This options tells **`new-world`** to create a Security World that conforms to the FIPS 140-2 requirements for roles and services at level 3. If you do not specify this flag, **`new-world`** creates a Security World that complies with FIPS 140-2 requirements for level 2. **Note:** 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.
`--no-strict-rsa-keygen` `--strict-rsa-keygen`	If you are using the **`--strict-fips-140-2-level-3`** flag to create a FIPS 140-2 level 3 Security World, you can use the **`--no-strict-rsa-keygen`** flag to *disable* UseStrongPrimes. Otherwise it is enabled by default. If you are creating a FIPS 140-2 level 2 Security World, you can use the **`--strict-rsa-keygen`** flag to *enable* UseStrongPrimes. Otherwise, it is disabled by default. If UseStrongPrimes is on, the Security World always generates RSA keys in a manner compliant with FIPS 186-3. Otherwise, the Security World leaves the choice of RSA key generation algorithm to individual clients.
`--no-recovery`	This option tells **`new-world`** to disable 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 nCipher 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.

Option	Description
--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* can be one of the following: • **DLf1024s160mDES3**, which uses a Triple DES key. • **DLf1024s160mRijndael**, which uses an AES key. • **DLf3072s256mRijndael**, which uses an AES key to create a Security World compliant with SP800-131A. **Note:** The **--disablepkcs1pad** option will only work on SP800-131A Security Worlds. **Note:** The default cipher suite is DLf3072s256mRijndael. For software versions before Security World Software v12.40, the default cipher suite is DLf1024s160mDES3.
--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.
--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. **Note:** 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. **Note:** 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 pass phrase recovery; no NVRAM, RTC, or FTO; and no NSO delegate keys. However, such a reduced-features Security World can perform many operations faster than more fully featured Security Worlds.

Option	Description
`--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. This option can only be used with a *DLf3072s256mRijndael* Security World. The other two Security World types cannot be used with this option and attempts to create them will fail. PKCS#1 v1.5 signature operations are not affected. PSS and OAEP are not affected.
`--pp-min=LENGTH`	This option enables a minimum pass phrase 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 pass phrase length check is then applied after the Security World is created. When enabled and you attempt to create a card pass phrase with fewer characters than the specified minimum length, the following warning message displays: **Warning: short passphrase** However, the pass phrase can still be used. Examples: `new-world --initialize --acs-quorum=K/N --cipher-suite=CIPHER-SUITE --pp-min=14` If `--pp-min=<length>` is not used, the minimum pass phrase length is set to the default value (0). **Note:** The minimum pass phrase length option is not applicable to software versions before Security World Software v11.72.
`--pp-strength`	This option enables pass phrases to have at least one uppercase, lowercase, number, and symbol. If the –pp-strength argument is omitted, the complexity requirements are not enforced.

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.

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

Note: 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 *OCS and softcard replacement* on page 61 and *Replacing Operator Card Sets* on page 112.
p	This feature enables pass phrase replacement; see *Pass phrase replacement* on page 61 and *Changing card and softcard pass phrase* on page 108.
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* on page 62).
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:

- **p**
- **nvram**
- **rtc**
- **fto**

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

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

Note: 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.

Note: 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
- Pass phrase 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).

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. See *Appendix K: Checking and changing the mode on an nShield Solo module* on page 268 or *Appendix L: Checking and changing the mode on an nShield Edge* on page 275 for more about changing the mode.

Note: If the HSM is in the pre-initialization mode, **new-world** prompts you for smart cards and pass phrases as required.

Creating a Security World using KeySafe

To create a Security World using KeySafe:

1. Start KeySafe. (For an introduction to KeySafe and information on starting the software, see *Using KeySafe* on page 186.)
2. Click the **World** menu button, or select **World** from the **Manage** menu. KeySafe opens the **World Operations** panel.
3. Click the **Initialize Security World** button.
4. If KeySafe finds existing Security World data, it takes you to the **Existing Security World** panel to confirm whether or not you want to delete this data by overwriting the existing Security World.

   Note: 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 %NFAST_KMDATA%\local directory in which these reside as %NFAST_KMDATA%\local *N* (where *N* is an integer assigned depending on how many Security Worlds have been previously saved during overwrites)

   To overwrite an existing Security World, click **Next** to go to the **Initialize Security World** panel.

5. On the **Initialize Security World** panel, you can configure the following options:
   a. The total number of administrator cards (*N*) that you wish to have in the ACS. This must be a value in the range 1 – 64.
   b. The number of administrator cards that are needed to authorize any action. This number (*K*) must be less than or equal to the total number of cards (*N*).

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

   c. The Cipher suite for the Security World—that is, whether the Security World key is to be an AES (original) or AES (SP800-131A compliant).

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

   d. Whether or not you want the new Security World to comply with the roles and services, key management, and self-test sections of the FIPS 140-2 standard at level 3.

      Note: This option provides compliance with the letter of the FIPS 140-2 level 3 standard, but does not improve the security of your keys. It is included for those customers who have a regulatory requirement for compliance.

   e. Whether or not the Security World is to permit receipt of remote operator card shares.
6. Click **Set Advanced Options** to open the **Initialize Security World Advanced Options** panel.

   Note: KeySafe always generates recovery and replacement data for the Security World key. However, if you want to be able to recover Operator Cards and application keys, you must proceed to the **Initialize Security World Advanced Options** panel. This allows you to create key recovery and pass phrase replacement data that is protected by the cryptographic keys on the ACS. This does not give nCipher, or any other third party, access to your keys. Keys can only be recovered by using the Administrator Cards.

   We recommend that you do not disable the recovery and replacement option

   If you disable OCS and softcard replacement, 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.

7. On the **Initialize Security World Advanced Options** panel you can configure the following options:
   a. Whether to enable key recovery

If you select **No** for the **Do you want to enable key recovery?** option, you cannot replace lost or damaged Operator Card Sets and, therefore, cannot access keys that are protected by such cards. This feature cannot be enabled later without reinitializing your Security World and discarding all your existing keys

   b. The number of administrator cards required to authorize key recovery
   c. Whether to enable passphrase recovery
   d. The number of administrator cards required to authorize passphrase recovery
   e. The number of administrator cards required to load this Security World onto a new HSM
   f. Whether to generate a foreign token key
   g. The number of administrator cards required to authorize foreign token key operations

   Click **OK** to save your selections and return to the **Initialize Security World** panel, or **Cancel** to return without saving.

8. Click **SEE options** to open the **Initialize Security World - SEE Options** panel.

   Note: If you wish to use SEE, you must have ordered and enabled it as described in *Enabling optional features on the module* on page 46

9. On the **Initialize Security World - SEE Options** panel, you can configure the following options:
   a. Whether to generate a real-time clock (RTC) authorization key
   b. The number of Administrator Cards required to change RTC details

      Note: This option is not applicable unless you have purchased and installed the CodeSafe Developer kit

   c. Whether to generate a nonvolatile memory (NVRAM) authorization key
   d. The number of Administrator Cards required to change NVRAM details

      Note: This option is not applicable unless you have purchased and installed the CodeSafe Developer kit

   e. The parameters for SEE debugging. These are:
      - **Restricted**
      - **Generate Authorization Key**
      - **No Access Control**.

      Note: To use extended debugging from the HSM, you must set SEEDebugging to **No Access Control**.

   f. The number of Administrator Cards required to change SEE debugging settings

   Click **OK** to save these settings and return to the **Initialize Security World** panel, or **Cancel** to return without saving

10. Check that the HSM card reader (slot 0) is empty, and then click **Commit**. This opens the **Create Administrator Card Set** panel.

    KeySafe prompts you to insert the cards that are to form the ACS.

11. Insert a blank card and click the **OK** button.
    a. If you insert a non-blank card that KeySafe can erase, the **Overwrite card** button is enabled, giving you the option of overwriting that card.
    b. When creating a card set, KeySafe recognizes a card that belongs to the set before the card set is complete. If you accidentally insert an already written card from the card set you are in the process of creating, KeySafe warns you of this.
    c. If you insert a blank card, KeySafe prompts you to click **OK** to set a passphrase.
12. Click **OK** to open the **Set Card Protection Passphrase** panel.

    To set a pass phrase for this Administrator Card:

    a. Select the **Yes** option
    b. Enter the same pass phrase in both text fields
    c. Click **OK**.

    A given pass phrase is associated with a specific card, so each card can have a different pass phrase. You can change these pass phrases at any time by using the KeySafe **Examine/Change Card** option (available from the **List Operator Card Sets** panel) or the `cardpassphrase` command-line utility.

    If you do not want to set a pass phrase for this Administrator Card
    a. Select the **No** option
    b. Click **OK**.

13. KeySafe then prompts you for the next card (if any).

    Note: When creating a card set, KeySafe recognizes a card that belongs 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, a warning message will be displayed and you will be prompted to insert a blank card.

14. After you have created all the Administrator Cards, Keysafe creates a new HSM key. When initialization is complete, KeySafe displays a message: **Security world successfully initialized**. Click **OK** to return to the introduction panel.
15. After you have added on HSM to the Security World, restart the HSM in the operational mode. See *Appendix K: Checking and changing the mode on an nShield Solo module* on page 268 or *Appendix L: Checking and changing the mode on an nShield Edge* on page 275 for more about changing the mode.

    If you have more than one HSM on this server, you can use the KeySafe **Reprogram Module** option (on the **World Operations** panel) to incorporate a new HSM into your Security World (or to restore an existing HSM after a firmware upgrade), as described in *Adding or restoring an HSM to the Security World* on page 77. Alternatively, you can also incorporate a new HSM into your Security World by using the `new-world` command-line utility; see *Adding an HSM to a Security World with new-world* on page 79

Creating a Security World using the CSP or CNG wizard

Note: If you have installed the Security World Software on a 64-bit machine, two different sets of nShield CSPs have been made available to you: one for 32-bit applications and one for 64-bit applications. Run the appropriate wizard for the application you intend to use (32-bit or 64-bit application).

If you intend to use both 32-bit and 64-bit applications, first run the 32-bit CSP wizard to create the Security World and install the CSPs. Then run the 64-bit wizard with the newly created Security World to install the 64-bit CSPs. Because the default SChannel CSP for 32-bit applications is set distinctly from the default SChannel provider for 64-bit applications, if you wish the nCipher SChannel CSP to be the default for *all* applications (both 32- and 64-bit), you must run both installers, selecting the option to set the default SChannel CSP on each.

Note: The CSP wizard includes SChannel settings and a modexp offload provider that are not found in the CNG wizard. The CNG wizard includes provider configuration options for different algorithms that are not found in the CSP wizard.

To create a Security World by using the nShield CSP or CNG wizard:

1. Run the wizard by double-clicking its shortcut in the Start menu under **All Programs** > **nCipher**. The wizard displays the welcome window.
2. Click the **Next** button.
   The wizard allows you to configure HSM Pool mode for the CAPI / CNG CSP.
3. Click the **Next** button.
   The wizard determines what actions to take based on the state of the Security World and the state of the HSMs that are attached to your computer:
   - If there is no existing Security World, the wizard offers the option to create a new security world or to install cryptographic acceleration only.
   - If there is an existing Security World, the wizard offers the option to use the existing security world, to create a new security world or to install cryptographic acceleration only.

   Note: The install cryptographic acceleration only option is supported in the nShield CSP wizard but not the CNG wizard.
4. Select one of the following options:
   - To create your first Security World, select **Create a new security world**
   - To replace the existing Security World, select **Create a new security world**
   - To use the existing Security World, select **Use the existing security world**, and click the **Next** button. If there is an HSM in initialization mode, the wizard assumes you are adding the HSM to the security world and continues from the step for adding an HSM, otherwise the wizard continues from choosing a key protection method.
   - To proceed without creating a Security World, select **Install cryptographic acceleration only** and click the **Next** button. The wizard installs the acceleration-only hook. This option only provides acceleration and does not use the HSM to manage keys. The wizard continues from installing the CAPI CSP or registering the CNG CSP.
5. Click the **Next** button.
   If there are no HSMs already in initialization mode the wizard displays the **Set Module State** panel. Ensure all HSMs are in initialization mode before continuing. See *Checking and changing the mode on an nShield Edge* on page 275 or *Checking and changing the mode on an nShield Solo module* on page 268 for more about changing the mode.
6. When all the HSMs are in the initialization mode, click the **Next** button. The wizard displays the **Administrator Card Set Properties** window.
7. Either accept the default 2/3 scheme for the ACS or select the **Use custom security policy** option and enter your own parameters for the sharing scheme:
   - The number of cards required (*K*) must be less than or equal to the total number of cards
   - The total number of cards (*N*) must be a value in the range 1 – 64.

8. Select the Cipher suite for the Security World—that is, whether the Security World key is to be an AES (original) or AES (SP800-131A compliant).
9. Select the **Configure advanced security world options** option if you want to continue on to further configuration options for the Security World when you have completed the selection of the options on the current screen. Select the **SEE settings** option to continue on to configure options for SEE.
10. If you require compliance to level 3 of the roles and services section of the FIPS 140-2 standard, select the **Create a FIPS 140-2 level 3 compatible security world** option. If you select this option, you must insert a card from the ACS or an existing OCS before you can create a new OCS.

    Note: The **Create a FIPS 140-2 level 3 compatible security world** option provides compliance with the letter of the FIPS 140-2 standard, but does not improve the security of your keys. It is included for those customers who have a regulatory requirement for compliance with this standard.
11. Click the **Next** button.
    If you did not select the **Configure advanced security world options** option, skip ahead to the next step in this process (in which you choose whether to enable remote card set support).
    If you selected the **Advanced settings** option, the World Advanced Options window is displayed.

Each tab in this window lets you specify the number of cards required to authorize specific tasks:

- On the **Add module** tab, select the number of cards required to add an HSM to the Security World.
- On the **Key rec.** tab, disable or enable OCS and softcard replacement and key recovery. If you leave this option enabled (which is the default), select the number of Administrator Cards required to replace an OCS or softcard and to recovery keys.

We recommend that you do not disable the recovery and replacement option.
If you disable OCS and softcard replacement, 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.

- On the **PP rec.** tab, disable or enable pass phrase recovery. If you enable pass phrase recovery, select the number of Administrator Cards required to recover a pass phrase.
- On the **NVRAM** tab, specify whether you require a key to authorize the creation of entries in the HSM's NVRAM. If you require this key, select the number of Administrator Cards required to access the NVRAM. The NVRAM authorization key is required if you want to use NVRAM to store SEE program data or you require NVRAM key storage.
- On the **RTC** tab, specify whether you require a key to authorize setting the HSMs real-time clock (RTC). If you require this key, select the number of Administrator Cards required to set the clock.
- On the **DSEE** tab, disable SEE debugging, enable SEE debugging for all users, or enable SEE debugging and create an associated key. If you enable debugging with this key, select the number of Administrator Cards required to authorize SEE debugging.
- On the **FTO** tab, specify whether you require a key to authorize foreign token operations (FTO). If you require this key, select the number of Administrator Cards required to perform foreign token operations.

12. Select **Enable remote cardset support** if required.
13. The wizard prompts you to insert the next card.
    If you want to protect this card with a pass phrase, select the **Card requires a pass phrase** option. The wizard prompts you to enter and confirm the pass phrase.
    Note: The **Next** button is only enabled when you have entered two matching pass phrases.
14. When you have entered the pass phrase, click the **Next** button.
15. Place the smart card in the HSM.
16. Click the **Next** button.
17. When the wizard successfully writes to this card, it prompts you to insert the next card and to enter a pass phrase for this card. Continue the process, following the onscreen instructions. When the wizard has written the number of cards you requested, it prompts you to reset HSM 1 to the operational state to enable you to choose a key-protection method and install the CSPs. These smart cards form the ACS for this Security World. The security of your key data depends on the security of these smart cards.
18. If there are no more HSMs to add to the Security World, skip ahead to the next step in this process (in which you restart the HSM in operational mode).

If there are further HSMs to add to the Security World, the wizard prompts you to insert a card from the ACS in the next HSM. Follow this process:

a. Insert one of the cards from the ACS that you have just created, and click the **Next** button. If the card is not from the ACS, the wizard prompts for another card.
b. If you defined a pass phrase for this smart card, the wizard prompts you to enter it. Enter the pass phrase, and click the **Next** button.
   If the pass phrase is incorrect, the wizard displays an error message. If you type the pass phrase incorrectly three times, the wizard does not continue, and this HSM is returned to the factory state.
   Note: If the wizard stops because the incorrect pass phrase has been entered three times, the HSM is not be added to the Security World. Run the wizard again in order to add this HSM to the Security World.
c. When your correct pass phrase is entered successfully, the wizard prompts you for further smart cards and their pass phrase until you have loaded the required number of Administrator Cards and pass phrases (as specified earlier in this process).

If there are yet more HSMs to be programmed on this host, the wizard returns you to repeat the process of inserting Administrator Cards and entering their pass phrases for each additional HSM being added to the Security World.
When all the HSMs have been added to the Security World, the wizard prompts you to reset module 1 to the operational state to enable you to choose a key-protection method and install the CSPs.

19. Restart the HSM in operational mode. See *Appendix K: Checking and changing the mode on an nShield Solo module* on page 268 or *Appendix L: Checking and changing the mode on an nShield Edge* on page 275 for more about changing the mode.
    The wizard prompts you to specify the method for protecting private keys generated by the CSPs.
20. If you are using Microsoft's CA and you want the wizard to prompt you every time you create or import a key, turn on the **Always use the wizard when creating or importing keys** option. If you leave this option turned off, the wizard attempts to protect the new or imported key by using the card currently in the HSM.
21. Choose a key protection method:
    - If you want to use OCSs to protect your keys, select the **Operator Card Set protection** option. This option is not supported if you have enabled HSM Pool mode for CAPI/CNG.
      If you choose to use OCSs to protect your keys and you do not already have a suitable OCS, you cannot proceed with Security World creation without creating an OCS. Click the **Next** button to continue by creating an OCS. For more information, see *Creating an Operator Card Set with the CSP or CNG wizard* on page 98.
    - If you want to use the HSM to protect your keys, select the **Module protection** option. If you choose module-protection for your keys, the CSP only creates keys that are protected by the Security World. This option is suited to applications where 24-hour operation is required, because no human intervention is required to use application keys.
      If you choose to use module-protected keys, you can click the **Next** button to continue by installing the CAPI CSP or registering the nShield CNG CSP. For more information, see *Installing the CAPI CSP* on page 151 and *Registering the CNG CSP* on page 155.

CSP containers

In versions of the CSP after 1.11.0, the Security World host data and CSP container information is stored in the Key Management Data directory.

The Key Management Data directory, together with the ACS and OCSs, contain all the Security World and CSP key container files.

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* on page 121.

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* on page 58). 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.

    Note: 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.

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* on page 76.

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

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* on page 228.

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.

Note: 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 hardserver* on page 55. 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, the nShield CSP wizard, or **new-world** when you created the Security World. This data is stored in the **local** directory within the Key Management Data directory.

  Note: 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 a user who is permitted to create privileged connections; see *Hardserver start-up settings* on page 43 and *server_startup* on page 252.
- Have started the HSM in pre-initialization mode. See *Appendix K: Checking and changing the mode on an nShield Solo module* on page 268 or *Appendix L: Checking and changing the mode*

on an nShield Edge on page 275 for more about changing the mode.
- Possess a sufficient number of cards from the ACS and the appropriate pass phrases.

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.

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.

Note: 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.

Adding an HSM to a Security World with the CSP or CNG wizard

To add an HSM to an existing Security World:

1. Ensure the HSM is in initialization mode and run the wizard by double-clicking its shortcut in the Start menu under **All Programs** > **nCipher**.
2. Click the **Next** button. The wizard allows you to configure HSM Pool mode for CAPI/CNG.
3. Click the **Next** button.
   If the wizard finds an existing Security World, it prompts you to specify whether you want to use the existing Security World or create a new Security World.
   If the wizard displays any other windows:
   a. Cancel the operation.
   b. Check that you have correctly set the environment variable **`NFAST_KMDATA`**.
   c. Copy the **`local`** sub-directory from the Key Management Data of another computer in the same Security World or from a backup tape of this computer to the Key Management Data directory of this computer.
   d. Run the wizard again.
4. Ensure that the **Use the existing security world** option is selected, and click the **Next** button.
   You can then proceed to add HSMs in the same manner that you add multiple HSMs when you create a Security World. Follow the instructions in the section *Creating a Security World using the CSP or CNG wizard* on page 71 from the step that describes what to do if there are further HSMs to add to the Security World.

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 tells **`new-world`** to add 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 tell **`new-world`** not to make the HSM 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. See *Appendix K: Checking and changing the mode on an nShield Solo module* on page 268 or *Appendix L: Checking and changing the mode on an nShield Edge* on page 275 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 pass phrases as required.

2. After **`new-world`** has reprogrammed the HSM, restart the HSM in the operational state. See *Appendix K: Checking and changing the mode on an nShield Solo module* on page 268 or *Appendix L: Checking and changing the mode on an nShield Edge* on page 275 for more about changing the mode.
3. Store the ACS in a safe place.

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

Transferring keys between Security Worlds

You must enter Administrator Cards in the client computer to transfer keys between Security Worlds. If your security policy does not permit this, do not attempt to carry out this procedure.

You use the command-line utilities **`mk-reprogram`** and **`key-xfer-im`** are used to transfer keys between Security Worlds.

Note: To transfer existing Security World data into an SP800-131A Security World, use the **`migrate-world`** command-line utility instead; see *Migrating keys to an SP800-131A Security World* on page 84.

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

To transfer keys between Security Worlds:

1. Ensure that the source Security World (from which you will transfer keys) has the necessary features enabled:
   - OCS and/or softcard replacement.
   - Key recovery or module-protected keys.

     Note: The destination Security World does not need to have these options enabled.
2. Move the Security World files (by default, these are the files in the %NFAST_KMDATA%\local Security World directory) for the source and destination Security Worlds into directories named **`source`** and **`destination`** respectively. For more information, see *Security World files* on page 58.
3. Decide which Security World to use as the *working Security World* when running the **`key-xfer-im`** command-line utility.
   The working Security World can be any FIPS 140-2 level 2 Security World. If both your source and destination security are compliant FIPS 140-2 level 3, you can create a temporary, dummy FIPS 140-2 level 2 Security World to use as the working Security World.
   The following table shows various possible configurations:

Source Security World	Destination Security World	Working Security World	Additional module to be added (**`mk-reprogram`**)	ACL export change options (**`key-xfer-im`**)
Level 2	Level 2	source	destination	–
Level 2	Level 2	destination	source	–
Level 2	Level 3	source	destination	**`--export-delete`**
Level 3	Level 2	destination	source	**`--export-add`**
Level 3	Level 3	Dummy	source and destination	–

   Note: In this table, Level 3 means a FIPS 140-2 level 3 compliant Security World. Level 2 means a FIPS 140-2 level 2 Security World.
4. Ensure that you have a quorum for each relevant card set:
   - ACS for the source and destination Security Worlds (and for the dummy Security World, if needed when both the source and destination are compliant with FIPS 140-2 level 3).
   - OCSs for the destination Security World.

     Note: If necessary, create OCSs for the destination Security World. For more information, see *Creating Operator Card Sets (OCSs)* on page 93.
5. Add the HSM to the working Security World. For more information, see *Adding or restoring an HSM to the Security World* on page 77.
6. Program the module key (or keys) from Security Worlds that are not the working Security World into the working Security World by running a command of the form:

```
mk-reprogram --owner <working_security_world_dir> add <non-working_security_world_dir>
```

In this command, <*working_security_world_dir*> is the Security World directory of the working Security World and <*non-working_security_world_dir*> is the working directory for the Security World that is not the working Security World. In the following example, the working Security World is the destination Security World:

```
mk-reprogram --owner C:\nfast\destination\local add C:\nfast\source\local
```

Supply any appropriate ACS cards (and pass phrases) as prompted.
If you are using a dummy Security World to transfer keys between two Security Worlds that are compliant with FIPS 140-2 level 3, you must run mk-reprogram twice: once to transfer module keys from the source and destination Security Worlds to the dummy Security World, as in the following example commands:

```
mk-reprogram --owner C:\nfast\Dummy\local add C:\nfast\Dummy\source\local
mk-reprogram --owner C:\nfast\Dummy\local add C:\nfast\destination\local
```

7. Transfer a key (or keys) by running the **`key-xfer-im`** command-line utility:

```
key-xfer-im SOURCE-KMDATA-LOCALDESTINATION-KMDATA-LOCALNEW-PROTECTKEY-FILE
[KEY-FILE ...] [NEW-PROTECTKEY-FILE [KEY-FILE ...]]
```

In this command:

- *<SOURCE-KMDATA-LOCAL>*
  The full path name of the **kmdata** file for the source Security World.
- *<DESTINATION-KMDATA-LOCAL>*
  The full path name of the **kmdata** file for the target Security World.
- *<NEW-PROTECT>*
  The protection for the key in the target Security World.
  You must specify either **--module** or **--cardset**. If you specify **--cardset**, it must be followed by the key hash for the destination card set.
  In addition, you can also specify options to configure the key protection further:
  - **--export-leave**
    This option is used to leave the key's list of operations requiring authorization from the Administrator Card Set (ACS) the same. This is the default.
  - **--export-add**
    This option is used to add export to the key's list of operations requiring authorization from the ACS. This option is only available when exporting keys from a FIPS 140-2 level 3 Security World into a FIPS 140-2 level 2 Security World.
  - **--export-delete**
    This option is used to remove export from the key's list of operations requiring authorization from the ACS. This option is only available when exporting keys from a FIPS 140-2 level 2 Security World into a FIPS 140-2 level 3 Security World.
  - **--aclbase-recovery**
    This option is used to base the Access Control List (ACL) of the exported key on the ACL in the recovery key blob. This is the default.
  - **--aclbase-working**
    This option is used to base the ACL of the exported key on the ACL in the working key blob.
- **<KEY-FILE>**
  This is the full path of source **kmdata** file for the key. The module must have module keys from both worlds: program it into one world with **new-world** and use **mk-reprogram** to add the other module key. If transferring between FIPS-140 level 3 and FIPS-140 level 2 worlds, the module owning world must be FIPS-140 level 2.

The following example command demonstrates transfer of a module key between source and destination Security Worlds that are both compliant with FIPS 140 2 level 3:

```
key-xfer-im C:\example\source\local\ C:\example\destination\local --module
C:\example\source\local\key_pkcs11_ua753157d8a9b86e943c5e4a6c100963f26839749a
```

The following example command demonstrates transfer of a card set key from a source Security World that is compliant with FIPS 140 2 level 3 to a destination Security World that is not:

```
key-xfer-im C:\example\destination\local C:\example\destination\local --cardset
1234578..cardsethash...abcdef  --export-add C:\nfast\example\local\key_pkcs11_
ua753157d8a9b86e943c5e4a6c100963f26839749a
```

The following example command demonstrates transfer of a card set key from a source Security World that is not compliant with FIPS 140 2 level 3 to a destination Security World that is :

```
key-xfer-im  C:\example\source\local  C:\example\destination\local --cardset
1234578..cardsethash...abcdef  --export-delete  C:\example\source\local\key_pkcs11_
ua753157d8a9b86e943c5e4a6c100963f26839749a
```

Supply appropriate cards and pass phrases for the ACS (source Security World) and OCSs (destination Security World) as prompted.

Note: You could use any directory instead of the **example** directory shown in these example commands.

8. If necessary, copy the Security World files to the Security World directory (by default, %NFAST_KMDATA%\local), and then add the HSM to the destination Security World. For more information, see *Adding or restoring an HSM to the Security World* on page 77.
9. Check that keys have been transferred. For example:
   - Check the Security World directory (by default, %NFAST_KMDATA%\local) to see if the key was successfully exported.
   - Verify that a module-protected key was imported successfully by running a command of the form:

```
preload --module-prot exit
```

Note: You can also use the **preload** command-line utility to load other types of keys, thereby also verifying that they have been imported successfully.

Security World migration

The current version of the Security World for nShield enables you to create a Security World that fully complies with NIST Recommendations for the Transitioning of Cryptographic Algorithms and Key Sizes (SP800-131A). The software also includes a **migrate-world** command-line utility that you can use for migrating existing keys into the new Security World. We recommend that where compliance with SP800-131A is required, you create a new Security World and new keys within that world. This utility is provided as a convenience for customers who require compliance with SP800-131A and who need to continue using existing keys.

Note: If your current Security World or keys are non-recoverable, you cannot use the migration utility to migrate keys. Contact Support.

Note: The **--disablepkcs1pad** option will only work on SP800-131A Security Worlds.

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 pass phrases.
- **Perform mode**: Enables you to migrate keys interactively.

Usage and options

```
migrate-world [OPTIONS] --source=<source-kmdata-path> [--plan | --perform]
```

Option	Enables you to...
--version	View the version number of the utility.
-h, --help	Obtain information about the options you can use with the utility.
-m, --module	Specify which module ID to use. If no module is specified, the default is module 1.
-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 very many different card sets or softcards to migrate. (0=unlimited, default=0)
-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 very many 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.
--source=*SOURCE*	Specify the path to the folder that contains the source Security World data.
--plan	View the list of steps that will be carried out .
--perform	Migrate keys interactively.

Migrating keys to an SP800-131A Security World

Note: Throughout the following sections, the term:

- *Source world* refers to the source Security World from which you want to migrate keys.
- *Destination world* refers to the SP800-131A Security World to which you want to migrate keys.

Note: The **--disablepkcs1pad** option will only work on SP800-131A Security Worlds.

Migrating keys to an SP800-131A Security World involves:

1. Preparing for migration, which includes setting up a destination world (if it does not exist already).
2. Migrating keys by running the migration utility.

Preparing for migration

Before you begin:

- Install the latest version of the Security World Software from the installation disc. See the Installation Guide for more information.
- Copy the source Security World data to a location that can be accessed by the migration tool.

- If the destination world does not exist already, create a new destination world and program the module to use this world. For instructions, see:
  - *Creating a Security World using new-world* on page 63
  - *Creating a Security World using KeySafe* on page 68
  - *Creating a Security World using the CSP or CNG wizard* on page 71
- Run the utility in the **plan** mode. The utility returns a list of steps for migration along with the details of the required card sets and pass phrases. Ensure that you have:
  - The required card sets and pass phrases of the source and destination worlds.
  - The appropriate number of blank smart cards to create the required card sets.

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

Migrating keys

Note: If your source world, destination world, or keys are non-recoverable, you cannot migrate keys with the migration utility. Contact Support.

Note: If you do not have a quorum of the ACS of the source and destination worlds, you cannot migrate keys.

Note: During migration, the keys are temporarily in a vulnerable state. To ensure the security of your keys, we recommend that:

- The migration process is overseen by ACS-holding personnel.
- 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.

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* on page 83.
   Ensure that the data for the current (destination) world is in the standard location for Security World data, derived from one of the following:
   - Either the environment variable **NFAST_KMLOCAL** or **NFAST_KMDATA**.
   - The default directory: **C:\ProgramData\Key Management Data\local**, or **C:\Documents and Settings\All Users\Application Data\nCipher\Key Management Data\local**, as appropriate.
2. 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.
   The next steps in the migration procedure vary depending on whether you are transferring keys from:
   - Either a FIPS 140-2 level 3 or FIPS 140-2 level 2 Security World to a FIPS 140-2 level 2 Security World.
   - A FIPS 140-2 level 2 Security World to a FIPS 140-2 level 3 Security World.
   - A FIPS 140-2 level 3 Security World to another FIPS 140-2 level 3 Security World.

   Depending on the scenario, the utility guides you through specific steps, prompting you to supply the required card sets and pass phrases. The high-level steps and the requirements for each task are described in the following sections. For detailed steps, follow the onscreen instructions.

Note: The utility prompts you to change the module state to either `initialization` or `operational` at various points during the procedure.
See *Appendix K: Checking and changing the mode on an nShield Solo module* on page 268 or *Appendix L: Checking and changing the mode on an nShield Edge* on page 275 for more about changing the mode.

Any Security World to FIPS 140-2 level 2 Security World

Step	Requirements
Program the source module key into the destination world. This key is used for decrypting keys when they are transferred to the destination world.	• ACS of the destination world. • ACS of the source world.
Create replacement softcards and/or Operator Card Sets, as appropriate. You can set pass phrases of your choice.	• ACS of the destination world. • Blank smart cards to create Operator Card Sets, if appropriate.
Migrate keys to the destination world. **Note:** The module is programmed to use the new world by default.	The replacement softcards or Operator Card Sets and pass phrases.

FIPS 140-2 level 2 Security World to FIPS 140-2 level 3 Security World

Step	Requirements
Create replacement softcards and/or Operator Card Sets, as appropriate. You can set pass phrases of your choice.	• ACS of the destination world. • Blank smart cards to create the OCS, if appropriate.
Program the source world into the module.	ACS of the source world.
Program the destination module key into the source world. This key is used for encrypting keys when they are transferred to the destination world.	• ACS of the source world. • ACS of the destination world.
Migrate keys to the destination world.	• ACS of the source world. • The replacement softcards or OCS and pass phrases.
If you want to start using the destination world, program the module to use the new Security World.	ACS of the destination world.

FIPS 140-2 level 3 Security World to FIPS 140-2 level 3 Security World

Note: If both Security Worlds are FIPS 140-2 level 3, the utility creates an intermediate FIPS 140-2 level 2 Security World (intermediate world), which is removed after migration. The utility also prompts you to create a 1-of-1 ACS for the intermediate world; we recommend that you destroy this ACS after migration.

Step	Requirements
Create replacement softcards and/or Operator Card Sets, as appropriate. You can set pass phrases of your choice.	• ACS of the destination world. • Blank smart cards to create the OCS, if appropriate.
Create an intermediate world and card set.	Blank smart card to create the ACS for the intermediate world.
Program the destination module key into the intermediate world. This key is used for encrypting keys when they are transferred to the destination world.	• ACS of the intermediate world. • ACS of the destination world.
Program the source module key into the intermediate world. This key is used for decrypting keys when they are transferred to the intermediate Security World.	• ACS of the intermediate world. • ACS of the destination world.
Migrate keys to the intermediate world.	ACS of the source world.
Migrate keys to the destination world.	• ACS of the intermediate world. • The replacement softcards or OCS and pass phrases.
If you want to start using the new world, program the module to use the destination world.	ACS of the destination world.

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>
```

Note: Do not use the **nfkmverify** utility with the default **-C** option. If you use this option, the utility returns errors because the certificate in the old Security World and the certificate in the new Security World are different.

Troubleshooting

Note: If you encounter any errors that are not listed in the following table, contact Support.

Error	Explanation	Action
	Any migrateable keys found in the source world already exist in the destination world.	
No keys to migrate.	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 world has indistinguishable cardsets or softcards. • Destination world has indistinguishable keys.	There are irregularities in one of the Security Worlds, but these irregularities *do not* affect the migration process.	None.
• Destination world has indistinguishable card sets or softcards. • Source world has indistinguishable card sets or softcards. • Source world has indistinguishable keys. Cannot determine protection of keys.	There are problems with one of the Security Worlds.	Contact Support.
• Source world not recoverable. • Destination world not recoverable.	The source and destination worlds are 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*.	• The path for the source world is wrong. • There is no Security 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. Ensure that the destination world is the current 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.	The software installation is partially completed.	
NAME utility is too old, need at least version *VERSION*.	• 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 *Migrating keys on page 85*.

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 a user who is permitted to create privileged connections.
- Have started the module in the pre-initialization mode. See *Appendix K: Checking and changing the mode on an nShield Solo module on page 268* or *Appendix L: Checking and changing the mode on an nShield Edge on page 275* 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 nCipher (that is, it has a random module key and a known K_{NSO}).

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 this command:

Option	Description
-e, **--factory**	These options tell **new-world** to 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.

Output

If **new-world** successfully erased a module, it does not display any messages. Otherwise, **new-world** returns an error message.

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* on page 186.)
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.
5. KeySafe erases all secrets from the module, returning it to its factory state.

Note: 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.

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 this 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.

Note: The **--disablepkcs1pad** option will only work on SP800-131A Security Worlds.

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.

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 standard nShield 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 softcards* on page 102.

Note: 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 nCipher 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 files in the Key Management Data directory.

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.

Chapter 6: 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 pass phrases.
OCS	Keys that are protected by an OCS can only be used by the operator who possesses the OCS and any relevant pass phrases (if set).

For more information about creating a Security World, see *Creating and managing a Security World* on page 57.

For more information about key management, see *Working with keys* on page 177.

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* on page 177).

If you want to use the Remote Operator feature to configure smart cards for use with a remote module, see *Remote Operator* on page 171.

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 module 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 pass phrases for any card within a given set
- Formal FIPS 140-2 level 3 compliance.

Note: Some third-party applications impose restrictions on the OCS smart card quorums (*K/N*) or the use of smart card pass phrases. For more information, see the appropriate integration guide for the application. Integration guides for third-party applications are available from the nCipher web site: https://www.ncipher.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* on page 94
- KeySafe, as described in *Creating an Operator Card Set with KeySafe* on page 96
- The nShield CSP wizard, as described in *Creating an Operator Card Set with the CSP or CNG wizard* on page 98
- The nShield CNG wizard, as described in *Microsoft Cryptography API: Next Generation (CNG)* on page 154.

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* on page 27.

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 supplies the final required pass phrase. 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.

FIPS 140-2 level 3-compliant security worlds

When you attempt to create an OCS for a security world that complies with FIPS 140-2 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.

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. **Note:** 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.
-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 pass phrase replacement for this OCS is disabled. Setting this option overrides the default setting, which is that the card pass phrases are replaceable. You can specify the enablement of pass phrase 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* on page 171. **Note:** 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.

Note: With Security World Software v11.72 and later, pass phrases are limited to a maximum length of 254 characters, when using **createocs**. See *Maximum pass phrase length* on page 28 for more information.

If you have created a FIPS 140-2 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 pass phrase.
   Note: The nCipher PKCS #11 library requires Operator Cards with pass phrases.
   Note: Some applications do not have mechanisms for entering pass phrases. Do not give pass phrases to Operator Cards that are to be used with these applications.
3. Type a pass phrase and press `Enter`. Alternatively, press `Enter` if you do not want this card to have a pass phrase.
   A pass phrase can be of any length and can contain any character that you can type.
   If you entered a pass phrase, **createocs** prompts you to confirm it.
4. Type the pass phrase again and press `Enter`.
   If the pass phrases do not match, **createocs** prompts you to input and confirm the pass phrase 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.

Creating an Operator Card Set with KeySafe

KeySafe enables you to create OCSs with:

- Their own names
- *K/N* policies
- Optional pass phrases for any card within the OCS
- Formal FIPS 140-2 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* on page 186.)
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 pass phrase recovery will be enabled for the OCS. (Only available if the Security World has pass phrase 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* on page 171.
   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.

   Note: 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.

   Note: If you are creating the card set in a FIPS 140-2 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.

   Note: 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).

   Note: 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 pass phrase for the currently inserted card. Each card in a set can have an individual pass phrase, and you can also create a set in which some cards have pass phrases and others do not.

8. If setting a pass phrase for the currently inserted card, enter the same pass phrase in both text fields. A pass phrase can contain any characters you can type except for tabs or carriage returns (because these keys are used to move between data fields).

   Note: You can change a pass phrase at any time. If you do not set a pass phrase now, you can use the KeySafe **Change Pass Phrase** option (on the **Examine/Change Card** panel) to add one later. Likewise, if you later decide that you do not need a pass phrase on a card, you can use this option to remove it.
9. After entering your desired pass phrase (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.

Creating an Operator Card Set with the CSP or CNG wizard

You can use the nShield CSP or CNG wizard to create a *K/N* OCS that is suitable for use with the nShield Cryptographic Service Provider (CSP) or Cryptography API: Next Generation (CNG), as appropriate. You can only create an OCS using the CSP or CNG wizard if you already have a Security World and have an ACS available for that Security World.

To create an OCS using the CSP or CNG wizard, follow these steps:

1. Ensure that you have created the security world and that at least one HSM is in the operational state.
2. Run the wizard by double-clicking its shortcut in the Start menu under **All Programs** > **nCipher**.
3. The wizard displays the welcome screen.
4. Click the **Next** button. The wizard allows you to configure HSM Pool mode for CAPI/CNG.

   Note: Do not enable HSM Pool mode when creating an Operator Card Set because HSM Pool mode only supports module-protected keys.
5. Click the **Next** button.

The wizard determines what actions to take based on the state of the security world and of the HSMs that are attached to your computer:

- If the wizard cannot find the security world, it prompts you to create a new security world or to install cryptographic acceleration only.
  In such a case, you should:
  - cancel the operation
  - check that the environment variable `NFAST_KMDATA` is set correctly
  - Copy the `local` sub-directory from the Key Management Data directory of another computer in the same Security World or from a backup tape of this computer to the Key Management Data directory of this computer.
  - run the wizard again.
- If there is an existing security world, the wizard gives you the option of using the existing security world, creating a new security world or installing cryptographic acceleration only.
  - In order to use the existing security world, ensure that the **Use the existing security world** option is selected, and click the **Next** button.
  - If there are any hardware security modules in the pre-initialization state, the wizard adds them to the security world; see *Adding or restoring an HSM to the Security World* on page 77.

6. When at least one hardware security module is in the operational state, the wizard prompts you to select a method to protect private keys generated by the CSPs.
7. Ensure that the **Operator Card Set** option is enabled, and then select the **Create a new Operator Card Set** option.
   If you want the OCS to be persistent, select the **Persistent** option. Persistence is described in *Persistent Operator Card Sets* on page 94.
8. Click the **Next** button, and if you have a FIPS world, the wizard prompts you to insert a card created with the current security world.

   Note: This shows that your security world is compliant 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.

   Under the constraints of level 3 of the FIPS 140-2 standard, Operator Cards cannot be created without authorization. To obtain authorization, insert any card from the ACS or any Operator Card belonging to the current security world.
   The wizard does not enable the **Next** world, the wizard warns you and prompts you for another card.
9. Click the **Next** button.
   The wizard prompts you for a smart card to use as the first card in the OCS.
10. Insert a blank smart card to be used as the Operator Card, and click the **Next** button.

Do not use a card from the ACS or an existing Operator Card.
If you insert a card that is not blank, the wizard asks you if you want to erase it.

11. When you have inserted an appropriate card, the wizard prompts you for the name of the card and, if required, a pass phrase.
    If you want to protect this card with a pass phrase, turn on the **Card will require a pass phrase** option, and enter the pass phrase. You must enter the pass phrase in both fields to ensure that you have typed it correctly.

    Note: Operator Cards with pass phrases are required by the nCipher PKCS #11 library.

12. If you have not yet written all the smart cards in the OCS, the wizard prompts you for another card. Repeat the appropriate preceding steps of the OCS creation process for all smart cards in the set.
13. When the wizard has finished creating the OCS, it displays a screen telling you this. If you want to create another OCS, click the **Back** button on this screen.
    When you have created all the OCSs that you require, click the **Next** button to install the CAPI CSP or register the CNG CSP. For more information, see *Installing the CAPI CSP* on page 151 or *Registering the CNG CSP* on page 155.

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 pass phrase; 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 pass phrase 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.

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

Note: Softcards are not supported for use with the nCipherKM JCA/JCE CSP in Security Worlds that are compliant with FIPS 140-2 level 3.

Note: 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 pass phrase 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* on page 100)
- KeySafe (see *Creating softcards with KeySafe* on page 101)

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 pass phrase to be replaceable or non-replaceable. To create a softcard with a replaceable pass phrase, run the command:

```
ppmk --new --recoverable NAME
```

To create a softcard with a non-replaceable pass phrase, 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 140-2 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 pass phrase.
2. When prompted, type a pass phrase for the new softcard, and press `Enter`.
   A pass phrase 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).
3. When prompted, type the pass phrase again to confirm it, and press `Enter`.
   If the pass phrases do not match, **ppmk** prompts you to input and confirm the pass phrase again.

After you have confirmed the pass phrase, **ppmk** completes creation of the new softcard.

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* on page 186.)
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 pass phrase replacement to be enabled for the softcard.
      Note: 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**.
   Note: If you are creating the softcard in a FIPS 140-2 level 3 security world, insert an Administrator Card or an existing Operator Card when prompted.
   The **Set Softcard Protection Pass Phrase** pane is displayed.
6. Set a pass phrase for the softcard by entering the same pass phrase in both text fields.
   A pass phrase 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 pass phrase at any time. You must provide a passphrase for each card.

7. After entering your desired pass phrase 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.

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 --ignoreauth`. 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).

Note: 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 `%NFAST_KMDATA\local%` directory after you have erased the last card.

Note: 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* on page 105.

FIPS 140-2 level 3-compliant Security Worlds

When you attempt to erase cards for a Security World that complies with FIPS 140-2 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.

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* on page 186.)
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 pass phrase (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.

   Note: 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 **%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.

   Note: You can erase an entire card set at one time with the KeySafe **Discard Card Set(s)** feature.

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).

Note: 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 140-2 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**.

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.

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.

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 140-2 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.

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 pass phrase 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 pass phrase is associated with a softcard, you can use the **ppmk** command-line utility.

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.

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* on page 186.)

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 pass phrase (if it has one)
- Give a pass phrase to a card that does not already have one
- Remove a pass phrase from a card that currently has one
- Erase the card.

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* on page 186.)
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* on page 104)
- Create a new card set (see *Creating an Operator Card Set with KeySafe* on page 96)
- Replace an Operator Card Set (see *Replacing OCSs with KeySafe* on page 113)
- Replace an Administrator Card Set (see *Replacing an ACS with KeySafe* on page 122)
- Discard a card set (see *Erasing cards with KeySafe* on page 102).

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
```

Note: Not all software can give names to individual cards.

Viewing softcards

To view softcards, use KeySafe or the command line. The command line provides several options for viewing softcard information.

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* on page 102.

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
```

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**).

Verifying the pass phrase of a card or softcard

Verifying the pass phrase of a card with cardpp

To verify the pass phrase 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 tells **cardpp** to check the pass phrase.
`--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 pass phrase is set or prompts you to enter the pass phrase and checks to see if it is correct.

Verifying the pass phrase of a softcard with ppmk

In order to verify the pass phrase 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 pass phrase 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 pass phrase and then tells you whether the pass phrase you entered is correct for the specified softcard.

Changing card and softcard pass phrase

Each softcard or card of a card set can have its own individual pass phrase: you can even have a card set in which some cards have a pass phrase and others do not, and you can have distinct softcards that nevertheless use the same pass phrase. A pass phrase can be of any length and can contain any characters that you can type.

Normally, in order to change the pass phrase of a card or softcard, you need the card or softcard and the existing pass phrase. Known card pass phrase can be changed using KeySafe or the **cardpp** command-line utility; softcard pass phrase can be changed using KeySafe or the **ppmk** command-line utility. You can also add a pass phrase to a card or softcard that currently does not have one or remove a pass phrase from a card that does currently have one.

If you generated your Security World with the pass phrase replacement option, you can also replace the pass phrase of a card or softcard even if you do not know the existing pass phrase. Such a pass phrase replacement operation requires authorization from the ACS.

Changing known pass phrase

To change a card pass phrase, you need the card and the old pass phrase.

Each card in a set can have its own individual pass phrase. You can even have a set in which some cards have a pass phrase and others do not.

Note: Prior to Security World Software v11.72, we set no absolute limit on the length of a pass phrase. However, some applications may not accept a pass phrase 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. nCipher recommends that your password only contains 7-bit ASCII characters:

```
A-Z, a-z, 0-9, ! @ # $ % ^ & * - _ + = [ ] { } | \ : ' , . ? / ` ~ " < > ( ) ;
```

See *Maximum pass phrase length* on page 28 for more about pass phrase length when using Security World Software v11.72.

Changing known pass phrase with KeySafe

To change a known pass phrase for an Operator Card using KeySafe:

1. Start KeySafe. (For an introduction to KeySafe and information on starting the software, see *Using KeySafe* on page 186.)
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 pass phrase**. The Set Card Protection pass phrase panel is displayed.
5. Enter the old pass phrase, and click the **OK** button.
6. A screen is displayed asking **Do you want to set a pass phrase?**. Select **Yes**.
7. Enter your new pass phrase, and enter it again in the second box as confirmation of the change.
8. Click **OK**.

Changing a known softcard pass phrase with KeySafe

To change a known pass phrase for a softcard using KeySafe:

1. Start KeySafe. (For an introduction to KeySafe and information on starting the software, see *Using KeySafe* on page 186.)
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 pass phrase, and click the **Change Pass phrase** button. KeySafe takes you to the **Change/Recover Softcard Pass phrase** panel.
   Note: If a softcard is listed as PIN Recovery Enabled = No, then you will be unable to change the pass phrase.
4. Select the softcard whose pass phrase you want to change, and click the **`Change Pass phrase`** button. KeySafe takes you to the **Get Softcard Protection pass phrase** panel.
5. Enter the old pass phrase, and click the **OK** button.
   KeySafe either displays an error dialog (if the pass phrase is not correct) or takes you to the **`Set Softcard Protection`** pass phrase panel.
6. Enter your new pass phrase, and enter it again in the second field to confirm the pass phrase is correct.
7. Click the **OK** button.
   After changing a pass phrase, KeySafe displays a dialog to confirm that the pass phrase has been successfully changes.
8. Click the **OK** button to continue using KeySafe.

Changing known pass phrase with cardpp

Each card in a card set can have its own individual pass phrase. You can even have a set in which some cards have a pass phrase and others do not. A pass phrase can be of any length and can contain any characters that you can type.

Note: With Security World Software v11.72 and later, pass phrases are limited to a maximum length of 254 characters, when using **cardpp**. See *Maximum pass phrase length* on page 28 for more information.

To change a known card's pass phrase 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 pass phrase you want to change. (If there is a card already in the slot, you are not prompted.)
3. If prompted, enter the existing pass phrase for the card (If the card has no current pass phrase you are not prompted.) If you enter the pass phrase correctly, **cardpp** prompts you to enter the new pass phrase.
4. Enter a new pass phrase, and then enter it again to confirm it.
   After you have confirmed the new pass phrase, **cardpp** changes the card's pass phrase.

Changing known softcard pass phrase with ppmk

Note: With Security World Software v11.72 and later, pass phrases are limited to a maximum length of 254 characters, when using **ppmk**. See *Maximum pass phrase length* on page 28 for more information.

To change a known softcard's pass phrase when you know the pass phrase, follow these steps:

1. Give the following command:

   ```
   ppmk --change NAME|IDENT
   ```

   In this command, you can identify the softcard whose pass phrase 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 pass phrase.
2. Type the old pass phrase, and press `Enter`. If you enter the old pass phrase correctly, **ppmk** prompts you to enter the new pass phrase.
3. Type the old pass phrase, and press `Enter`. Type the new pass phrase again, and press **Enter** to confirm it.
   After you have confirmed the new pass phrase, **ppmk** then changes the softcard's pass phrase.

Changing unknown or lost pass phrase

Changing unknown card pass phrase with cardpp

If you generated your Security World with the pass phrase replacement option, you can change the pass phrase of a card even if you do not know its existing pass phrase. Such a pass phrase replacement operation requires authorization from the ACS.

To change an unknown card pass phrase 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 pass phrase replacement.
- When prompted, insert the Operator Card whose pass phrase you want to replace. To replace its pass phrase:
  a. When prompted, type the new pass phrase, and then press `Enter`.
  b. When prompted, type the new pass phrase again to confirm it, and then press `Enter`. **cardpp** sets the new pass phrase, and then prompts you for another Operator Card.
- Repeat the process in the previous step to change the pass phrase on further cards, or press `Q` to quit.

Only insert Administrator Cards into a hardware security module that is connected to a trusted server.

Replacing unknown pass phrase with ppmk

If you generated your Security World with the pass phrase replacement option, you can change the pass phrase of a softcard even if you do not know its existing pass phrase. Such a pass phrase replacement operation requires authorization from the ACS.

To change an unknown softcard pass phrase 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 pass phrase replacement.
3. When prompted, type the new pass phrase, and then press `Enter`.

4. When prompted, type the new pass phrase again to confirm it, and then press `Enter`.
   If the pass phrase does not match, **ppmk** prompts you to input and confirm the pass phrase again.

After you successfully confirm the new pass phrase, **ppmk** finishes configuring the softcard to use the new pass phrase.

Only insert Administrator Cards into a hardware security module that is connected to a trusted server.

Replacing Operator Card Sets

Note: 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.

Note: 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* on page 102.

Deleting the information about an OCS from the host does not remove the data for keys protected by that card set. On the KeySafe **List Keys** panel (reachable from the **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, nCipher 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
  Note: 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 nCipher PKCS #11 library version 1.6 or later

If you initialized the token using `ckinittoken` from the nCipher 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
  Note: 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 nCipher PKCS #11 library version 1.6 or later.
  Note: 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* on page 37.

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)* on page 93.

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* on page 186.)
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**.
   Note: 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.
   Note: 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.

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.

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* on page 118.
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* on page 116.
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* on page 119.

   Note: 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* on page 117.
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* on page 118.
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* on page 119.
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* on page 118.
   rocs prompts you to insert a card from the ACS.
8. Insert a card from the ACS.
   rocs prompts you for the pass phrase 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 pass phrase 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 pass phrase 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* on page 119). 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* on page 118).

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*

Note: 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`.

help

With no arguments specified, **help** shows a list of available commands with brief usage messages 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**.

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.

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.

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* on page 120.

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.

Note: 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.

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.

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.

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.

rescan

This command updates the card set and key information.

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.

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.

status

This command lists the currently selected hardware security module and target OCS or softcard.

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.

unmark *key-spec*

This command unmarks the listed keys. Unmarked keys are not recovered.

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* on page 120.
`-k`, **`--keys=`***KEYS-SPEC*	These options select the keys to be recovered. For more information, see *Specifying keys* on page 120.
`-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* on page 120.
`-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* on page 114.

Note: 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.

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.

Note: 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.

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"`

Value	Description
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.

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
- The KeySafe **Replace Adminstrator Card Set** option
  Accessed from the **Card Operations** panel

Note: If further cards are damaged, you may not be able to re-create your Security World.

Note: You cannot mix standard 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.

Note: 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.

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* on page 186).
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.

   Note: 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.

   Note: 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 Pass Phrase** panel.
8. If you want to set a pass phrase for this Administrator Card:
   a. Select the **Yes** option.
   b. Enter the same pass phrase in both text fields.
   c. Click the **OK** button.

   KeySafe then prompts you for the next card (if any). A given pass phrase is associated with a specific card, so each card can have a different pass phrase. You can change these pass phrases 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 pass phrase 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* on page 102.

Replacing an Administrator Card Set with racs

The **racs** utility creates a new ACS to replace a set that was created with the **new-world** utility.

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* on page 102.

Chapter 7: Application interfaces

This chapter explains how to use an HSM with various types of application:

- Cryptographic Hardware Interface Library (CHIL) applications
- nCipherKM JCA/JCE CSP
- PKCS #11 applications
- nShield native and Custom applications
- CodeSafe applications
- Microsoft CAPI CSP
- Microsoft CNG CSP.

Note: For information about using the Microsoft Cryptographic API, see the appropriate third-party integration guide at: https://www.ncipher.com.

You can use KeySafe or the **generatekey** command-line utility to generate or import keys for use with your applications (see *Working with keys* on page 177). By default, KeySafe uses the same mechanisms and supports the same applications as the **generatekey** command-line utility.

Note: By default, any user is allowed to use any application that uses an nShield HSM.

Cryptographic Hardware Interface Library (CHIL)

The Cryptographic Hardware Interface Library (CHIL) is a simple programming interface for accelerating modulo exponentiation and accessing the RSA/DSA keys that are used by some application software. The Cryptographic Hardware Interface Library supports only RSA/DSA and Diffie-Hellman keys.

If your application offers RSA/DSA keys in hardware support through the Cryptographic Hardware Interface Library, use the **hwcrhk** key type.

Some Cryptographic Hardware Interface Library applications that do not support the **hwcrhk** key can still be configured for key storage by using the **embed** key type (provided that they can read **PEM** (Privacy Enhanced Mail) format key files and use the Cryptographic Hardware Interface Library for all cryptographic operations).

By default the library makes use of all hardware security modules that it finds when starting up. With module firmware version 2.65.2 or later, if your CHIL application only uses module protected keys, you can use HSM Pool mode with multiple hardware security modules to make your application tolerant of hardware or network errors as long as one module remains functional. For CHIL, HSM Pool mode is enabled by setting the **NFAST_HWCRHK_HSM_POOL** environment variable in the environment of the process running the application that wants to use HSM Pool mode, see *Setting environment variables* on page 40.

Using keys

Configure the application to load the Cryptographic Hardware Interface Library plug-in: *%NFAST_HOME%***toolkits\nfhwcrhk\nfhwcrhk.dll**.

Refer to the documentation for your application.

Generating keys

Generate the key with KeySafe or **generatekey**, choosing a new identifier for the key; see *Generating keys* on page 177. Use the **hwcrhk** or **embed** key type, as appropriate.

The key identifier can only contain digits and lowercase letters; it *cannot* contain spaces, underscores (_), or hyphens (-).

nCipherKM JCA/JCE CSP

Overview of the 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 nShield HSMs. This provider is used with the standard JCE (Java Cryptographic Extension) programming interface.

To use the nCipherKM JCA/JCE CSP, you must install:

- The **javasp Java Support (including KeySafe)** bundle
- The **jcecsp nCipherKM JCA/JCE provider classes** component.

See the Installation Guide for more about:

- The bundles and components supplied on your Security World Software installation media
- The versions of Java that are supported by the Security World Software

Detailed documentation for the JCE interface can be found on the Oracle Technology web page https://docs.oracle.com/javase/8/docs/technotes/guides/security/crypto/CryptoSpec.html.

Note: Softcards are not supported for use with the nCipherKM JCA/JCE CSP in Security Worlds that are compliant with FIPS 140-2 level 3.

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* on page 55.
2. Copy the **nCipherKM.jar** file from the *%NFAST_HOME%***java\classes** directory to the extensions folder of your local Java Virtual Machine installation.
   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`
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).

3. Add *%JAVA_HOME%/*`bin` to your **`PATH`** system variable.
4. Install the unlimited strength JCE jurisdiction 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.

      Note: 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).

      Note: 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 configuration file **`java.security`** (located in the security directory for your local Java Virtual Machine (JVM) installation).
   The **`java.security`** file contains list of providers in preference order that is used by the Java Virtual Machine to decide from which provider to request a mechanism instance. Ensure that the nCipherKM provider is registered in the first position in this list, as shown in the following example:

```
#
# List of providers and their preference orders (see above):
#
security.provider.1=com.ncipher.provider.km.nCipherKM
security.provider.2=sun.security.provider.Sun
security.provider.3=sun.security.rsa.SunRsaSign
security.provider.4=com.sun.net.ssl.internal.ssl.Provider
security.provider.5=com.sun.crypto.provider.SunJCE
security.provider.6=sun.security.jgss.SunProvider
security.provider.7=com.sun.security.sasl.Provider
```

Placing the nCipherKM provider first in the list permits the nCipherKM provider's algorithms to override the algorithms that would be implemented by any other providers (except in cases where you explicitly request another provider name).

Note: The nCipherKM provider cannot serve requests required for the SSL classes unless it is in the first position in the list of providers.

Do not change the relative order of the other providers in the list.

Note: If you add the nCipherKM provider as **`security.provider.1`**, ensure that the subsequent providers are re-numbered correctly. 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).

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* on page 57.

   Note: 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.

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:

```
java com.ncipher.provider.InstallationTest
```

Note: 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* on page 125.

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* on page 125. For more information about hardserver configuration file settings, see *server_startup* on page 252.
- 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* on page 57.
- 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* on page 125.

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* on page 125.

keytool

You can use either the Oracle **keytool** utility or the IBM **keytool** utility to read and edit an nShield KeyStore. These utilities are shipped with the Oracle and IBM JVMs. You must specify the correct **nCipher.sworld** KeyStore type when you run the **keytool** utility, and you must specify the correct package name for the Oracle or IBM **keytool** utility.

To generate a new key in an OCS-protected KeyStore with the Oracle or IBM **keytool** utility, run the appropriate command:

- Sun Microsystems **keytool** utility:

  For Java 8, use the following command:

  ```
  java sun.security.tools.keytool.Main -genkey -storetype nCipher.sworld -keyalg RSA -
  sigalg SHA1withRSA -storepass KeyStore_passphrase -keystore KeyStore_path
  ```

  For Java 7 or earlier, use the following command:

  ```
  java sun.security.tools.KeyTool -genkey -storetype nCipher.sworld -keyalg RSA -sigalg
  SHA1withRSA -storepass KeyStore_passphrase -keystore KeyStore_path
  ```

- IBM **keytool** utility:

  ```
  java com.ibm.crypto.tools.KeyTool -genkey -storetype nCipher.sworld -keyalg RSA -sigalg
  SHA1withRSA -storepass KeyStore_passphrase -keystore KeyStore_path
  ```

In these example commands, *KeyStore_passphrase* is the pass phrase for the OCS that protects the KeyStore and *KeyStore_path* is the path to that KeyStore.

To generate a new key in a module-protected KeyStore with the Oracle or IBM **keytool** utility, run the appropriate command:

- Sun Microsystems **keytool** utility:

  For Java 8, use the following command:

  ```
  java -Dprotect=module -DignorePassphrase=true sun.security.tools.keytool.Main -genkey -
  storetype nCipher.sworld -keyalg RSA -sigalg SHA1withRSA -keystore KeyStore_path
  ```

  For Java 7 or earlier, use the following command:

  ```
  java -Dprotect=module -DignorePassphrase=true sun.security.tools.KeyTool -genkey -
  storetype nCipher.sworld -keyalg RSA -sigalg SHA1withRSA -keystore KeyStore_path
  ```

- IBM **keytool** utility:

```
java -Dprotect=module -DignorePassphrase=true com.ibm.crypto.tools.KeyTool -genkey -
storetype nCipher.sworld -keyalg RSA -sigalg SHA1withRSA -keystore KeyStore_path
```

In these example commands, *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.

Using keys

Only the nCipherKM provider can use keys stored in an nShield KeyStore because the underlying 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 140-2 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.

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 -Dproperty=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* on page 132.
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* on page 132. 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:***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.
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 pass phrase 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 pass phrase be provided; setting this property allows a dummy pass phrase to be used.

Property	Function for different values
`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.

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.
`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.

JCECSP_DEBUG value	Function
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**.

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 140-2 at either level 2 or level 3.

Note: In a Security World that complies with FIPS 140-2 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.

Note: 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 pass phrase.

Note: 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 nCipher 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 pass phrase used with the KeyStore must be the pass phrase of the card from the OCS that protects the keys in the KeyStore.

nCipher PKCS #11 library

To use the nCipher 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 nCipher PKCS #11 library with specific applications are available from the nCipher Web site: https://www.ncipher.com. Alternatively, contact Support.

Depending on the application, you may need to set the path and library name *%NFAST_ HOME%* `\toolkits\pkcs11\cknfast.dll` in a dialog or configuration file.

The nCipher PKCS #11 library has security options which you must configure before you use the PKCS #11 library. For more information, see *nCipher PKCS #11 library* on page 134.

From version 1.7, the nCipher PKCS #11 library can be used with FIPS 140-2 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 contains at least as many cards as you have HSMs. All the cards on the OCS must have the same pass phrase.

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.

Note: If you are using the `preload` command-line utility in conjunction with the nCipher PKCS #11 library, you can create *K/N* OCSs.

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.

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.

Digital signatures

The nCipher 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)

A nCipher 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.

Asymmetric encryption

The nCipher 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.

Symmetric encryption

The nCipher 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.

Message digest

The nCipher 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.

Mechanisms

The mechanisms currently supported by the nCipher PKCS #11 library, including some vendor-supplied mechanisms, are listed in the *Cryptographic API Integration Guide*.

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 nCipher 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* on page 142).

Note: 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** #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_OVERRIDE_SECURITY_ASSURANCES* on page 142.

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.

Using the nCipher PKCS #11 library

After you have loaded the nCipher 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* on page 141. 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* on page 141.

nCipher PKCS #11 library with load-sharing mode

If load-sharing mode is enabled, the nCipher 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* on page 140).

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* on page 141). 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 nCipher 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 nCipher 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 pass phrase for the OCS. The exact mechanism for supplying the pass phrase 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 nCipher 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.

nCipher PKCS #11 library with HSM Pool mode

If HSM Pool mode is enabled, the nCipher 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 140-2 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.

nCipher PKCS #11 library without load-sharing

There will be two entries for each HSM, unless you have set **CKNFAST_NO_ACCELERATOR_SLOTS**.

Note: 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 nCipher PKCS #11 library logs out any previously logged-in token from the slot and closes any open sessions.

nCipher 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 pass phrase for the tokens when using **preload** to load the card set. However, because the application is not aware that the card set has been preloaded, the application operates normally when handling the login activity (including prompting for a pass phrase), but the PKCS #11 library will not actually check the supplied pass phrase.

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 *nCipher PKCS #11 library environment variables* on page 138). 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* on page 215).

A logical token preloaded by **preload** for use with the nCipher 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.

nCipher PKCS #11 library environment variables

The nCipher 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_FAKE_ACCELERATOR_LOGIN**
- **CKNFAST_HSM_POOL**
- **CKNFAST_LOADSHARING**
- **CKNFAST_NO_ACCELERATOR_SLOTS**
- **CKNFAST_NO_SYMMETRIC**
- **CKNFAST_NO_UNWRAP**
- **CKNFAST_NONREMOVABLE**
- **CKNFAST_NVRAM_KEY_STORAGE**
- **CKNFAST_OVERRIDE_SECURITY_ASSURANCES**
- **CKNFAST_SEED_MAC_ZERO**
- **CKNFAST_SESSION_THREADSAFE**
- **CKNFAST_TOKENS_PERSISTENT**
- **CKNFAST_USE_THREAD_UPCALLS**
- **CKNFAST_LOAD_KEYS**
- **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**. If the **NFAST_HOME** environment variable is not set, or if environment variables are cleared by your application, the file **cknfastrc** must be in the *%NFAST_HOME%* directory of the client.

Note: 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
```

Note: 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*.

Note: For more information concerning Security World Software environment variables that are not specific to PKCS #11 and which are used to configure the behavior of your nShield installation, see the Security World Software installation instructions.

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 modification on disc; if the key file has been modified, then the key is reloaded. Calling **C_SetAttributeValues** 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.

CKNFAST_ASSURANCE_LOG

This variable is used to direct all warnings from the Security Assurance Mechanism to a specific log file.

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 **CKNFAST_CARDSET_HASH** is the SHA-1 hash of the secret on the card. Use the **nfkminfo** command-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* on page 228.

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.

Note: The default will be the old SP800-56a to maintain backward compatibility.

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 nCipher PKCS #11 library
7	API function calls
8	API return values

Value	Description
`9`	API function argument values
`10`	Details
`11`	Mutex locking detail

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.

CKNFAST_DEBUGFILE

You can use this variable to write the output for `CKNFAST_DEBUG` (`Path name > file name`).

CKNFAST_FAKE_ACCELERATOR_LOGIN

If this variable is set, the nCipher 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 arbitrary PIN and obtain the certificate.

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 application, 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 accelerator slot.

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.

Note: 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).

CKNFAST_NO_ACCELERATOR_SLOTS

If this variable is set, the nCipher 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.

Note: Setting this environment variable has no effect on **ckcheckinst** because **ckcheckinst** needs to list accelerator slots.

CKNFAST_NO_SYMMETRIC

If this variable is set, the nCipher PKCS #11 library does not advertise any symmetric key operations.

CKNFAST_NO_UNWRAP

If this variable is set, the nCipher 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.

CKNFAST_NONREMOVABLE

When this environment variable is set, the state changes of the inserted card set are ignored by the nCipher PKCS #11 library.

Note: 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.

CKNFAST_NVRAM_KEY_STORAGE

When this environment variable is set, the PKCS #11 library generates only keys in nonvolatile memory (NVRAM). You must also ensure this environment variable is set in order to delete NVRAM-stored keys.

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**
- **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:

```
set CKNFAST_OVERRIDE_SECURITY_ASSURANCES=token1;token2=value3
```

In the configuration file, enter the following to set the variable:

```
CKNFAST_OVERRIDE_SECURITY_ASSURANCES=token1;token2=value3
```

Unknown parameters generate a warning; see *Diagnostic warnings about questionable operations* on page 146.

The meaning of these parameters is described in the rest of this section.

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.

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.

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.

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.

Note: 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 functions 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_PERMITTED**, 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_EncryptInit**, and **C_DecryptInit** check the **CKA_CREATION_DATE** against the current time.

explicitness

The **explicitness** parameter permits applications to create insecure keys without explicitly recognizing that they are insecure by setting the flag which allows export as plain text. An insecure key is one whose plain text is available to an attacker on the host; thus it makes no sense to restrict legitimate users' access to the plain text of the key value.

Note: 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. However, when the **explicitness** parameter is set, these operations are allowed.

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.

Note: 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 environment.

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.

unwrap_mech

The **unwrap_mech** parameter allows keys transferred into the HSM in an insecurely encrypted form to be treated as if the encryption had been secure. This parameter allows you to use key-decryption keys for insecure decryption mechanisms as well as for raw decryption.

There are no key decryption or wrapping mechanisms that are both secure and suitable for long keys. Set the **unwrap_mech** parameter to use PKCS #11 **unwrap** to create keys that are treated as secure. Set the **unwrap_mech** parameter at the time that the wrapping key is created or imported.

When the **unwrap_mech** parameter is set, the PKCS #11 API adds the **CKA_DECRYPT** permission on decryption, even if the template has **CKA_DECRYPT=false**. By default, trying to create a key with **CKA_UNWRAP=true** and **CKA_DECRYPT=false** fails with **CKR_TEMPLATE_INCONSISTENT**. If **unwrap_mech** is supplied as a parameter for **CKNFAST_OVERRIDE_SECURITY_ASSURANCES**, then when the **CKA_UNWRAP** permission is requested on a key, the library automatically adds the **CKA_DECRYPT** permission, even if the template has **CKA_DECRYPT** false, because abuse of the decryption mechanisms would allow a program to use the library to decrypt with the key.

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 considered 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).

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).

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).

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.

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.

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 minimum 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.

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.

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**. On Windows platforms, they are also always sent to the Event Viewer. 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.

Note: 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**.

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.

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.

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.

Note: Use of the nCipher PKCS #11 library to create tokens is deprecated, because it can only create 1/1 tokens in FIPS 140-2 level 2 security worlds. Use KeySafe or one of the command-line utilities to create OCSs.

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.

CKNFAST_LOAD_KEYS

This variable will load private objects at **C_Login** time, rather than at the first cryptographic operation.

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.

Note: This environment variable does not prevent you from deleting an object from your token.

Checking the installation of the nCipher PKCS #11 library

After you have created a security world, ensure that the nCipher PKCS #11 library has been successfully installed by using the **ckcheckinst** command-line utility.

To verify the installation of the nCipher 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 initialization 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 2.01
                           flags 0
                  manufacturerID "nCipher Corp. Ltd                "
              libraryDescription "nFast PKCS#11 1.#.#              "
          implementation version 1.##
                  Strictfips 140 enabled
        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 2.01** 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.

> **Note:** If you are using the **preload** command-line utility in conjunction with the nCipher 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 140-2 level 3 compliant Security World, **ckcheckinst** prompts you to enter the pass phrase for the selected Operator Card.
3. Type the pass phrase, 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 nCipher library.

How the nCipher 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
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 nCipher 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.

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.

Note: 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.

CodeSafe applications

If you have enabled the Secure Execution Engine (SEE), your system can run CodeSafe applications that implement special functionality.

Note: If you wish to use the SEE to run applications, it must have been ordered and enabled as described in *Enabling optional features on the module* on page 46.

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.

Note: To use **see-sock-serv** directly, you must select **BSDlib sockserv** .

Microsoft CAPI CSP

We provide a Cryptographic Service Provider (CSP) that implements the Cryptographic API (CAPI) supported in Windows 2003 and later.

Installing the CAPI CSP

A shortcut to the CSP installation wizard is placed in the Start menu under **All Programs** > **nCipher** when installing the Security World Software. If you want to use 32-bit applications with the nShield CAPI provider run the 32-bit installation wizard to install the CAPI CSP, and if you want to use 64-bit applications with the nShield CAPI provider run the 64-bit CSP installation wizard to install the CAPI CSP.

You can also use the CSP installation wizard to create new Security Worlds, see *Creating a Security World using the CSP or CNG wizard* on page 71, load existing Security Worlds, see *Adding an HSM to a Security World with the CSP or CNG wizard* on page 78, generate new Operator Card Sets, see *Creating an Operator Card Set with the CSP or CNG wizard* on page 98, and configure the set-up parameters of the CAPI CSP including HSM Pool mode.

With module firmware version 2.65.2 or later, if your application only uses module protected keys, you can use HSM Pool mode with multiple hardware security modules. HSM Pool mode exposes a single pool of HSMs and supports returning or adding a hardware security module to the pool without restarting the system. With a FIPS 140-2 level 3 Security World, keys cannot be created in HSM Pool mode, however keys created outside HSM Pool mode can be used in HSM Pool mode.

Note: The CSP installation wizard is not suitable for creating complex Security World setups. When creating such Security Worlds, or if you require more flexibility than the CSP installation wizard provides, we recommend following the instructions in *Creating a Security World using new-world* on page 63.

Use the standard Security World utility **nfkmverify** to check the security of all stored keys in the Security World; **nfkminfo**, **nfkmcheck** and other standard utilities can also be used to assist in this process.

The CSP installation wizard registers the CAPI CSP as a key provider on your system.

Importing a key

Use the **cspimport** utility to move keys between containers or to import a pre-generated NFKM key into a container. For more information about using the **cspimport** utility, run **cspimport** specifying either the **--help** or **--usage** options.

Supported algorithms

The nShield CSPs support a similar range of algorithms to the Microsoft CSP.

Symmetric algorithms

- **CALG_DES**
- **CALG_3DES_112** (double-DES)
- **CALG_3DES**
- **CALG_RC4**
- **CALG_AES_128**
- **CALG_AES_192**
- **CALG_AES_256**

Asymmetric algorithms

- **CALC_RSA_SIGN** (only Enhanced RSA and AES Cryptographic Provider)
- **CALC_RSA_KEYX** (only Enhanced RSA and AES Cryptographic Provider)
- **CALC_DSA_SIGN** (only Enhanced DSS and Diffie-Hellman Cryptographic Provider and DSS Signature Cryptographic Provider)
- **CALC_DSS_SIGN** (only Enhanced DSS and Diffie-Hellman Cryptographic Provider)
- **CALC_DH_KEYX** (only Enhanced DSS and Diffie-Hellman Cryptographic Provider)
- **CALC_DH_SF** (only Enhanced DSS and Diffie-Hellman Cryptographic Provider)
- **CALC_DH_EPHEM** (only Enhanced DSS and Diffie-Hellman Cryptographic Provider)

Hash algorithms

- **CALG_SHA1**
- **CALG_SHA256**
- **CALG_SHA384**
- **CALG_SHA512**
- **CALG_SSL3_SHAMD5**
- **CALG_MD5**
- **CALG_MAC**
- **CALG_HMAC**

In addition, the Enhanced SChannel Cryptographic Provider and the Enhanced DSS and Diffie-Hellman SChannel Cryptographic Provider support all the internal algorithm types necessary for SSL3 and TLS1 support.

The nShield CSPs do not support SSL2.

Container storage format

Note: Versions of the CSP later than 1.11.0 have an updated container storage mechanism. CSP containers are now stored as part of the Security World instead of in the Windows registry file.

Versions of the CSP later than 1.11.0 use a non-backwards-compatible container and key storage format. If you are installing version 1.11.0 or later of the CSP over older versions, you must run the **cspmigrate** utility in order to convert containers and keys from the old system to the new system.

CSP versions 1.11.0 and later have a number of advantages over older versions:

- The CSP state is easily mirrored between multiple machines simply by copying the contents of the Key Management Data directory or by sharing the Key Management Data directory across a network.
- The CSP key files can have arbitrary names (previously, the names of key files were linked to their key type and their container name). This new method facilitates the importation of existing Security World keys into the CSP.
- Every different container is now guaranteed to have a distinct storage location. There were circumstances in CSP versions older than 1.11.0 in which two containers with similar names could have shared the same keys wrongly.

However, there are some points to bear in mind concerning CSP versions 1.11.10 and later:

- If you want to share the same key between multiple computers, we supply the **cspimport** utility for transferring keys between containers.
- Any existing containers with older versions of the CSP must be migrated to the new format. We provide a utility, **cspmigrate**, to migrate containers from the old to the new system.

Utilities for the CAPI CSP

To help you migrate from Windows registry-based CSP container storage to the new CSP format, CSP version 1.11.0 and later provides you with a set of utilities. The new CSP format stores all information about a Security World in the Key Management Data directory. There are also utilities to manage the interfaces between the MSCAPI library and the module.

These utilities are:

Utility	Description
cspcheck	This utility checks that CSP container files are intact and uncorrupted, and also that referenced key files exist. Use **cspcheck** in conjunction with **nfkmcheck**, but run **nfkmcheck** first in order to test the integrity of your Security World files.
cspimport	This utility allows you to insert keys manually into existing CSP containers. This utility has two modes that either allow you to change a container's key association to that of an arbitrary Security World key or to copy CSP keys between containers.

Utility	Description
`cspmigrate`	This utility moves the CSP container information from the registry into the Security World. If a new container already exists and has a key in it, and an identically-named old container exists with the same key, the utility asks you which key to keep. You can either: Enter `-q` to keep the new keys. Enter `-f` to overwrite new keys with old keys.
`cspnvfix`	Regenerate the NVRAM key counter area for a specified nShield CSP key.
`csptest`	Test the installed Cryptographic Service Providers.
`csputils`	This utility lists CSP containers and provides detailed information about them. It can also be used to delete container files if the current user has administrative privileges.
`configure-csp-poolmode`	The `--mscapi` option allows HSM Pool mode to be enabled or disabled for the nCipher CAPI CSP without using the CSP wizard.
`keytst`	This utility displays information about existing CSP key containers by using the Microsoft CryptoAPI. If you have the appropriate permissions, `keytst` also allows you to create containers and their keys, as well as delete containers.

Note: Each of these commands has an `-h` option that displays the usage message for the command.

Uninstalling the CAPI CSP

To uninstall the CAPI CSP and unregister it as a cryptographic provider on your system, run the `cngregister` and `cnginstall` commands with the `-u` option. For more information, see *Utilities for CNG* on page 164.

Microsoft Cryptography API: Next Generation (CNG)

Cryptography API: Next Generation (CNG) is the successor to the Microsoft Cryptographic API (CAPI) and its long-term replacement. CNG is designed to be extensible at many levels and cryptography agnostic in its behavior.

The Security World Software implementation of Microsoft CNG is supported on Microsoft Windows Windows Server 2008 (for both x86 and x64 architectures) and later releases, including Windows Server 2012. The nCipher CNG CSP provides the benefits of hardware-based encryption accessed through the standard Microsoft API, and supports the National Security Agency (NSA) classified Suite B algorithms.

Configuring the nCipher CNG CSP

The DLL files that support the nCipher CNG CSP are installed during product installation. However, you need to register the nCipher CNG CSP before you can use it.

You can unregister the nCipher CNG CSP without removing the provider DLL files from your system. After unregistering, you can reregister the nCipher CNG CSP at any time as long as the files have not been uninstalled from your system. For more information, see *Unregistering or reregistering the CNG CSP* on page 157.

You can completely uninstall the nCipher CNG CSP, removing the files from your system. After uninstalling, you must reinstall the files and then reregister the nCipher CNG CSP before you can use it. For more information, see *Unregistering or reregistering the CNG CSP* on page 157

Registering the CNG CSP

You can register the nCipher CNG CSP with:

- CNG Configuration Wizard
- The **cngregister** command-line utility

To register the nCipher CNG CSP, the hardserver must be running and able to communicate with at least one module. This requirement is normally fulfilled during the product installation process. You can check that this requirement is fulfilled by running the **enquiry** command-line utility and checking the output for details about the module. See the Installation Guide for more information.

Registering the CNG CSP with the CNG Configuration Wizard

We recommend using the CNG Configuration Wizard to register the nCipher CNG CSP. The product installation process places a shortcut to the CNG Configuration Wizard in the Start menu under **All Programs** > **nCipher**.

Note: You can also use the CNG Configuration Wizard to create new Security Worlds, see *Creating a Security World using the CSP or CNG wizard* on page 71, load existing Security Worlds, see *Adding an HSM to a Security World with the CSP or CNG wizard* on page 78, generate new OCSs, see *Creating an Operator Card Set with the CSP or CNG wizard* on page 98, and configure the set-up parameters of the CNG CSP including HSM Pool mode.

With module firmware version 2.65.2 or later, if your application only uses module protected keys, you can use HSM Pool mode with multiple hardware security modules. HSM Pool mode exposes a single pool of HSMs and supports returning or adding a hardware security module to the pool without restarting the system. With a FIPS 140-2 level 3 Security World, keys cannot be created in HSM Pool mode, however keys created outside HSM Pool mode can be used in HSM Pool mode.

To register the CNG CSP with the CNG Configuration Wizard, you must have already created a Security World and chosen a key protection method, either module-protection or OCS-protection. If you chose OCS-protection, you must also have already created an OCS before you can register the nCipher CNG CSP with the CNG Configuration Wizard.

Note: The CNG Configuration Wizard is not suitable for creating complex Security World setups. When creating such Security Worlds, or if you require more flexibility than the CNG configuration wizard provides, we recommend following the instructions in *Creating a Security World using new-world* on page 63.

If you use the CNG Configuration Wizard to create a Security World (and, if appropriate, an OCS), the wizard automatically prompts you to register the CNG CSP after you have fulfilled the necessary prerequisites.

You can also use the CNG Configuration Wizard to change an existing configuration at any time by running the wizard as usual and choosing the **Use the existing security world** option on the **Initial setup** screen.

To register the CNG CSP with the CNG Configuration Wizard after the necessary key-protection prerequisites have been fulfilled:

1. If the wizard is not already running:
   a. Run the wizard by double-clicking its shortcut in the Start menu under **All Programs** > **nCipher**.
      The wizard displays the welcome window.
   b. Click the **Next** button.
      The wizard allows you to configure HSM Pool mode for CNG.
   c. Click the **Next** button.
      If the prerequisite to create a Security World has been fulfilled, the wizard displays a confirmation screen.
   d. Click the **Next** button.
      The wizard displays a screen confirming that your Security World and (if you chose to create an OCS) an OCS have been created.
      Note: If you chose module-protection for your keys, the wizard does not confirm that an OCS has been created.
2. When the wizard has confirmed that it is ready to register the nCipher CNG providers, click the **Next** button.
   The wizard registers the nCipher CNG CSP.
   Note: You cannot use the CNG Configuration Wizard to configure the nCipher CNG providers for use as defaults. We recommend that you always use the nCipher CNG providers by selecting them directly with the application that is using CNG.
   When configuration of your nCipher CNG CSP is complete, the wizard displays a confirmation screen.

Registering the CNG CSP with cngregister

You can use the **cngregister** command-line utility to register the nCipher CNG CSP manually even if you have not already created a Security World (or, if you choose OCS-protection for your keys, even if you have not already created an OCS).

To register the nCipher CNG CSP with the **cngregister** command-line utility, run the command without specifying any options:

```
cngregister
```

Note: You cannot use the **cngregister** command-line utility to configure the nCipher CNG providers for use as defaults. we recommend that you always use the nCipher CNG providers by selecting them directly with the application that is using CNG.

For more information about the **cngregister** command-line utility, see *cngregister* on page 165.

Unregistering or reregistering the CNG CSP

You can use the **cngregister** command-line utility to unregister or reregister the nCipher CNG CSP manually.

To unregister the nCipher CNG CSP, run the command:

```
cngregister -U
```

This command unregisters the CNG CSP, but does not remove the provider DLL files from your system. For information about removing these files, see *Uninstalling or reinstalling the CNG CSP* on page 157.

If any applications or services are using the nCipher CNG providers for key storage or cryptography, unregistering the nCipher CNG CSP can cause system instability.

After unregistering the nCipher CNG CSP, you can reregister it at any time as long as the files have not been uninstalled from your system. To reregister the nCipher CNG CSP on your system, run the command:

```
cngregister
```

Note: You cannot use the **cngregister** command-line utility to configure the nCipher CNG providers for use as defaults. We recommend that you always use the nCipher CNG providers by selecting them directly with the application that is using CNG.

For more information about these command-line utilities, see *Utilities for CNG* on page 164.

Uninstalling or reinstalling the CNG CSP

To uninstall the nCipher CNG CSP:

1. To remove any and all dependencies that you have set, run the command:

   ```
   ncsvcdep -x
   ```

   Note: Always run **ncsvcdep** as a user with full administrative privileges.
2. Unregister the nCipher CNG CSP on your system by running the command:

   ```
   cngregister -U
   ```

   This command unregisters the CNG CSP, but does not remove the provider DLL files from your system.

3. Uninstall the nCipher CNG DLLs from your system:
   - On 32-bit versions of Windows, run the command:

     ```
     cnginstall -U
     ```

   - On 64-bit versions of Windows, run the command:

     ```
     cnginstall64 -U
     ```

To reinstall the nCipher CNG CSP after you have previously uninstalled it:

1. Reinstall the nCipher CNG CSP files on your system:
   - On 32-bit versions of Windows, run the command:

     ```
     cnginstall -i
     ```

   - On 64-bit versions of Windows, run the command:

     ```
     cnginstall64 -i
     ```

2. Reregister the nCipher CNG CSP on your system by running the command:

   ```
   cngregister
   ```

For more information about these command-line utilities, see *Utilities for CNG* on page 164.

Supported algorithms for CNG

This section lists the National Security Agency (NSA) classified Suite B algorithms supported by the nCipher CNG providers.

Note: The MQV algorithm is not supported by the nCipher CNG providers.

Signature interfaces (key signing)

Interface name	*Type of support*
RSA PKCS#1 v1	Hardware
RSA PSS	Hardware
DSA	Hardware

Interface name	Type of support
ECDSA_P224	Hardware
ECDSA_P256	Hardware
ECDSA_P384	Hardware
ECDSA_P521	Hardware

Note: Hashes used with ECDSA must be of the same length or shorter than the curve itself. If you attempt to use a hash longer than the curve the operation returns `NOT_SUPPORTED`. In FIPS 140-2 level 3 Security Worlds ECDSA signing is only supported where the length of the curve is approximately the length of the hash.

Hashes

Hash name	Type of support
SHA1	Hardware (HMAC only)/software
SHA256	Hardware (HMAC only)/software
SHA384	Hardware (HMAC only)/software
SHA512	Hardware (HMAC only)/software
SHA224	Hardware (HMAC only, requires firmware version 2.33.60 or later) /software
MD5	Hardware (HMAC only)/software

Note: MD5 is not supported in FIPS 140-2 mode.

Asymmetric encryption

Algorithm name	Type of support
RSA Raw (NCRYPT_NO_PADDING_ FLAG)	Hardware
RSA PKCS#1 v1 (NCRYPT_PAD_ PKCS1_FLAG)	Hardware
RSA OAEP (NCRYPT_PAD_OAEP_FLAG)	Hardware

Symmetric encryption

Algorithm name	Type of support
RC4	Hardware and Software (not supported in FIPS 140-2 level 3 mode)
AES ECB,CBC	Hardware and Software

Algorithm name	Type of support
DES ECB,CBC	Hardware and Software (DES is not supported in FIPS 140-2 level 3 mode)
3DES ECB,CBC	Hardware and Software
3DES_112 ECB,CBC	Hardware and Software

Key exchange

Protocol name	Type of support
DH	Hardware
ECDH_P224	Hardware
ECDH_P256	Hardware
ECDH_P348	Hardware
ECDH_P521	Hardware

Note: Elliptic curve cryptography algorithms must be enabled before use. Use the **`fet`** command-line utility with an appropriate certificate to enable a purchased feature. If you enable the elliptic curve feature on your modules after you first register the CNG providers, you must run the configuration wizard again for the elliptic curve algorithm providers to be registered.

Random Number Generation

Name	Type of support
RNG	Hardware

Migrating keys for CNG

We provide functionality for migrating existing keys from other providers into the Security World Key Storage Provider. To identify installed providers, run the command:

```
cnglist --list-providers
```

To identify the keys that are available from a particular provider, run the command:

```
cnglist --list-keys --provider="ProviderName"
```

In this command, *ProviderName* is the name of the provider.

The following command provides an example of identifying keys from the *Security World Key Storage Provider*:

```
cnglist --list-keys --provider="nCipher Security World Key Storage Provider"
MyApp Personal Data Key: RSA
CertReq-5eb45f6d-6798-472f-b668-288bc5d961da: ECDSA_P256 machine
WebServer Signing Key: DSA machine
ADCS-Root-Key: ECDSA_P521 machine
```

Note: To list the keys available from the *Security World Key Storage Provider*, run the command **`cnglist --list-keys`** (without specifying the **`--provider`** option) .

Importing a Microsoft CAPI key into the Security World Key Storage Provider

To import a Microsoft CAPI key into the Security World Key Storage Provider, first run the CAPI utility **`csputils`** to identify the existing CAPI containers and their key contents.

CAPI containers can contain either a signing key or a key exchange key, or both. The following example shows how to import both a signing key and a key exchange key from a Microsoft CAPI container:

```
cngimport -m --csp="Microsoft Strong Cryptographic Provider"
  -k "EXAMPLE_CAPICONTAINER"
     "EXAMPLE_IMPORTED_SIGNATURE_CAPICONTAINER"
     "EXAMPLE_IMPORTED_KEYEXCHANGE_CAPICONTAINER"
```

To check the success of the import, list the keys present in the *Security World Key Storage Provider*:

```
cnglist --list-keys
EXAMPLE_IMPORTED_SIGNATURE_CAPICONTAINER: RSA
EXAMPLE_IMPORTED_KEYEXCHANGE_CAPICONTAINER: DH
```

The following example command shows how to import a single signing key:

```
cngimport -m -s --csp="Microsoft Strong Cryptographic Provider"
  --key="EXAMPLE_CAPICONTAINER"
        "EXAMPLE_IMPORTED_SIGNATURE_ONLY_CAPICONTAINER"
```

Run the **`cnglist`** command with the **`--list-keys`** option to check the success of the key import:

```
cnglist --list-keys
EXAMPLE_IMPORTED_SIGNATURE_ONLY_CAPICONTAINER: RSA
```

Note: The **`cngimport`** option **`-m/--migrate`** cannot be used to migrate nCipher CAPI container keys to CNG. For information about importing nCipher CAPI container keys into CNG, see *Importing a Security World key into the Security World Key Storage Provider* on page 162.

Importing a Microsoft CNG key into the Security World Key Storage Provider

To import a Microsoft CNG key into the Security World Key Storage Provider, run the **cngimport** command as shown in the following example:

```
cngimport -m
  -k "EXAMPLE_RSA_1024"
     "IMPORTED_RSA_1024"
```

Run the **cnglist** command with the **--list-keys** option to check the success of the key import:

```
cnglist --list-keys
  IMPORTED_RSA_1024: RSA
```

The original key is not deleted from the provider from which it was imported:

```
cnglist --list-keys --provider="Microsoft Software Key Storage Provider"
  EXAMPLE_RSA_1024
```

Note: Certain applications, such as Certificate Services, create keys using the Microsoft Software Key Storage Provider which cannot be exported. Attempting to import such a key into the nCipher provider results in the following message:

```
cngimport -m -k WIN-KQ1Z6JMCUTB-CA WIN-ncipher-CA
Unable to continue. This key can not be exported from Microsoft Software Key Storage
Provider.
```

Importing a Security World key into the Security World Key Storage Provider

To import a Security World key into the Security World Key Storage Provider, run the **cngimport** utility as shown in the following example:

```
cngimport --import --key=nfkmsimple1 --appname=simple nfkmsimple1
Found key 'nfkmsimple1'
Importing NFKM key.. done
```

Run **cnglist** with the **--list-keys** option to confirm that the key has been successfully imported:

```
cnglist --list-keys
nfkmsimple1: RSA
```

To import an nCipher CAPI container into the Security World Key Storage Provider, run the **csputils** command to identify the container name:

```
csputils -l
File ID    Container name       Container owner      DLL name   S X
=========  ===================  ===================  =========  = =
31e994f07  CONTAINER2           SYWELL\Administrato  ncsp       * *
3a2b082a8  CAPICONTAINER        SYWELL\Administrato  ncsp       * *
2 containers and 4 keys found.
```

Note: Run the **csputils** command with the **-l** and **-m** options to migrate an nCipher CAPI machine container.

Identify the Security World key names of the keys in the container by running the **csputils** command as follows:

```
csputils -d -n CAPICONTAINER
Detailed report for container ID #3a2b082a8f2ee1a5acb756d5e95b09817072807a
Filename:       key_mscapi_container-3a2b082a8f2ee1a5acb756d5e95b09817072807a
Container name: CAPICONTAINER
User name:      SYWELL\Administrator
User SID:       s-1-5-21-352906761-2625708315-3490211485-500
CSP DLL name:   ncsp.dll
Filename for signature key is key_mscapi_ce51a0ee0ea164b993d1edcbf639f2be62c53222
   Key was generated by the CSP
   Key hash:    ce51a0ee0ea164b993d1edcbf639f2be62c53222
   Key is recoverable.
   Key is cardset protected.
      Cardset name:             nopin
      Sharing parameters:       1 of 1 shares required.
      Cardset hash:             d45b30e7b60cb226f5ade5b54f536bc1cc465fa4
      Cardset is non-persistent.
Filename for key exchange key is key_mscapi_dbd84e8155e144c59cf8797d16e7f8bd19ac446a
   Key was generated by the CSP
   Key hash:    dbd84e8155e144c59cf8797d16e7f8bd19ac446a
   Key is recoverable.
   Key is cardset protected.
      Cardset name:             nopin
      Sharing parameters:       1 of 1 shares required.
      Cardset hash:             d45b30e7b60cb226f5ade5b54f536bc1cc465fa4
      Cardset is non-persistent.
1 container and 2 keys found.
```

The key name to pass to the **cngimport** command **--key** option is the part of the key name that follows **key_mscapi_** in the output line that starts **Filename for signature key is key_mscapi_**.

For example, the signature key file name for **CAPICONTAINER** in the example shown above is **key_mscapi_ce51a0ee0ea164b993d1edcbf639f2be62c53222**, so **ce51a0ee0ea164b993d1edcbf639f2be62c53222** is the key name that should be passed to **cngimport**:

```
cngimport --import --key="ce51a0ee0ea164b993d1edcbf639f2be62c53222" --
appname="mscapi" Signature_Key_Imported_From_nCipher_CAPI
Found unnamed key
Importing NFKM key.. done
```

Run **cnglist** with the **--list-keys** option to confirm that the key has been successfully imported:

```
cnglist --list-keys
Signature_Key_Imported_From_nCipher_CAPI: RSA
cngsoak: ECDH_P256
```

Follow the same procedure for importing the key exchange key from the nCipher CAPI container.

Using CAPI keys in CNG

We now provide the capability to use keys generated by CAPI in CNG applications. This is provided through the standard **NCryptOpenKey** CNG API call. Passing either **AT_SIGNATURE** or **AT_KEYEXCHANGE** as the **dwLegacyKeySpec** parameter and the CAPI container name as the **pszKeyName** parameter will invoke this mode of operation. The CAPI key will be loaded into the CNG provider and will behave as if it was a CNG key. Any key authorization required will be handled with a user interface being invoked to prompt the application user to insert the smart card or enter appropriate pass phrases. There is support for Key Usage and Key Counting properties properties.

The CNG application has to be written such that it calls **NCryptOpenKey** to open a CAPI key explicitly.

Utilities for CNG

Use the **nfkmverify** command-line utility to check the security of all stored keys in the Security World. Use **nfkminfo**, **nfkmcheck**, and other command-line utilities to assist in this process.

The following table lists the utilities specific to the nCipher CNG CSP:

x86	x64	Utility description
cngimport.exe	**cngimport.exe**	This key migration utility is used to migrate Security World, CAPI, and CNG keys to the Security World Key Storage Provider.
cnginstall.exe	**cnginstall64.exe**	This utility is the nCipher CNG CSP installer. Only use this utility to remove or reinstall the provider DLLs and associated registry entries manually.
cnglist.exe	**cnglist.exe**	This utility lists information about CNG CSP.
cngregister.exe	**cngregister.exe**	This is the nCipher CNG CSP registration utility. You can use it to unregister and re-register the nCipher providers manually.
cngsoak.exe	**cngsoak64.exe**	This utility is the nCipher CNG soak tool. You can use it to evaluate the performance of signing, key exchange, and key generation using a user-defined number of threads.
ncsvcdep.exe	**ncsvcdep.exe**	This utility is the service dependency tool. You can configure some service based applications, such as Microsoft Certificate Services and IIS, to use the nCipher CNG CSP. The nCipher Service dependency tool allows you to add the nFast Server to the dependency list of such services.
configure-csp-poolmode	**configure-csp-poolmode64**	This utility allows HSM Pool mode to be enabled or disabled for the nCipher CNG CSP without using the CNG wizard.

These utilities are located in the **bin** directory of your Security World Software installation (for example, *%NFAST_HOME%***bin**).

Note: On 64-bit versions of Windows, both the 32-bit and 64-bit versions of the listed utilities are installed. When working on an 64-bit version of Windows, always ensure that you use the 64-bit version of the utility (if one is available).

cngimport

Use **cngimport** to migrate keys to the Security World Key Storage Provider. For more information, see *Migrating keys for CNG* on page 160.

cnginstall

The **cnginstall** utility is used by the Security World Software installation wizard. You can also use this utility to manually uninstall (or reinstall) the nCipher CNG DLLs and registry entries.

To uninstall the nCipher CNG DLL files, run the command:

```
cnginstall -U
```

This command removes the provider DLL files from your system. It produces output of the form:

```
nckspptdll removed.
nckspswdll removed.
ncppdll removed.
```

Before you uninstall the nCipher CNG DLL files, ensure that you unregister the nCipher CNG CSP. For more information, see:

- *cngregister* on page 165
- *Unregistering or reregistering the CNG CSP* on page 157.

After unregistering the nCipher CNG CSP, you can reregister it at any time as long as the files have not been uninstalled from your system. To reregister the nCipher CNG CSP on your system, run the command:

```
cngregister
```

For more information about uninstalling and reinstalling the nCipher CNG CSP with **cnginstall**, see *Uninstalling or reinstalling the CNG CSP* on page 157.

cngregister

Use **cngregister** to unregister the nCipher CNG CSP manually.

To unregister the nCipher CNG CSP, run the command:

```
cngregister -U
```

This command produces output for the form:

```
Unregistered provider 'nCipher Primitive Provider'
Unregistered provider 'nCipher Security World Key Storage Provider'
```

This command unregisters the CNG CSP, but does not remove the provider DLL files from your system. For information about removing these files, see:

- *cnginstall* on page 165
- *Uninstalling or reinstalling the CNG CSP* on page 157.

If any applications or services are using the nCipher CNG CSP for key storage or cryptography, unregistering it can cause system instability.

After unregistering the nCipher CNG CSP, you can reregister it at any time as long as the files have not been uninstalled from your system. To reregister the nCipher CNG CSP on your system, run the command:

```
cngregister
```

Note: You cannot use the **cngregister** command-line utility to configure the nCipher CNG providers for use as defaults. We recommend that you always use the nCipher CNG providers by selecting them directly with the application that is using CNG.

cngsoak

Use **cngsoak** to obtain statistics about the performance of the nCipher CNG CSP. Specifically, use **cngsoak** to determine the speed of:

- Signing a hash (**cngsoak --sign**)
- encryption (**cngsoak --encrypt**)
- key exchange (**cngsoak --keyx**)
- key generation (**cngsoak --generate**).

The output from **cngsoak** displays information as columns of information. From left to right, these columns display:

- The time in second that **cngsoak** has been running
- the total number of operations completed
- the number of operations completed in last second
- the average number of operations completed each second.

ncsvcdep

Use the **ncsvcdep** utility to ensure that the nShield **nFast Server** service is running before certain services are enabled. For example, Active Directory Certificate Services or Internet Information Services require that the hardserver is running in order to use the nCipher CNG CSP. Failure to set this dependency can lead to system instability.

To list installed services, run the **ncsvcdep** command with the **-l** option:

```
ncsvcdep -l
```

Output from this command has the form:

```
Installed Services (Count - "Display Name" - "Service Name")
0 - "Application Experience" - "AeLookupSvc"
1 - "Application Layer Gateway Service" - "ALG"
2 - "Application Information" - "Appinfo"
3 - "Application Management" - "AppMgmt"
4 - "Windows Audio Endpoint Builder" - "AudioEndpointBuilder"
.
.
108 - "nFast Server" - "nFast Server"
109 - "Active Directory Certificate Services" - "CertSvc"
```

Note: Always run **ncsvcdep** as a user with full administrative privileges.

To set a dependency, run the command:

```
ncsvcdep -a "DependentService"
```

In this command, *DependentService* is the service that has the dependency. The following example shows how to make the Active Directory Certificate Services dependent on the nFast Server:

```
ncsvcdep -a "CertSvc"
Dependency change succeeded.
```

To remove a specific dependency relationship, run **ncsvcdep** with the **-r** option, for example:

```
ncsvcdep -r "CertSvc"
Dependency change succeeded.
```

To remove all dependencies, run **ncsvcdep** with the **-x** option:

```
ncsvcdep -x
```

Note: Microsoft Certificate Services require that the **certsvc** service is made dependent on the hardserver.

Note: Microsoft Internet Information Services require that the **http** service is made dependent on the hardserver.

cnglist

Use **cnglist** to display details of CNG providers, keys, and algorithms.

To list details of the CNG providers, run the **cnglist** command with the **--list-providers** option:

```
cnglist --list-providers
```

Output from this command is of the form:

```
Microsoft Primitive Provider
Microsoft Smart Card Key Storage Provider
Microsoft Software Key Storage Provider
Microsoft SSL Protocol Provider
nCipher Primitive Provider
nCipher Security World Key Storage Provider
```

To list details of the algorithms, run the **cnglist** command with the **--list-algorithms** option:

```
cnglist --list-algorithms
```

Output from this command has the form:

```
BCryptEnumAlgorithms(BCRYPT_CIPHER_OPERATION):
   Name                           Class      Flags
   AES                            0x00000001 0x0
   RC4                            0x00000001 0x0
   DES                            0x00000001 0x0
   DESX                           0x00000001 0x0
   3DES                           0x00000001 0x0
   3DES_112                       0x00000001 0x0
BCryptEnumAlgorithms(BCRYPT_HASH_OPERATION):
   Name                           Class      Flags
   SHA1                           0x00000002 0x0
   MD2                            0x00000002 0x0
   MD4                            0x00000002 0x0
   MD5                            0x00000002 0x0
   SHA256                         0x00000002 0x0
   SHA384                         0x00000002 0x0
   SHA512                         0x00000002 0x0
   AES-GMAC                       0x00000002 0x0
   SHA224                         0x00000002 0x0
BCryptEnumAlgorithms(BCRYPT_ASYMMETRIC_ENCRYPTION_OPERATION):
   Name                           Class      Flags
   RSA                            0x00000003 0x0
```

To list details of the algorithms for the Security World Key Storage Provider, run the **cnglist** command with the **--list-algorithms**, **--keystorage**, and **--nc** options:

```
cnglist --list-algorithms --keystorage --nc
```

Output from this command has the form:

```
NCryptEnumAlgorithms(NCRYPT_CIPHER_OPERATION) no supported algorithms
NCryptEnumAlgorithms(NCRYPT_HASH_OPERATION) no supported algorithms
NCryptEnumAlgorithms(NCRYPT_ASYMMETRIC_ENCRYPTION_OPERATION):
   Name                          Class      Operations Flags
   RSA                           0x00000003 0x00000014 0x0
NCryptEnumAlgorithms(NCRYPT_SECRET_AGREEMENT_OPERATION):
   Name                          Class      Operations Flags
   DH                            0x00000004 0x00000008 0x0
   ECDH_P224                     0x00000004 0x00000008 0x0
   ECDH_P256                     0x00000004 0x00000008 0x0
   ECDH_P384                     0x00000004 0x00000008 0x0
   ECDH_P521                     0x00000004 0x00000008 0x0
NCryptEnumAlgorithms(NCRYPT_SIGNATURE_OPERATION):
   Name                          Class      Operations Flags
   RSA                           0x00000003 0x00000014 0x0
   DSA                           0x00000005 0x00000010 0x0
   ECDSA_P224                    0x00000005 0x00000010 0x0
   ECDSA_P256                    0x00000005 0x00000010 0x0
   ECDSA_P384                    0x00000005 0x00000010 0x0
   ECDSA_P521                    0x00000005 0x00000010 0x0
```

To list details of the algorithms for a specific named key storage provider, run the **cnglist** command with the **--list-algorithms** and **--provider**="*ProviderName*" options:

```
cnglist --list-algorithms --provider="Microsoft Software Key Storage Provider"
```

Output from this command has the form:

```
Microsoft Software Key Storage Provider
NCryptEnumAlgorithms(NCRYPT_CIPHER_OPERATION) no supported algorithms
NCryptEnumAlgorithms(NCRYPT_HASH_OPERATION) no supported algorithms
NCryptEnumAlgorithms(NCRYPT_ASYMMETRIC_ENCRYPTION_OPERATION):
   Name                          Class      Operations Flags
   RSA                           0x00000003 0x00000014 0x0
NCryptEnumAlgorithms(NCRYPT_SECRET_AGREEMENT_OPERATION):
   Name                          Class      Operations Flags
   DH                            0x00000004 0x00000008 0x0
   ECDH_P256                     0x00000004 0x00000018 0x0
   ECDH_P384                     0x00000004 0x00000018 0x0
   ECDH_P521                     0x00000004 0x00000018 0x0
NCryptEnumAlgorithms(NCRYPT_SIGNATURE_OPERATION):
   Name                          Class      Operations Flags
   RSA                           0x00000003 0x00000014 0x0
   DSA                           0x00000005 0x00000010 0x0
   ECDSA_P256                    0x00000005 0x00000010 0x0
   ECDSA_P384                    0x00000005 0x00000010 0x0
   ECDSA_P521                    0x00000005 0x00000010 0x0
```

configure-csp-poolmode

The **`configure-csp-poolmode`** utility allows HSM Pool mode to be enabled or disabled for the nCipher CNG CSP without using the CNG wizard.

To enable HSM Pool mode for CNG run the command:

```
configure-csp-poolmode --cng --enable
```

To disable HSM Pool mode for CNG run the command:

```
configure-csp-poolmode --cng --disable
```

To remove HSM Pool mode setting for CNG from the registry, use the command:

```
configure-csp-poolmode --cng --remove
```

Chapter 8: Remote Operator

This chapter explains:

- The concept of Remote Operator
- How to configure Remote Operator.

Note: If you wish to use the Remote Operator feature, you must have enabled it as described in *Enabling optional features on the module* on page 46. 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.

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. (You can import or export slots only if the module uses firmware version 2.6.10 or higher.) Before you can import a slot to one module, you must first export it from another module.

Configuring Remote Operator

This section explains how to configure Remote Operator.

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, must have firmware version 2.6.10 or greater, 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. See *Appendix K: Checking and changing the mode on an nShield Solo module* on page 268 or *Appendix L: Checking and changing the mode on an nShield Edge* on page 275 for more about changing the mode.

2. Configure the HSM hardservers on their respective host machines for slot import and export, as appropriate.

Note: Dynamic Slots cannot be exported or imported 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 Operator).
2. Generate keys that are protected by the Remote OCS.
3. Ensure your application is configured to use keys protected by the Remote OCS.

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* on page 77.

   Note: 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`** *MODULE* (where *MODULE* is the HSM's ID number).
2. Check whether firmware version 2.6.10 or greater is installed in both the attended and unattended HSMs by running the **`enquiry`** command-line utility.
   If an earlier version of firmware is installed, you must upgrade the HSM firmware as necessary; see *Upgrading firmware* on page 277.
   - If you upgrade an HSM's firmware, you must reinitialize it into the Security World, by using KeySafe, or the **`new-world`** command-line utility.
3. 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
```

Configuring hardservers for Remote Operator

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* on page 172.

Ensure that your network firewall settings are correct. See the Installation Guide for more information about firewall settings.

When the HSMs have been configured, follow these steps to configure the hardservers:

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*). Follow these steps:
   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
```

   Note: For more information about the parameters controlled by the **slot_exports** section of the hardserver configuration file, see *slot_exports* on page 254.
   b. Run the **cfg-reread** command-line utility to prompt the hardserver to read the configuration 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**.
   Note: For more information about the parameters controlled by the **slot_imports** section of the hardserver configuration file, see *slot_exports* on page 254.
   b. Run the **cfg-reread** command-line utility to prompt the hardserver to read the configuration changes.

You can check that the slot was imported successfully by, on the unattended machine, running the command:

```
slotinfo -m 1
```

If slot importation 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 pass phrase, 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* on page 37).

The same applies for any keys that an application on an unattended HSM needs to load but that were not generated on that machine.

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.

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* on page 94.

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.

Note: 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.

Note: 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.

Note: 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.

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* on page 124. For more information about Remote Operator slots, see *Remote Operator* on page 171.

Note: 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.

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* on page 177.

Note: 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.

Note: 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.

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.

Chapter 9: 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

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.

Note: Some applications can generate keys directly.

When you attempt to generate keys for a Security World that complies with FIPS 140-2 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.

Note: 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**
- **seessl**

These requests are generated in PKCS #10 format with base-64 encoding.

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 and lowercase letters; it cannot contain spaces, underscores (_), or hyphens (-).

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.

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*), see *Key application type (APPNAME)* on page 261 in *Key generation options and parameters* on page 261.
- 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)* on page 262 in *Key generation options and parameters* on page 261.

For details of the available application types (**APPNAME**) and parameters that control other key properties (**NAME=VALUE**), see *Key generation options and parameters* on page 261.

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 140-2 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 pass phrases, as required.

Example of key generation with generatekey

To generate in batch mode an RSA key protected by an OCS named **operatorone** and for an application that uses the Cryptographic Hardware Interface Library (CHIL), use the command:

```
generatekey --generate --batch --cardset=operatorone hwcrhk protect=token type=rsa size=1024
plainname=keya ident=abc
```

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.

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* on page 186.)
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.
   Note: In order to generate a key protected by a FIPS 140-2 level 3 compliantSecurity 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 pass phrases 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.

Generating NVRAM-stored keys

NVRAM key storage provides a mechanism for generating keys stored in a module's nonvolatile 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 the keys would no longer be stored solely within the physical boundary of the HSM.

Note: 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.

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 pass phrase.
- DES, DES2 and Triple DES keys (entered in hex).

Note: You cannot import keys into a Security World that complies with FIPS 140-2 level 3. Attempting to import keys into a FIPS 140-2 Level 3 Security World returns an error.

This request is a PKCS #10 format request in base-64 encoding.

Importing keys from the command line

You can import keys using the **`generatekey`** command-line 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 pass phrases, as required.
- If prompted, insert an Administrator Card or an Operator Card from the current Security World.

Example of key importation with generatekey

To import an RSA key stored in **C:\projects\key.pem** for use with an nShield native application and protect it with the Security World, use the command:

```
generatekey --import simple pemreadfile=C:\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.

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* on page 186.)
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 pass phrases 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.

Listing supported applications with generatekey

To list supported applications, use the command:

```
generatekey --list-apps
```

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 application. Because no key material is exposed during retargeting, this operation is as secure as generating a new key.

Note: 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.

Option	Description
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`.

Viewing keys

You can view existing keys in the Security World using KeySafe or the **nfkminfo** command-line utility.

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* on page 186.)
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* on page 179)
   - import a key (see *Importing keys with KeySafe* on page 181)
   - discard a key from the security world (see *Discarding keys* on page 185)

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**
- **hwcrhk**
- **pkcs11**
- **kpm**
- **kps**
- **mscapi**
- **seeconf**
- **seeinteg**
- **simple**

You can also specify your own application names for *APPNAME* as appropriate to your system.

Note: 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 '
```

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.

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.

Note: If you have NVRAM-stored keys, you must additionally ensure you have a backup of the key data stored on the relevant modules.

Appendix A: Using KeySafe

KeySafe provides a GUI based interface to perform many of the main tasks required to use an nCipher 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.

Note: By default, KeySafe uses the same mechanisms and supports the same features and applications as the **generatekey** command-line utility.

Setting up KeySafe

1. You must have Java JRE/JDK 1.6, 1.7 or 1.8. We recommend that you install Java before you install the Security World Software.
   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.
   Note: 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 **%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.
   Note: See the Installation Guide for more about ports and firewall settings.

Starting KeySafe

Start KeySafe by selecting the **KeySafe** entry from the Start menu under **All Programs** > **nCipher**. You may need administrator privileges to run KeySafe.

The Windows KeySafe launcher checks that the components required to run KeySafe are installed. You will be prompted to install any missing components.

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.

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).

Note: The options listed below are also available from the **Manage** menu.

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: • Initialize a Security World • Add modules to a Security World • Remove modules from a Security World. **Note:** You cannot perform these operations on a module that is not in the pre-initialization mode.

Menu button	Description
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 pass phrase • 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: • Createa new softcard • Change or recover the pass phrase 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.

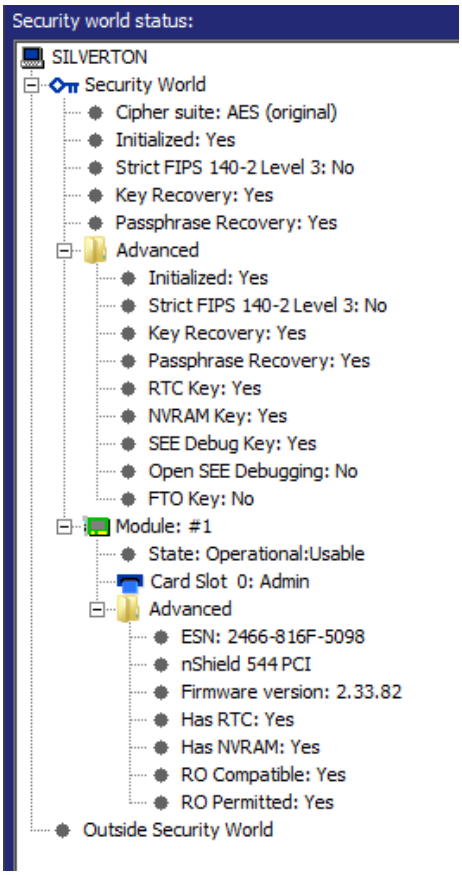
Menus

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* on page 187.
  - **World** - opens the **World Operations** panel. See *World* on page 187.
  - **Card sets** - opens the **List Operator Card Sets** panel. See *Card Sets* on page 188.
  - **Softcards** - opens the **List Softcards** panel. See *Softcards* on page 188.
  - **Keys** - opens the **Keys** panel. See *Keys* on page 188.
- **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 numbers if you contact Support about KeySafe.

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.

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 140-2 Level 3** — whether the Security World is operating at FIPS 140-2 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 *Pass phrase replacement* on page 61

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)

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.
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).

  Note: For FIPS 140-2 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** —

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.

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 correct 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.

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).

Errors

If KeySafe detects an error from which it cannot recover, it may display a Fatal Error message.

Error message	Possible causes	Suggested solutions
Unable to establish KeySafe session. Please ensure that the hardserver is running and accepting TCP connections. Click OK to exit.	• 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.	• Check the hardserver configuration file settings: see *server_startup* on page 252. • To restart the hardserver: 1. Exit KeySafe 2. Restart the server (as described in *Stopping and restarting the hardserver* on page 55) 3. Restart KeySafe.
Unable to generate key: Error reported by nShield hardware module nFast error: UnknownFlag, in response to GenerateKayPair	Your hardware or firmware may not be up to date.	To update your firmware: 1. Exit KeySafe 2. Update the firmware as described in *Upgrading firmware* on page 277 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 module must survive intact during the upgrade or initialization of the key-management module, a backup and recovery mechanism of your *%NFAST_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.

Appendix B: Warrant Management

This appendix describes how you can ensure that a suitable warrant is available to allow an nShield Remote Administration Card to be used with an nShield Solo or nShield Edge.

To be able to use an nShield Remote Administration Card you need to ensure that:

- The appropriate firmware is installed on your nShield Solo or nShield Edge (2.61.2 or later) See *Appendix M: Upgrading firmware* on page 277 for more about firmware versions.
- The nShield Solo or nShield Edge has a KLF2 warrant installed in the appropriate place.

Warranting steps

Note: You need an appropriate support contract to obtain a KLF2 warrant from nCipher.

Ensure v12.xx Security World Software has been installed on your host computer (to access the nfwarrant tool and the nShield Solo or nShield Edge has 2.61.2 firmware or later installed.

You then need to carry out the following steps to ensure a suitable warrant is available

1. Check if the relevant module has the appropriate firmware.
2. Check if a warrant upgrade is required, if so, follow steps 3-6.
3. Generate a Certificate Signing Request (CSR) for the warrant.
4. Send the CSR to nCipher.

   Note: Ensure that the ESN contained in the upgrade request is the one that belongs to the relevant module, for example, by running the **`nfkminfo`** command-line utility. See *Displaying information about a Security World with nfkminfo* on page 76 for more about viewing an ESN.
5. Validate the warrant that you receive from nCipher to ensure that it matches the sent request.
6. Install the warrant.

nfwarrant command-line utility

The **`nfwarrant`** command-line utility enables you to carry out all of the relevant warrant steps. It is used to:

- Identify modules that have the appropriate firmware and KLF2 key
- Identify modules that need their KLF2 key to be warranted by nCipher
- Generate a warrant upgrade request for a specific module, as required
- Install an upgraded warrant
- List KLF2 warrants

Running nfwarrant

Usage

```
nfwarrant [--help] [--list] [--check] [--warrant] [--csr] [--details= FILE]
[--install= FILE] [--req= MODULE] [--out= FILE] [--verbose] [--version]
```

Options

Option	Description
-h, --help	Displays the options you can use with the utility.
--list	List ESNs of installed warrants
--check	List ESNs of known modules and their warrant state
--warrant	Perform warrant operations
--csr	Perform CSR operations
--details=*<file>*	Display the module ESN found in the CSR/warrant *<file>*
--install= <*file*>	Install the warrant from *<file>*
--req=*<module>*	Request a warrant CSR for the given module number/ESN
--out=*<file>*	Save the new requested CSR to *<file>*
--verbose	Print extra information about CSR and warrant files
--version	Print the version number of the nfwarrant tool

Checking the available hardware

Run the following command:

```
$ nfwarrant --check
```

The following is an example output:

```
1 XXXX-XXXX-E0D2 Local, Warrant installed
2 XXXX-XXXX-CF11 Local, Warrant upgrade request possible
3 XXXX-XXXX-F1F2 Local, Warrant upgrade not supported
4 XXXX-XXXX-213B Remote, Warrant upgrade not required
```

In this example:

- (1) already has a relevant warrant installed.
- (2) is available for a warrant upgrade.

- (3) cannot be upgraded. For example, the appropriate firmware is not installed.
- (4) no warrant upgrade is required. The module is an nShield Connect.

Generating a warrant upgrade request

Run the following command:

```
$ nfwarrant --csr --req <module>
```

The following is an example output, displaying the location of the resultant upgrade request for a module with ESN XXXX-XXXX-CF11:

```
CSR written to '/opt/nfast/kmdata/warrants/csr_XXXX-XXXX-CF11'
```

Ensure that the ESN in this request file is the correct one and send the file to nCipher to be signed.

Validating the warrant you receive from nCipher

1. Run the following command:

   ```
   $ nfwarrant --warrant --details <file>
   ```

   The following is an example output:

   ```
   Warrant details:
    Filename: XXXX-XXXX-CF11
    ESN: XXXX-XXXX-CF11
    Keytype: ECDSAPublic
    Curve: NISTP521
   ```

2. Compare the ESN in the file received from nCipher with the one in the original request, by running the following command:

   ```
   $ nfwarrant --csr --details <file>
   ```

   The following is an example output:

   ```
   XXXX-XXXX-CF11
   ```

Installing a warrant

Run the following command:

```
$ nfwarrant --warrant --install <file>
```

<file> is the signed warrant provided by nCipher.

Appendix C: 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.

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.

Utility	Enables you to...
enquiry	Obtain information about the hardserver (Security World Software server) and the modules connected to it. Use this utility to: • 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 Solos See the Installation Guide for more information.
checkmod	Check modulo exponentiations performed on the module against the test data located in the *%NFAST_HOME%***testdata** directory.
cfg-mkdefault	Create a default client configuration file for the hardserver configuration sections.
cfg-reread	Load the hardserver configuration from the configuration file.

Utility	Enables you to...
`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. **Note:** To enable features, and view the status of or verify features on an nShield Connect unit, use the front panel rather than the `fet` utility. For more information, see *Enabling features* on page 52.
`ncdate`	View, set, and update the time on a module's real-time clock.
`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
`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* on page 226. **Note:** Run this utility only if requested to do so by Support.
`nfwarrant`	Ensure that a suitable warrant is available to allow a Security World to be dynamically managed using an nShield Solo or nShield Edge. • Identify modules that have the appropriate firmware/KLF2 key • Identify modules that need their KLF2 key to be warranted by nCipher • Generate a warrant upgrade request for a specific module, as required • Install an upgraded warrant • List KLF2 warrants See *Appendix B: Warrant Management* on page 193 for more information.

Utility	Enables you to...
`nopclearfail`	Clear an HSM, put an HSM into the error state, retry a failed HSM, or change the HSM mode. See *Appendix K: Checking and changing the mode on an nShield Solo module* on page 268for more about changing the mode.
`nvram-backup`	Copy files between a module's NVRAM and a smart card, allowing files to be backed up and restored.
`nvram-sw`	View and modify information about NVRAM areas.
`pubkey-find`	Obtain information of the public key from a certificate or certificate request (in a Base-64 encoded PEM file).
`randchk`	Run a universal statistical test on random numbers returned by the module.
`rtc`	View and set the module's real-time clock.
`slotinfo`	• Obtain information about tokens in a module • Format a smart card

`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* on page 313

`stattree`	Obtain statistics gathered by the Security World Software server and modules. For more information, see *stattree: information utility* on page 239.

Hardware utilities

Use the following utilities to manage the firmware installed on an nShield HSM.

Utility	Enables you to...
`fwcheck`	Verify the firmware installed on a module.
`loadrom`	• Upgrade the module firmware • Obtain information about the firmware installed on a module To determine the version security number of the firmware in a file and for more information, see *Firmware on the installation media* on page 277. **Note:** The `loadrom` command is intended to update Solo and Edge HSMs; it is not intended to be used to update a Connect firmware image.

Utility	Enables you to...
`nfloadmon`	Upgrade the module monitor and firmware of nShield Edge and nShield Solo modules. For more information, see *Upgrading firmware* on page 277.

Test analysis tools

Use the following utilities to test the cryptographic operational behavior of a module.

Note: All the listed utilities, except the `floodtest` utility, are supported only on FIPS 140-2 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* on page 237.
`sigtest`	Measure module speed using RSA or DSA signatures or signature verifications.

Security World utilities

Use the utilities described in this section to:

- Set up and manage Security Worlds.
- Create and manage card sets and pass phrases.
- 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. **Note:** Do not use the `bulkerase` utility to erase Administrator Cards from the current Security World.

Utility	Enables you to...
`cardpp`	Change, verify, and recover a pass phrase of an Operator Card. For more information, see: • *Verifying the pass phrase of a card with cardpp* on page 107. • *Changing known pass phrase with cardpp* on page 110. • *Changing unknown or lost pass phrase* on page 111.
`createocs`	Create and erase an OCS. For more information, see: • *Creating an Operator Card Set using the command line* on page 94. • *Erasing cards using the command line* on page 103.
`initunit`	Initialize an nShield module. For more information, see *Erasing a module with initunit* on page 90.
`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 using the command line* on page 177. • *Importing keys from the command line* on page 180. • *Example of key generation with generatekey* on page 178, for an example of key generation in batch mode. • *Example of key importation with generatekey* on page 181, for an example of importing an RSA key. • *Listing supported applications with generatekey* on page 182. • *Retargeting keys with generatekey* on page 182.
`key-xfer-im`	Transfer keys between Security Worlds, when used in conjunction with the `mk-reprogram` utility. During the key transfer process, the `key-xfer-im` utility imports the module key that has been programmed into a new Security World. For more information, see *Transferring keys between Security Worlds* on page 79. **Note:** You can also use the `migrate-world` utility to transfer keys between Security Worlds.
`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* on page 83.

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. **Note:** 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.
`mk-reprogram`	Transfer keys between Security Worlds, when used in conjunction with the `key-xfer-im` utility. During the key transfer process, the `mk-reprogram` utility programs the module key of a Security World to another Security World. For more information, see *Transferring keys between Security Worlds* on page 79. **Note:** You can also use the `migrate-world` utility to transfer keys between Security Worlds.
`new-world`	Create and manage Security Worlds on nShield modules. For more information, see: • *Creating a Security World using new-world* on page 63. • *Adding an HSM to a Security World with new-world* on page 79. • *Erasing a module with new-world* on page 90.
`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* on page 76. • *Viewing card sets using the command line* on page 105. • *Viewing softcards with nfkminfo* on page 106. • *Viewing keys using the command line* on page 183. • *nfkminfo: information utility* on page 228.
`nfkmverify`	Perform Security World verification.
`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. **Note:** 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 pass phrase of a softcard For more information, see: • *Creating a softcard with ppmk* on page 100. • *Erasing softcards with ppmk* on page 104. • *Viewing softcards with ppmk* on page 107. • *Verifying the pass phrase of a softcard with ppmk* on page 108. • *Changing known softcard pass phrase with ppmk* on page 110. • *Replacing unknown pass phrase with ppmk* on page 111.
`preload`	Load keys into a module before an application is run in another session. For more information, see *Transferring keys between Security Worlds* on page 79.
`racs`	Create a new ACS to replace an existing ACS. For more information, see *Replacing an Administrator Card Set with racs* on page 123.
`rocs`	• Restore an OCS from a quorum of its cards • Restore softcards For more information, see: • *Replacing OCSs or softcards with rocs* on page 114. • *Using rocs from the command line* on page 119.

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.

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 nCipher PKCS #11 libraries. For more information, see *Checking the installation of the nCipher PKCS #11 library* on page 148.
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. **Note:** 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 nCipher PKCS #11 library calls.

If you have installed **Cipher Tools**, you can 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.
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**.

Utility	Enables you to...
`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.

nShield Connect utilities

The utilities described in this section are used with nShield Connect only. Use these utilities to:

- Create and manage client configuration files.
- Enroll nTokens with an nShield Connect.
- Set up a Remote File System (RFS) and synchronize Security World data between an nShield Connect and the RFS.
- Administer an nShield Connect without using the front panel

Utility	Enables you to...
`anonkneti`	View the ESN and HK_{NETI} key hash from a module identified by its IP address.
`cfg-pushnethsm`	Copy a specified configuration file from a remote file system to the file system on a specified module.
`config-serverstartup`	Edit the `[server_startup]` section of the configuration file for the client's hardserver to enable or disable TCP sockets.
`nethsmadmin`	Administer an nShield Connect without using the front panel. Options include: • Check the Security World files on a specified nShield Connect • Copy Security World files from the RFS to the nShield Connect • Command the specified nShield Connect to reboot • Command the nShield Connect to upgrade using the specified nShield Connect image file from its RFS • Retrieve a list of nShield Connect image files available on the RFS • Retrieve a list of feature certificates available on the RFS for a specified nShield Connect • Command the nShield Connect to apply a specified feature certificate from the RFS
`nethsmenroll`	Edit the local hardserver configuration file to add the specified nShield Connect unit. As an alternative to hand-editing a client's configuration file, you can run this utility on a client to configure it to access an nShield Connect.
`ntokenenroll`	Enroll a locally attached nToken with an nShield Connect unit. This utility installs the Electronic Serial Number (ESN) of the nToken within the client configuration file and displays the module's ESN and the hash of the key to be used in nToken authentication.

Utility	Enables you to...
`rfs-setup`	Create a default RFS hardserver configuration on the client. Run this utility when you first configure a client.
`rfs-sync`	• Synchronize the Security World data of a client with the RFS, when you run the utility on the client • Synchronize the Security World data of the RFS with the client, when you run the utility on the RFS For more information, see: • *Setting up client cooperation* on page 37. • *rfs-sync* on page 39. **Note:** You can use this utility with nShield modules if an nShield Connect unit is present.

MSCAPI utilities

Use the following utilities to migrate from Windows registry-based CSP container storage to the new CSP formats. These utilities also enable you to manage the interfaces between the MSCAPI library and the module.

For more information about these utilities, see *Utilities for the CAPI CSP* on page 153.

Note: Utility names that end with `64` run only on 64-bit version of Microsoft Windows. All other utilities run on both 32-bit and 64-bit versions of Microsoft Windows.

Utility	Enables you to...
`cspcheck` `cspcheck64`	Check that CSP container files and keys in the `kmdata` directory are intact and uncorrupted and that the referenced key files exist.
`cspimport` `cspimport64`	Insert keys manually into existing CSP containers. For more information, see *Installing the CAPI CSP* on page 151.
`cspmigrate` `cspmigrate64`	Move CSP container information for an existing Security World from the registry into the Security World.
`cspnvfix` `cspnvfix64`	Regenerate the NVRAM key counter area for a specified nShield CSP key.
`csptest` `csptest64`	Test the installed Cryptographic Service Providers.
`csputils` `csputils64`	Obtain detailed information about CSP containers. **Note:** You must have Administrator privileges to view or delete machine containers or containers that belong to other users.

Utility	Enables you to...
keytst **keytst64**	Create, test, and display information about keys and CSP key containers.
configure-csp-poolmode **configure-csp-poolmode64**	Configure HSM Pool mode for the nCipher CAPI CSP.

CNG

Use the following helper utilities to manage keys and the interfaces between the CNG library and the HSM. For a list of utilities specific to the nCipher CNG CSP, see *Utilities for CNG* on page 164.

Note: Utility names that end with **64** run only on 64-bit version of Microsoft Windows. All other utilities run on both 32-bit and 64-bit versions of Microsoft Windows.

Utility	Enables you to...
cngimport	Migrate Security World, CAPI and CNG keys to the Security World Key Storage Provider. For more information, see: • *Importing a Microsoft CAPI key into the Security World Key Storage Provider* on page 161. • *Importing a Microsoft CNG key into the Security World Key Storage Provider* on page 162. • *Importing a Security World key into the Security World Key Storage Provider* on page 162. • *cngimport* on page 165.
cnginstall **cnginstall64** (nCipher CNG provider installer utility)	Remove or reinstall the provider DLLs and associated registry entries manually. For more information, see *cnginstall* on page 165.
cnglist **cnglist64**	View information about CNG providers. For more information, see: • *Migrating keys for CNG* on page 160. • *Importing a Microsoft CAPI key into the Security World Key Storage Provider* on page 161. • *Importing a Microsoft CNG key into the Security World Key Storage Provider* on page 162. • *Importing a Security World key into the Security World Key Storage Provider* on page 162. • *cnglist* on page 168.

Utility	Enables you to...
cngregister (nCipher CNG provider registration utility)	Unregister and re-register the nCipher providers manually. For more information, see: • *Registering the CNG CSP* on page 155. • *Unregistering or reregistering the CNG CSP* on page 157. • *Uninstalling or reinstalling the CNG CSP* on page 157. • *cngregister* on page 165.
cngsoak **cngsoak64** (nCipher CNG soak tool)	Evaluate the performance of signing, key exchange, and key generation by using a user-defined number of threads. For more information, see *cngsoak* on page 166.
ncsvcdep (nShield Service dependency tool)	Configure service-based applications such as Microsoft Certificate Services and IIS to use the nCipher CNG CSP. Use this tool to add the nFast Server to the dependency list of such services. For more information, see: • *Uninstalling or reinstalling the CNG CSP* on page 157. • *ncsvcdep* on page 167.
configure-csp-poolmode **configure-csp-poolmode64**	Configure HSM Pool mode for the nCipher CNG CSP. For more information, see *configure-csp-poolmode* on page 170.

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. **Note:** 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.

Appendix D: Configuring silent installations

This appendix describes how to record your software installation choices during installation for use in future installations.

When you follow the standard installation instructions for Security World Software, the **setup.exe** installer runs automatically when you place the Security World Software installation media in the optical disc drive. You then follow the on-screen instructions from the installer to configure your installation.

However, if you run the **setup.exe** installer from the command line, you have the option to record your software installation choices. This allows future installations to run 'silently', using your pre-configured choices without the need for further interaction with the installer. Also, if you have previously performed silent installation configuration, you can run the installer with options so that it uses those pre-configured choices.

Note: The silent install program makes a recording of a manual install process, which can be replayed at a later time. However, if the directory structure where the software is to be installed has changed significantly since the original recording was made, the silent install may not be correctly carried out.

Note: See the Installation Guide for more information about installing Security World Software.

Configuring for future silent installations

Before starting, please ensure that:

- Any previously installed Security World Software is uninstalled
- If the directory `C:\Program Files (x86)\nCipher` or `C:\Program Files\nCipher` exists, it is deleted
- If the directory `C:\ProgramData\nCipher` exists, it is renamed or deleted

To run the installer so that it records your installation choices for use in future silent installations:

1. Log in as Administrator or as a user with local administrator rights.
2. Place the Security World Software installation media in the optical disc drive. If the installer runs automatically, quit the installer.
3. Open a Command Prompt, and run the command:

```
E:\setup.exe /r /f1<DRIVE>\<DIRECTORY>\setup.iss
```

In this command:

- **E** is the drive letter of your installation media
- **<*DRIVE*>\<*DIRECTORY*>\setup.iss** specifies the location in which to save the **setup.iss** file. This file stores a record of the choices you make while running the installer.

Note: Do *not* insert a space between the **/f1** option and the **<*DRIVE*>\<*DIRECTORY*>** location. The sequence **/f1<*DRIVE*>\<*DIRECTORY*>\setup.iss** must be entered without any intervening spaces.

4. Follow the installer's onscreen instructions making the appropriate choices for your installation as prompted. Complete the installation process.

After successfully running the installer to create a silent installation configuration file **setup.iss** as described, continue the installation process, as described in the Installation Guide.

In future you can run the installer to use your installation configuration choices saved in the **setup.iss** without the need for further interaction with the installer; see *Using previously configured silent installations* on page 210.

Using previously configured silent installations

Before starting, please ensure that:

- Any previously installed Security World Software is uninstalled
- If the directory `C:\Program Files (x86)\nCipher` or `C:\Program Files\nCipher` exists, it is deleted
- If the directory `C:\ProgramData\nCipher` exists, it is renamed or deleted

To run the installer so that it uses previously configured installation choices for a silent installation:

1. Log in as Administrator or as a user with local administrator rights.
2. Place the Security World Software installation media in the optical disc drive. If the installer runs automatically, quit the installer.
3. Open a Command Prompt, and run the command:

```
E:\setup.exe /s /f1<DRIVE>\<DIRECTORY>\setup.iss /f2<DRIVE>\<DIRECTORY>\setup.log
```

In this command:

- **E** is the drive letter of your installation media
- **<*DRIVE*>\<*DIRECTORY*>\setup.iss** specifies the location of the **setup.iss** file that contains your previously configured installation choices (*Configuring for future silent installations* on page 209)
- **<*DRIVE*>\<*DIRECTORY*>\setup.log** specify the location in which to save the **setup.log** file (with a file name of your choice) that captures results from the silent installation.
- Do *not* insert a space between the **/f1** option or **/f2** and the **<*DRIVE*>\<*DIRECTORY*>** locations. These sequences must be entered without any intervening spaces.

The software is installed without the need for further interaction with the installer. You see no direct feedback from the installer, but the task manager shows when the software installation process is complete.

4. When the installation is complete, locate the generated log file (for example, **setup.log**), and check the section **[ResponseResult]**. A value of **0** for the **ResultCode** entry confirms a successful silent installation:

```
[ResponseResult]
ResultCode=0
```

For example, if the Security World Software installer is located on the **E:** drive and the **.iss** file is called **setup.iss** and located on local drive **C:**, to run the installer so that it uses those previously configured installation choices for a silent installation, you must run the command:

```
E:\setup.exe /s /f1C:\setup.iss /f2C:\setup.log
```

5. This example command runs the installer on the **E:** drive in silent mode, loads the **setup.iss** file, and generates an installation log file named **setup.log** that is saved on the **C:** drive.

After a successful silent installation as described, continue the installation process normally as described in the Installation Guide.

Appendix E: Configuring silent uninstall

This appendix describes how to record your software uninstall choices during uninstallation for use in future uninstallations.

When the Security World Software **setup.exe** uninstaller is launched from the **Programs and Features** applet, you follow the standard uninstallation instructions for removing the software from your system.

However, if you run the **setup.exe** uninstaller from the command line, you have the option to record your software uninstallation choices. This allows future uninstallations to run 'silently', using your preconfigured choices without the need for further interaction with the uninstaller. Also, if you have a previously performed silent uninstallation configuration, you can run the uninstaller with options so that it uses those pre-configured choices.

The first part of this procedure describes how to record the un-install process for a previously successful installation.

Note: The silent uninstall program makes a recording of a manual uninstall process, which can be replayed at a later time. However, if the directory structure where the software is to be uninstalled from has changed significantly since the original recording was made, the silent uninstall may not be correctly carried out.

Configuring for future silent uninstallations

To run the uninstaller so that it records your uninstallation choices for use in future:

1. Log in as Administrator or as a user with local administrator rights.
2. Open a Command Prompt, and run the command:

```
"C:\Program Files (x86)\InstallShield Installation Information\{A4327200-59D7-11D2-80B2-
0080C8445B66}\setup.exe" /runfromtemp /r /L0x0409 /removeonly /uninst
/f1<DRIVE>\<DIRECTORY>\uninstall.iss
```

   In this command:
   - The directory *Program Files (x86)* is correct for 64-bit Windows systems and will be *Program Files* for 32-bit Windows systems.
   - *<DRIVE>**<DIRECTORY>***uninstall.iss** specifies the location in which to save the **uninstall.iss** file. This file stores a record of the choices you make while running the uninstaller.
   - The **/r** denotes that the command will record while performing the uninstall process.

     Note: Do *not* insert a space between the **/f1** option and the *<DRIVE>**<DIRECTORY>* location. The sequence **/f1***<DRIVE>**<DIRECTORY>***uninstall.iss** must be entered without any intervening spaces.
3. Follow the uninstaller's onscreen instructions as required.

In future you can run the uninstaller to use your uninstall configuration choices saved in the **uninstall.iss** without the need for further interaction with the uninstaller, see *Using previously configured silent uninstallations* on page 213.

Using previously configured silent uninstallations

To run the uninstaller so that it uses previously configured uninstall choices for a silent uninstallation:

1. Log in as Administrator or as a user with local administrator rights.
2. Open a Command Prompt, and run the command:

```
"C:\Program Files (x86)\InstallShield Installation Information\{A4327200-59D7-11D2-80B2-
0080C8445B66}\setup.exe" /runfromtemp /s /L0x0409 /removeonly /uninst
/f1<DRIVE>\<DIRECTORY>\uninstall.iss /f2<DRIVE>\<DIRECTORY>\uninstall.log
```

   In this command:
   - The directory **Program Files (x86)** is correct for 64-bit Windows systems: for 32-bit Windows systems, use **Program Files**.
   - *<DRIVE>**<DIRECTORY>***uninstall.iss** specifies the location of the **uninstall.iss** file that contains your previously configured installation choices (see *Configuring for future silent uninstallations* on page 212).
   - *<DRIVE>**<DIRECTORY>***uninstall.log** specifies the location in which to save the **uninstall.log** file (with a file name of your choice) that captures results from the silent installation.

     Note: Do *not* insert a space between the **/f1** or **/f2** options and the *<DRIVE>**<DIRECTORY>* locations. These sequences must be entered without any intervening spaces.

   The software is uninstalled without the need for further interaction with the uninstaller. You see no direct feedback from the uninstaller, but the task manager will show when the software uninstallation process is complete.
3. When the uninstallation is complete, locate the generated log file (for example, **uninstall.log**), and check the section **[ResponseResult]**. A value of **0** for the **ResultCode** entry confirms a successful silent uninstallation:

```
[ResponseResult]
ResultCode=0
```

   For example, if the Security World Software is installed on a 32-bit Windows system, the **.iss** file is called **uninstall.iss** and is located on local drive **C:**, to run the uninstaller so that it uses those previously configured uninstallation choices for a silent uninstallation, you must run the command:

```
"C:\Program Files\InstallShield Installation Information\{A4327200-59D7-11D2-80B2-
0080C8445B66}\setup.exe" /runfromtemp /s /L0x0409 /removeonly /uninst
/f1C:\uninstall.iss /f2C:\uninstall.log
```

This runs the uninstaller in silent mode, loads the **`uninstall.iss`** file, and generates an uninstallation log file named **`uninstall.log`** that is saved on the **`C:\`** drive.

Appendix F: Environment variables

This appendix describes the environmental variables used by Security World Software:

Variable	Description
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**.
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* on page 219). For more information, see also *Logging and debugging information for PKCS #11* on page 223 and *Hardserver debugging* on page 224.
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.
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**.
NFAST_HWCRHK_HSM_POOL	This variable configures HSM Pool mode for CHIL (Cryptographic Hardware Interface Library). Set the variable to 1, y or Y to enable HSM Pool mode for the CHIL-enabled application, or set to 0, n or N to explicitly disable HSM Pool mode for the CHIL-enabled application. Do not configure this variable with applications using other APIs.
NFAST_HWCRHK_LOGFILE	This variable sets the name of the file to which the CHIL (Cryptographic Hardware Interface Library) writes its log from applications. This is in addition to the logging done according to the mechanisms in the published **hwcrhk** API, which vary according to the application in use.

Variable	Description
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**.
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**.
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**.
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.
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.
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.
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.
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.

Variable	Description
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.
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.
NFAST_SERVER **NFAST_PRIVSERVER**	If these variables are set in the hardserver's environment, the values specify 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 pipes in the hardserver configuration file, see *server_startup* on page 252. If you set these variables they override the values in hardserver configuration file.
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* on page 252. If you set these variables, they override the values in the hardserver configuration file.
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* on page 219. If no value is supplied, all message categories are logged.
NFLOG_SEVERITY	This variable is used to filter log messages by supplying a minimum severity level to be logged; see *Logging, debugging, and diagnostics* on page 219. If no value is supplied, the default severity level is **WARNING**.

Variable	Description
`NFLOG_DETAIL`	This variable is used to filter log messages by supplying a bitmask of detail flags; see *Logging, debugging, and diagnostics* on page 219. The default is `time+severity+writeable`.
`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* on page 219. The default is `stderr` (the equivalent of file descriptor `&2`).
`OPENSSL_HWCRHK_LOG`	This variable sets the directory in which the CHIL (Cryptographic Hardware Interface Library) creates its log file. This must be a directory for which the user (that the CHIL-enabled application runs as) has write permission.

Note: When using these environment variables 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.

Appendix G: 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.

Logging and debugging

Note: 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.

Note: 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.

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.

Note: 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* on page 248):

Hexadecimal flag	Function	logdetail flags
0x00000001	This flag shows the external time (that is, the time according to your machine's local clock) with the log entry. It is on by default.	external_time
0x00000002	This flag shows the external date (that is, the date according to your machine's local clock) with the log entry.	external_date
0x00000004	This flag shows the external process ID with the log entry.	external_pid
0x00000008	This flag shows the external thread ID with the log entry.	external_tid
0x00000010	This flag shows the external **time_t** (that is, the time in machine clock ticks rather than local time) with the log entry.	external_time_t
0x00000020	This flag shows the stack backtrace with the log entry.	stack_backtrace
0x00000040	This flag shows the stack file with the log entry.	stack_file
0x00000080	This flag shows the stack line number with the log entry.	stack_line
0x00000100	This flag shows the message severity (a severity level as used by the **NFLOG_SEVERITY** environment variable) with the log entry. It is on by default.	msg_severity
0x00000200	This flag shows the message category (a category as used by the **NFLOG_CATEGORIES** environment variable) with the log entry.	msg_categories
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

NFLOG_CATEGORIES

This environment variable takes a colon-separated list of categories on which to filter log messages (categories may contain the wild-card characters * and ?). If you do not supply any values, then all categories of messages are logged. This table lists the available categories:

Category	Description
nflog	Logs all general messages relating to nflog.
nflog-stack	Logs messages from **StackPush** and **StackPop** functions.
memory-host	Logs messages concerning host memory.
memory-module	Logs messages concerning module memory.
gs-stub	Logs general generic stub messages. (Setting this category works like using the **dbg_stub** flag with the logging functionality found in previous Security World Software releases.)
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 *%NFAST_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.
hwcrhk	Logs messages from the CHIL (Cryptographic Hardware Interface Library).
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* on page 248.

Note: 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.

Logging from the nShield CSP and CNG

By default, logging is disabled for the nShield CSP and CNG.

To enable logging, use a suitable registry editor such as **regedit**.

Depending on whether the program you wish to debug is 64-bit or 32-bit based, you will have to create respective registry keys if they do not already exist.

Note: If using a 32-bit OS you can only use a 32-bit based program.

For a 64-bit program on a 64-bit OS, or a 32-bit program on a 32-bit OS, create the following registry key if it does not already exist: **HKEY_LOCAL_MACHINE\Software\nCipher\CryptoNG**

Open the registry at the required **Cryptography** key as described above, and under the key create the following variables.

1. Create a new **string** variable named **PathName**.
2. Open the **PathName** variable and provide a value which is a suitable path to where you want the log file(s) to be placed (for example, **C:\Users\MyName\Documents**.) Do not give a log file name. The log file name(s) will be created automatically when logging starts.
   Note: It must be possible for the provider to write to the specified path.
3. Create a new **DWORD (32 bit)** variable named **LogLevel**.
4. Open the **LogLevel** variable and provide a suitable value (for example, **2**).
   Permitted values for **LogLevel** are:

Value	Output
0	Messages are sent to the event log.
1	Error messages are sent to the log file.
2	Function calls and error messages are sent to the log file.
3	All information, including debugging information, is sent to the log file.

Note: Do not set this value to 3 unless either you are asked to do so by Support or you are debugging your own code. At this level of logging, a single SSL connection generates approximately 30 kilobytes of log messages. In addition, sensitive information may appear in the log file.
Note: If **LogLevel** is not set, then the provider only logs messages of warning severity or greater (equivalent to setting **NFLOG_SEVERITY=warning**).

If neither **PathName** nor **LogLevel** are set for the CSP or CNG, logging remains disabled.

If logging is successfully enabled, the log file(s) should appear at the location specified in **PathName**, and will have names similar to:

- **nfdebug.txt**
- **ncspdddebug.txt**
- **nckspswdebug.txt**

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 `%NFAST_HOME%\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:"**

PKCS #11 debug level	PKCS #11 debug meaning	NFLOG_SEVERITY value	Output in log
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 "`

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* on page 248. However, unless you also set the **NFAST_DEBUG** environment variable to a value in the range 1 – 7, no debugging is produced (regardless of whether or not you specify **servdbg-*** categories in the hardserver configuration file). This behavior helps guard against the additional load debugging places on the CPU usage; you can set the desired **servdbg-*** categories in the hardserver configuration file, and then enable or disable debugging by setting the **NFAST_DEBUG** environment variable.

The **NFAST_DEBUG** environment variable controls debugging for the general stub or hardserver. The value is an octal number, in the range 1 – 7. It refers bitwise to a number of flags:

Flag	Result
1	Generic stub debugging value.
2	Show bignum values.
4	Show initial **NewClient** or **ExistingClient** command and response.

For example, if the **NFAST_DEBUG** environment variable is set to 6, flags 2 and 4 are used.

Note: 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.

Debugging information for Java

This section describes how you can specify the debugging information generated by Java.

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**)

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
```

Note: 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.

Note: Set these variables only when developing code or at the request of Support.

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.

Diagnostics and system information

Note: 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 nCipher HSM. This tool is supplied separately. If you require a copy, contact your Sales representative.

nfdiag: diagnostics utility

The **nfdiag** command-line utility is a diagnostics tool that gathers information about the system 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.

Usage

Note: Run **nfdiag** with the standard **-h|--help** option to display information about the options and parameters that control the program's behavior.

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 nCipher 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.

Option	Actions to take
any additional diagnostic file(s)	By default, **nfdiag** asks if you want to supply any additional diagnostic files. If you do not want to supply additional diagnostic files, type `N`, or `Enter` to continue. If you want to supply additional diagnostic files: • Type `Y`. • When prompted, supply the name of the file to be attached. Attached files must be in plain text format. • After you have supplied the name of a file to attach, **nfdiag** asks again if you want to supply any additional diagnostic files. Repeat the process to attach additional files. • When you do not want to attach any more files, type `N`, or `Enter` to continue.

Note: 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.

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 **%NFAST_HOME%\log\logfile** exists, **nfdiag** automatically includes this file in its output. If the file **%NFAST_HOME%\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.

Note: The **nfdiag** utility does not capture any pass phrases in the output file.

nfkminfo: information utility

The **nfkminfo** utility displays information about the Security World and the keys and card sets associated with it.

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...]]
```

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.

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.

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 Useable
 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 Pass phrase
 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
```

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.

`StrictFIPS140`

This indicates that the Security World is FIPS 140-2 level 3 compliant. If this is not shown, the Security World is level 2 compliant only.

Note: This indicates that your Security World is compliant 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.

`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 pass phrase replacement feature enabled.

!PINRecovery

This indicates that the Security World has the pass phrase 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.

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.

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 Solos 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).

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 nShield Remote Administration Client has associated a card reader with this Dynamic Slot.

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 pass phrase.

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.

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 pass phrase replacement enabled. This is always true if pass phrase replacement is disabled for the Security World.

PINRecoveryRequired(enabled)

This indicates that the card in the slot does have pass phrase 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.

perfcheck: performance measurement checking tool

Use the **perfcheck** command-line utility to run various tests measuring the cryptographic performance of an nCipher HSM.

Note: 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 must supply the number of the desired test, along with its suite, in a command of the form:

```
perfcheck suite:test_number
```

For example, to use test 16 of the **signing** suite, run the command:

```
perfcheck signing:16
```

This command first produces some output containing information about operating environment. Then it produces additional output of the following form:

```
>>> Results: Signing
>>>     Test               Reps  Total time (s)  Mean time (s)  Std dev (s)  Latency (ms)
 Rate
>>>
(signatures/s)
>>> 16  DSA using SHA224   50    22.36           0.4472         7.3          11843.7858
2.2362
>>>      with 3072-bit p
>>>      and 224-bit q.
>>>
>>> ...
```

Output from test suites, such as that in the example, includes the following information:

Value	Description
Reps	This value shows the number of repetitions of the test that have been run. Generally, a larger number of repetitions produces more accurate data. You can specify the number of repetitions by setting the **--repetitions**=*REPS* option.
Total time	This value shows the total time in seconds taken to run all repetitions of the test.
Mean time	This value shows the average time in seconds taken to run for one repetition of the test. (This value is calculated by dividing the value of **Total time** by the value of **Reps**).
Std dev	This value shows the standard deviation in the **Mean time**. If this value is high, it indicates that a large number of repetitions are necessary in order for the throughput figure to be meaningful. You can suppress calculation of the standard deviation by running **perfcheck** with the **--no-std-dev** option; in some cases, setting **--no-std-dev** can increase throughput.
Latency	This value is a measure of how long in milliseconds a single command takes to travel from the host computer to the HSM and back to the host computer again.
Rate	This value is a measure of throughput. It is calculated by dividing the value of **Reps** by the value of **Total time**, and displayed in terms of a unit appropriate to the type of test (for example, for **signing** tests, the number of signatures performed each second).

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 hardserver queues the reply.
6. 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.

stattree: information utility

The **stattree** utility returns the statistics gathered by the hardserver and HSMs.

Usage

```
stattree [<node> [<node> [...]]]
```

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
```

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.
```

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 an nShield Edge. This tag holds statistics for the PCI connection between the HSM and the host computer.
ModuleSerialStats	This tag is for an nShield Edge only. It holds statistics for the serial connection between the HSM and the host computer.
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.

Category	Contains
`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.

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 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.
`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.

ID	Value
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 executing on one or more HSMs. When an HSM accepts a command from a client, this number decreases by 1 and **QOutstanding** increases by 1. Commands that are processed purely by the server are never included in this count.
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.

ID	Value
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.
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.
MemAllocKernel	The total amount of RAM allocated for kernel (that is, non-SEE) use in an HSM. This is principally used for the object store (keys, logical tokens, and similar) and for big-number buffers.
MemAllocUser	The total amount of RAM allocated for user-mode processes in the HSM (**0** for non-SEE use). This includes the size of the SEE Machine image, and the total heap space available to it. The HSM's kernel does not know (and does not report in the statistics) how much of the user-mode's heap is currently free, and how much is in use.

How data is affected when a module loses power and restarts

nCipher modules use standard RAM to store many kinds of data, and data stored in such RAM is lost in the event that a module loses power (either intentionally, because you turned off power to it, or accidentally because of a power failure.

Therefore, after restoring power to a module, you must reload any keys that had been loaded onto it before it lost power. After reloading, the **KeyID**s are different.

Likewise, after restoring power to a module, you must reload any cards that were loaded onto it before it lost power.

However, data stored in NVRAM is unaffected when a module loses power.

Note: If you are using multiple nCipher 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.

Appendix H: Hardserver configuration files

The default location of the hardserver configuration file is `%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.

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.

Note: 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.

General hardserver configuration settings

server_settings

The **server_settings** section defines the settings for the client hardserver you can modify while the hardserver is running.

Note: These flags are used by the **NFLOG_DETAIL** environment variable (see *Environment variables to control logging* on page 219).

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* on page 219. If the **NFAST_SERVERLOGLEVEL** environment variable is set, it overrides any **loglevel** value set in the configuration file. **Note:** **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.

Field	Description
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.
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 Connect mode changing using the **nopclearfail** utility is enabled. If set to **no**, mode changing using **nopclearfail** is disabled. **Note:** 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 Connect 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.

Field	Description
`enable_remote_upgrade`	If this field is set to **yes** (the default value) in the module configuration file, the nShield Connect 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* on page 219).

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.
`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.

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 nShield Connects, for which scaling is currently optimized, and disabled if using nShield Solos or any other local devices). 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.

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.

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.

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* on page 55.

The section contains the following fields:

Field	Description
unix_socket_name	This field is not used on Windows systems.
unix_privsocket_name	This field is not used on Windows systems.
nt_pipe_name	This field specifies the name of the pipe to use for non-privileged connections on Windows. An empty string specifies none. The default is **\\.\pipe\crypto**. If the **NFAST_SERVER** environment variable is set, it overrides any value set for **nt_pipe_name** in the hardserver configuration file.
nt_pipe_users	This field specifies a list of users who are allowed to issue non-privileged connections on Windows. If this field is empty (which is the default), any user can issue non-privileged connections.
nt_privpipe_name	This field specifies the name of the pipe to use for privileged connections on Windows. An empty string specifies none. The default is **\\.\pipe\privcrypto**. If the **NFAST_PRIVSERVER** environment variable is set, it overrides any value set for **nt_privpipe_name** in the hardserver configuration file.
nt_privpipe_users	This field specifies the name of the user who is allowed to issue privileged connections on Windows. If this field is empty (which is the default), any user can issue non-privileged connections.
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 firewall 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.

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 Connect, 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 Connect, 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 userdata from the RFS before loading on the remote module. This field will be ignored if set on client machine configurations. **Note:** 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.

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.
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 2.
remote_ip	This field specifies the IP address of the machine that hosts the slot to import.

Field	Description
`remote_port`	This field specifies the port for connecting to the nShield Connect.
`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.

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.

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.

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 Slot and the local slot becomes the specified slot number. This enables applications and utilities that only support slot 0 to use Remote Administration.

Field	Description
`esn`	ESN of the HSM to which the the mapping is applied.

Field	Description
`slot`	The slot number to be swapped with slot 0, so that: • Slot 0 refers to a Dynamic 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.

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 (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`	Maximum number of seconds that can pass without a response from the smartcard, 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.

Sections only in client configuration files

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 Connect.
`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.
`timelimit`	This field specifies the time interval in seconds between renegotiations of the session key. The value `0` means no limit. The default is `60*60*24` seconds (that is, one day).

Field	Description
`datalimit`	This field specifies the amount of data in bytes to encrypt with a given session key before renegotiating a new one. The value `0` means no limit. The default is `1024*1024*8` bits (that is, 8 megabytes). This field applies to Triple DES session keys only. AES session keys heed the `timelimit` field but not the `datalimit` field.
`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.

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 remote server against which to synchronize.
`remote_port`	This field specifies the port for connecting to the nShield Connect.
`use_kneti`	Setting this option to `yes` requires the client to use an authenticated channel to communicate with the RFS server.
`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_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.

Field	Description
`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**.
`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 Unix convention for transfers.

Note: 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.

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 Remote Administration Service. The default is 9005.

Appendix I: Cryptographic algorithms

Symmetric algorithms

In the following tables, the column labelled **nfkmverify** documents the implemented behavior of the **nfkmverify** command-line utility (and its underlying library). An entry *N* in this column means that **nfkmverify** always rejects the algorithm, while **Y** means that **nfkmverify** always accepts the algorithm. A constraint on the number of bits means that **nfkmverify** only accepts keys of at least that many bits.

Symmetric Algorithms							
Algorithm	FIPS approval	Key length in bits	Use for ...				nfkmverify
			Authorization	Digital Signatures	Encryption	Other	
AES	Y	128, 192, 256	Y		Y		Y
Arcfour		Any			Y		N
ARIA	N	128, 192, 256			Y		
Camellia	N	128, 192, 256			Y		
CAST 6		up to 256	Y		Y		N
DES		56	Y		Y		N
MD5 HMAC		8 to 2048	Y				N
RIPEMD160 HMAC		8 to 2048	Y				≥80 bits
SEED		128	Y		Y		N
SHA-1 HMAC	Y	8 to 2048	Y				≥80 bits
SHA-224 HMAC	Y	8 to 2048	Y				≥80 bits
SHA-256 HMAC	Y	8 to 2048	Y				≥80 bits
SHA-384 HMAC	Y	8 to 2048	Y				≥80 bits
SHA-512 HMAC	Y	8 to 2048	Y				≥80 bits
Tiger HMAC		8 to 2048	Y				≥80 bits
Triple DES	Y	112, 168	Y		Y		Y

Asymmetric algorithms

Asymmetric Algorithms							
Algorithm	FIPS approval	Key length in bits	Use for ...				nfkmverify
			Authorization	Digital Signatures	Encryption	Other	
RSA	Y	≥512	Y	Y	Y		≥1024 bits
Diffie-Hellman	Y	≥1024				Key exchange	≥1024 bits
DSA	Y	512 to 3072	Y	Y			≥1024 bits
El-Gamal		≥1024			Y		
KCDSA	N	≥1024	Y	Y			≥1024 bits
ECDSA	Y	≥160	Y	Y			all named/custom curves with cofactor ≤4 and order 160
ECDH	Y	≥160				Key exchange	all named/custom curves with cofactor ≤4 and order 160

Note: The RSA algorithm is based on the factorization of integers.

Note: The Diffie-Hellman, DSA, El-Gamal, and KCDSA algorithms are all based on discrete logarithms in a multiplicative group of a finite field.

FIPS information

The latest guidance from the National Institute of Standards and Technology (NIST) is that

- A module is only operating in FIPS mode when it uses NIST-approved algorithms
- A module operating at FIPS 140-2 level 3 must not only indicate whether it is in FIPS mode but also actively prevent use of algorithms not approved by NIST in FIPS-approved mode.

When a module is initialized into FIPS 140-2 level 3 mode, you are only offered NIST-approved algorithms. If you have a Security World created to comply with FIPS 140-2 level 3 and have any protocols that use algorithms not approved by NIST, you must either migrate to a FIPS 140-2 level 2 Security World or change your protocols. If you have a Security World created to comply with FIPS 140-2 level 3 and have existing long-term keys for unapproved algorithms, then these keys cannot be used with the current firmware. In such a case, we recommend that you either migrate your Security World to a FIPS 140-2 level 2 Security World or replace these keys with approved keys before upgrading to the current firmware.

These changes do not affect Security Worlds that were created to comply with FIPS 140-2 level 2, nor do they affect systems that use the nShield module to protect long-term keys but perform encryption with session keys on the host (as is the case with the nShield MSCAPI, CHIL, and PKCS #11 libraries).

The NIST-approved algorithms that nShield modules offer are:

- DSA
- ECDSA
- ECDH
- RSA
- Diffie-Hellman
- Triple DES
- AES
- SHA-1
- SHA-224
- SHA-256
- SHA-384
- SHA-512
- TLS key derivation
- HMAC (SHA-1, SHA-224, SHA-256, SHA-384, SHA-512)

The algorithms *not* approved by NIST that nShield modules offer are:

- KCDSA
- ARIA
- Camellia
- Arc Four
- CAST 6
- DES
- SEED
- SHA-160
- MD2
- MD5
- RIPEMD 160
- HMAC (MD5, RIPEMD160)

Appendix J: 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.

Note: For information about generating keys with the **generatekey** utility, see *Generating keys using the command line* on page 177 in *Working with keys* on page 177.

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** • **HMACSHA1** • **RSA** • **Rijndael** (**AES**) 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 140-2 level 3, or if you do not set the **--no-verify** option.

Parameter	Description
embed	Specifying the **embed** application type generates a key for use with CHIL applications that: • Do not support **hwcrhk** key storage • Have a key importation facility capable of reading PEM-format RSA key files. Specify the **hwcrhk** application type for CHIL applications that support **hwcrhk** key storage. You can use a key of the **embed** application type like a PEM-format RSA/DSA key file, even though it is really a specially encoded reference to a key stored in %NFAST_KMDATA%\local. This allows you to use an **embed** key when integrating with applications that normally require software RSA keys. For example, you can supply an **embed** key to the patched version of OpenSSL we have provided so that it uses the module to access the key rather than using its own built-in RSA operations.
hwcrhk	Specifying the **hwcrhk** application type generates a key for Cryptographic Hardware Interface Library (CHIL) applications that do not require **embed** keys. Only **RSA**, **DSA**, and **DH** key types are supported
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.

Key properties (*NAME=VALUE*)

The *NAME=VALUE* syntax is used to specify the properties of the key being generated.

Note: 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.

Option	Description
blobsavefile	When using the **custom** application type, the *VALUE* for **blobsavefile** specifies a file name of the form *FILENAME*.*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: NISTP192, NISTP224, NISTP256, NISTP384, NISTP521, NISTB163, NISTB233, NISTB283, NISTB409, NISTB571, NISTK163, NISTK233, NISTK283, NISTK409, NISTK571, ANSIB163v1, ANSIB191v1, and SECP160r1.
embedconvfile	The *VALUE* for **embedconvfile** specifies the name of the PEM file that contains the RSA key to be converted.
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 filename 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.

Option	Description
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** or **hwcrhk**, this is the key identifier to use (the exact identifier for **simple**, for **hwcrhk** the key type is implicitly included). For other application types, keys are assigned an automatically generated identifier and accessed by means of some application-specific name.
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.
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). **Note:** You can also specify a module by setting the **--module** option.
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 *%NFAST_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.

Option	Description
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.)
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.
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 information on available key types and sizes, see *Cryptographic algorithms* on page 258. 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 exactly 1024 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* on page 258. The **--verify** option limits the available 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.

Option	Description
`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.

Available key properties by action/application

The following table shows which actions (generate, import, and retarget) and applications are applicable to the different *NAME* options:

Property	Action			Application								
	generate	import	retarget	custom	embed	hwcrhk	pkcs11	seeconf	seeinteg	seessl	simple	kpm
`alias`	X	X	X									
`blobsavefile`	X	X	X	X								
`cardset`	X	X		X	X	X	X				X	X
`certreq`							X					
`checks`	X			X	X	X	X				X	X
`curve`	X			X	X	X	X	X	X		X	
`embedconvfile`		X			X							
`embedsavefile`	X	X	X		X		X					
`from-application`			X	X	X	X	X				X	X
`from-ident`			X	X	X	X	X				X	X
`hexdata`		X		X	X	X	X				X	
`ident`	X	X				X					X	X
`keystore`	X	X	X									
`keystorepass`	X	X	X									
`module`	X	X		X	X	X	X			X	X	X
`nvram`	X	X		X	X	X	X				X	
`paramsreadfile`	X			X	X	X	X	X	X		X	
`pemreadfile`		X		X		X					X	X
`plainname`	X	X	X	X	X		X	X	X	X	X	X
`protect`	X	X		X	X	X	X	X	X	X	X	X
`pubexp`	X			X	X	X	X				X	X
`qsize`	X			X	X	X	X				X	X
`recovery`	X	X		X	X	X	X	X	X		X	X

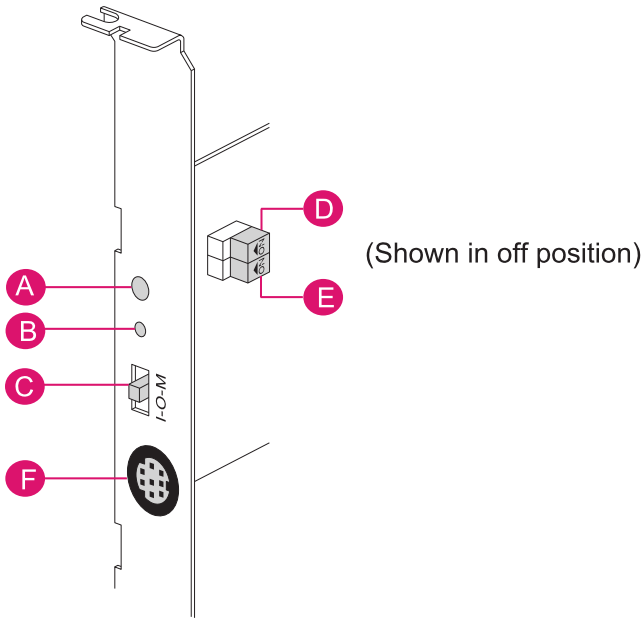
Property	Action			Application								
	generate	import	retarget	custom	embed	hwcrhk	pkcs11	seeconf	seeinteg	seessl	simple	kpm
seeintegname				X							X	
selfcert							X					
size	X			X	X	X	X	X	X	X	X	X
strict	X			X	X	X	X				X	
type	X			X	X	X	X	X	X	X	X	X
x509country	X	X	X		X							X
x509dnscommon	X	X	X		X							X
x509email	X	X	X		X							X
x509locality	X	X	X		X							X
x509org	X	X	X		X							X
x509orgunit	X	X	X		X							X
x509province	X	X	X		X							X
xsize	X			X	X	X	X				X	

Appendix K: Checking and changing the mode on an nShield Solo module

This appendix tells you how to check and change the mode on an nShield Solo module. You must change the mode to perform certain maintenance and configuration tasks.

nShield Solo back panel and jumper switches

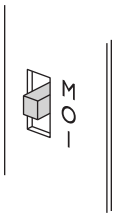
Figure 3. nShield Solo back panel and jumper switches



Label	Description
A	Status LED
B	Recessed reset button
C	Physical mode switch
D	Physical mode override jumper switch, in the **off** position. When set to **on**, the mode switch (C) is deactivated. See *Override switches* on page 271 for further information.
E	Remote Administration override jumper switch, in the **off** position. When set to **on**, remote mode switching is disabled. See *Override switches* on page 271 for further information.
F	Smart card connector, 8 Pole Female Mini-DIN connector on nShield PCIe and Solo XC module.

Physical mode switch

Figure 4. Physical mode switch



The physical mode switch on the back panel of an nShield Solo, as shown in Figure 4 and as 'C' in Figure 3 on page 268 enables you to select the mode on the module itself.

Available modes

The physical mode switch can be set to one of three positions:

- **Maintenance**
  - Sets the module to start in pre-maintenance mode
  - Allows you to upgrade the firmware of the module
- **Operational**
  - The default setting for day-to-day use
- **Initialization**
  - Sets the module to start in pre-initialization mode
  - Allows you to use the module to create a Security World or add the module to an existing one

Once you have selected a mode, the module needs to be reset before the mode is actually changed. See *Changing the mode* on page 271 for more about using the physical mode switch and resetting the module.

Note: If the Physical mode override jumper switch ('D' in Figure 3 on page 268) is set to **on**, the mode is set to Operational (O) and you cannot change it using the physical mode switch. See *Override switches* on page 271 for more about the Physical mode override jumper switch. You may, however, still be able to change the mode using the commanded mode switch. See *Remote mode switch* on page 269

Remote mode switch

The Remote mode switch enables you to change the mode of an nShield Solo from a computer using the `nopclearfail` command, without accessing the back panel of the module.

Available commands

You can use the following commands to change the mode of a module:

Command	Resulting mode
`nopclearfail --maintenance |-M`	Pre-maintenance

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

Limitations

A privileged user can only change the mode of a nShield Solo using the remote mode switch according to the following:

- The physical mode switch must be set to Operational (O) to be able to use the remote mode switch to change the mode.
  - If the module is physically set to either Maintenance (M) or Initialization (I), the remote mode switch has no effect, once the module has been reset following the `nopclearfail` command.
- If the physical mode override jumper switch ('D' in Figure 3 on page 268) is set to **on**, the module behaves as if the physical mode switch is set to Operational (O) and the remote mode switch can be used to change the mode.
- If the remote mode override jumper switch ('E' in Figure 3 on page 268) is set to **on**, the remote mode switch cannot be used.

The following table summarizes the resulting module modes when using the remote mode switch, taking into account the physical mode switch and physical mode override jumper switch settings.

Command	Physical jumper off (D)			Physical jumper on (D)
	Physical mode switch position			
	M	O	I	
`nopclearfail --maintenance |-M`	M	M	I	M
`nopclearfail --operational | -O`	M	O	I	O
`nopclearfail --initialization | -I`	M	I	I	I

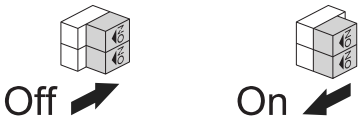
For you to be able to use the remote mode switch, the nShield Solo must be running 2.61.2 firmware or later. Otherwise the module responds with:

```
Module 1, command ClearUnitEx: HostDeviceDriverNotSupported
  -- device driver does not support software mode changes
```

See *Changing the mode* on page 271 for more about using the remote mode switch. See *Override switches* for more about the remote mode override jumper switch.

Override switches

Figure 5. nShield Solo override switches



As shown in Figure 3 on page 268

- Switch 'D', the physical mode override jumper switch, deactivates the physical mode switch
- Switch 'E', the command mode override jumper switch, deactivates the commanded mode switch

See the *Installation Guide* for more about accessing and setting a mode override jumper switch to **off** or **on**.

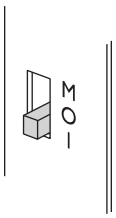
Changing the mode

Putting a module into pre-initialization mode using the physical mode switch

Do the following:

1. Switch the physical mode switch on the back panel of the module to the initialization (I) position, as shown below:

   Figure 6. Mode switch set to initialization

2. Reset the module by doing one of the following:
   - Press the **`Recessed reset button`** ('B' in Figure 3 on page 268or:
   - Run the **`nopclearfail --clear --all`** command.

The module performs self-tests, during which the Status LED is lit continuously.

Note: If the Status LED remains on continuously for more than a minute, the module self tests have resulted in a terminal failure. Contact Support. See *Chapter 1: Introduction* on page 14 for Support contact details.

When the self-tests are complete, the unit normally enters pre-initialization mode. In this mode, the Status LED flashes a series of single short pulses.

See *Status indications* on page 274 for more about Status LED codes.

You can use the **`enquiry`** command-line utility to check that the module is in the pre-initialization mode.

After the module has been put into pre-initialization mode, it is ready to be initialized. It enters *initialization mode* when it receives an **`initialization`** command (for example, when you run the **`new-world`** command-line utility).

Putting a module into pre-initialization mode using the commanded mode switch

Note: See *Limitations* on page 270 for more about the conditions that are required to use the commanded mode switch.

Do the following:

1. Run the **`nopclearfail --initialization | -I`** command.
   When finished, the system responds with **`OK`**.
   Note: The system responds with **`OK`**, regardless of whether the module has been changed to the pre-initialization mode or not. To confirm that state of the module, do the following:
2. Run the **`enquiry`** command.
   The `mode` line of the `Module` section displays the current mode.

See *Appendix K: Checking and changing the mode on an nShield Solo module* on page 268 for more about resolving commanded mode errors.

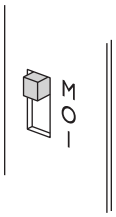
Putting a module into pre-maintenance mode using the physical mode switch

Only put a module into pre-maintenance mode if you need to upgrade module firmware.

Do the following:

1. Switch the physical mode switch on the module's back panel to the maintenance (M) position, as shown below:

   Figure 7. Mode switch set to maintenance

2. Reset the module by doing one of the following:
   - Press the **`Recessed reset button`** ('B' in Figure 3 on page 268or:
   - Run the **`nopclearfail --clear --all`** command.

The module performs self-tests, during which the Status LED is lit continuously.

Note: If the Status LED remains on continuously for more than a minute, the module self tests have resulted in a terminal failure. Contact Support. See *Chapter 1: Introduction* on page 14 for Support contact details.

When the self-tests are complete, the unit normally enters pre-maintenance mode. In this mode, the Status LED flashes a series of long pulses.

See *Status indications* on page 274for more about Status LED codes.

You can use the **enquiry** command-line utility to check that the module is in the pre-maintenance mode.

After the module has been put into pre-maintenance mode, it is ready for maintenance. It enters *maintenance mode* when it receives a **Maintenance** command (for example, when you run the **loadrom** command-line utility).

Putting a module into pre-maintenance mode using the commanded mode switch

Note: See *Limitations* on page 270 for more about the conditions that are required to use the commanded mode switch.

Do the following:

1. Run the**nopclearfail --maintenance | -M** command.
   When finished, the system responds with **OK**.
   Note: The system responds with **OK**, regardless of whether the module has been changed to the pre-maintenance mode or not. To confirm that state of the module, do the following:
2. Run the **enquiry** command.
   The mode line of the Module section displays the current mode.

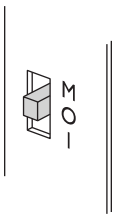
See *Appendix K: Checking and changing the mode on an nShield Solo module* on page 268 for more about resolving commanded mode errors.

Putting a module into operational mode using the physical mode switch

Do the following:

1. Switch the physical mode switch on the module's back panel to the operational (O) position, as shown below:

Figure 8. Mode switch set to operational



2. Reset the module by doing one of the following:
   - Press the **Recessed reset button** ('B' in Figure 3 on page 268or:
   - Run the **nopclearfail --clear --all** command.

The module performs self-tests, during which the Status LED is lit continuously.

Note: If the Status LED remains on continuously for more than a minute, the module self tests have resulted in a terminal failure. Contact Support. See *Chapter 1: Introduction* on page 14 for Support contact details.

When the self-tests are complete, the unit normally enters operational mode and ready to accept commands.

In operational mode, the Status LED is mainly on, but blinks off briefly at regular intervals.

See *Status indications* on page 274for more about Status LED codes.

Putting a module into operational mode using the commanded mode switch

Note: See *Limitations* on page 270 for more about the conditions that are required to use the commanded mode switch.

Do the following:

1. Run the`nopclearfail --operational | -o` command.
   When finished, the system responds with `OK`.
   Note: The system responds with `OK`, regardless of whether the module has been changed to the pre-maintenance mode or not. To confirm that state of the module, do the following:
2. Run the `enquiry` command.
   The `mode` line of the `Module` section displays the current mode.

See *Appendix K: Checking and changing the mode on an nShield Solo module* on page 268 for more about resolving commanded mode errors.

Status indications

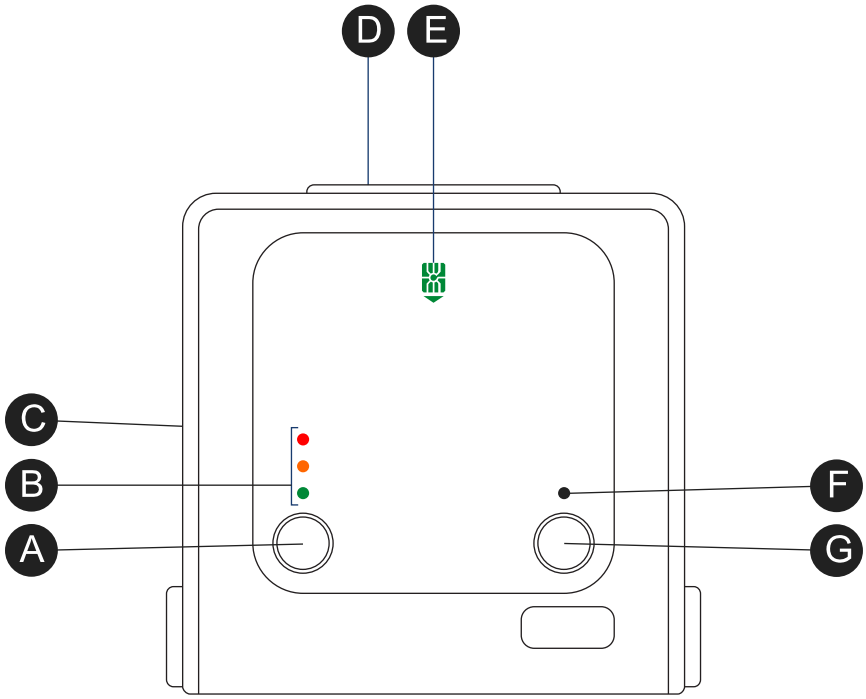
The following table explains the codes displayed by the Status LED.

LED	Mode	Reason
Mainly on but regularly blinks off (The exact timing depends on the nShield module. The longer the LED stays on the less the load. At 100% load the LED is off for as long as it is on.)	Operational	The Mode switch is in the operational position or the Mode override jumper switch is on. See the *nShield Solo Installation Guide* for more about accessing the Mode override jumper switch and setting it to off.
Emits repeated short flashes	Pre-initialization	The Mode switch is in the initialization position.
Emits repeated long flashes	Pre-maintenance	The Mode switch is in the maintenance position.
Flashes the Morse SOS pattern followed by a code	Error	The module has encountered an unrecoverable error. See *Appendix O: Morse code error messages* on page 315for more about these errors.

Appendix L: Checking and changing the mode on an nShield Edge

This appendix tells you how to check and change the mode on an nShield Edge. You must change the mode to perform certain maintenance and configuration tasks.

The Mode LEDs on an nShield Edge show the current or selected mode. The Status LED shows the status of the device.



A	Mode button	Selects a mode—the mode changes only when you press the **Clear** button.
B	Mode LEDs	Shows the current mode or selected mode.
C	B type USB port	For connecting the device to the computer.
D	Card slot	For inserting the required smart card.
E	Card slot LED	Lights green when a smart card is inserted.
F	Status LED	Shows the status of the device.
G	Clear button	When pressed and held for several seconds, clears the device's memory and changes the selected mode.

Mode LEDs

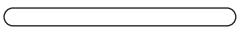
	Red	In Maintenance mode
	Red flashing	Maintenance mode selected
	Amber	In Initialization mode
	Amber flashing	Initialization mode selected
	Green	In Operational mode
	Green flashing	Operational mode selected

You mainly use the device in Operational (O) mode. You must put it into Initialization (I) mode when creating a Security World on the device, while Maintenance (M) mode allows you to upgrade the firmware.

To change the mode:

1. Use the **Mode** button to highlight the required mode.
2. Within a few seconds, press and hold the **Clear** button for a couple of seconds.
   If the mode changes, the new mode's LED stops flashing and remains lit. The Status LED might flash irregularly for a few seconds and then flashes regularly when the device is ready.
   Otherwise, the device remains in the current mode, with the appropriate mode LED lit.

Status LED

	Long blue flash	In Operational mode
	Short blue flash	In Maintenance or Initialization mode
	Off	No power
	Irregular flash	Changing mode or processing data

If the Status LED flashes irregularly and the device is unresponsive for more than a few minutes, the device has encountered an error. Disconnect the device, wait a few seconds, and then reconnect it.

Appendix M: Upgrading firmware

This appendix describes how to load an updated image file and associated firmware onto your nShield hardware security module.

Version Security Number (VSN)

All nShield firmware includes a Version Security Number (VSN). This number is increased whenever we improve the security of the firmware.

We supply several versions of the module firmware. You can always upgrade to firmware with an equal or higher VSN than that currently installed on your module..

You can never load firmware with a lower VSN than the currently installed firmware.

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. 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. Similarly, if you want to install a version with enhanced features without committing yourself to the upgrade, you can do so providing you upgrade only to firmware with a VSN equal to that currently installed on your module.

Firmware on the installation media

Your Security World Software 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.

Recognising firmware files

The firmware and monitor files are stored in subdirectories within the `firmware` directory on the installation media. The subdirectories are named by version number.

Firmware and monitor files for hardware modules have a **.nff** filename suffix. Monitor filenames have an **ldb** prefix. (Files that have a **.ftv** suffix are used for checking similarly named firmware files. They are not firmware files.)

Files for use with nShield Solo modules have **ncx3p** in the filename. Files for use with nShield Edge modules have **ncx1z** in the filename.

The VSN of a firmware file is incorporated into its filename and is denoted by a dash followed by the digits of the VSN. For example, **-24** means the VSN is 24.

To display information about a firmware file on the installation media, enter the following command:

```
loadrom --view E:\firmware\firmware_ver\firmware_file.nff
```

In this command, *E* is the drive letter of your installation media, *firmware_ver* is the firmware version number, and *firmware_file* is the file name.

Using new firmware

To use the new firmware, you must:

1. Install the latest software. See the Installation Guide for more information about software installation.
2. Install the latest firmware, as described below.

This appendix assumes that you have installed the hardserver as a service. This is the default installation procedure. See the Installation Guide for more information about software installation.

This chapter describes how to upgrade module firmware for nShield Solo and nShield Edge modules only. If you have another type of module (for example, an nShield Connect module), refer to the corresponding chapter in the User Guide.

Firmware installation overview

The process of installing or updating firmware on an nShield module depends on whether you need to upgrade the module's monitor.

Note: The Solo XC module does not have a separate monitor program, see *Upgrading firmware only* on page 280.

Each module has a monitor, which allows you to load firmware onto the module, refer to the *Release Notes* for monitor firmware versions.

To check the version number of the monitor on the module:

1. Log in to the host as an Administrator.
2. Put the module in Maintenance mode and reset the module.

See *Appendix K: Checking and changing the mode on an nShield Solo module* on page 268 or *Appendix L: Checking and changing the mode on an nShield Edge* on page 275 for more about changing the mode.

3. Run the **enquiry** command-line utility and check that the module is in the pre-maintenance state.

   The **Version** number shown is for the monitor.

If you need to upgrade both the monitor and firmware, you must use the **nfloadmon** utility; see *Upgrading both the monitor and firmware* on page 279.

If you need to upgrade the firmware only, you must use the **loadrom** utility; see *Upgrading firmware only* on page 280.

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.

Upgrading both the monitor and firmware

You must only use this procedure if you need to upgrade the monitor and firmware on an nShield module, for example, for Remote Administration functionality. If you only need to upgrade the firmware or have a Solo XC module, see *Upgrading firmware only* on page 280

Follow this procedure carefully. Do not interrupt power to the module during this upgrade process.

To upgrade the monitor and firmware on a module:

1. Log in to the host as an Administrator.
2. Run the command:

```
nfloadmon -m<module_number> --automode E:\firmware\monitor_ver\monitor_file.nff
E:\firmware\firmware_ver\firmware_file.nff
```

In this command, *<module_number>* is the module number (such as **-m2** for module 2).
--automode enables automated mode switching for nShield Solos, when supported in Remote Administration environments.

Note: Monitor version 2.60.1 is required to enable remote mode switching. Remote mode switching is not supported on an nShield Edge.

E is the drive letter of your installation media, *monitor_ver* is the monitor version number, *monitor_file* is the monitor file name, *firmware_ver* is the firmware version number, and *firmware_file* is the firmware file name.
For example:

```
nfloadmon -m2 --automode E:\firmware\2-50-16\ldb_ncx3p-24.nff E:\firmware\2-50-16\ncx3p-25.nff
```

The firmware files are signed and encrypted; you can load only the correct version for your module.

3. Confirm the version of the monitor and firmware.
4. Put the module into the different modes if and when prompted to do so.When supported, the mode of the nShield Solo changes automatically. Changing mode on the nShield Edge requires the **Clear** switch to be pressed.
   For information on changing the mode, see *Appendix K: Checking and changing the mode on an nShield Solo module* on page 268 or *Checking and changing the mode on an nShield Edge* on page 275.
5. When the **nfloadmon** utility has completed, put the module into Initialization mode (if prompted), and then initialize the module by running the command:

```
initunit
```

6. Put the module in Maintenance mode and reset the module.
7. Run the **enquiry** command to verify the module is in maintenance state and has the correct monitor version.
   In Maintenance mode, the **enquiry** command shows the version number of the monitor.
8. Put the module in Operational mode and reset the module.
9. Run the **enquiry** command to verify the module is in operational state and has the correct firmware version.
10. Log in to the host as normal.
    In Operational mode, the **enquiry** command shows the version number of the firmware.

Upgrading firmware only

To upgrade the firmware on a module:

1. Log in to the host as an Administrator.
2. Put the module in Maintenance mode and reset the module.
   Note: See *Appendix K: Checking and changing the mode on an nShield Solo module* on page 268 or *Appendix L: Checking and changing the mode on an nShield Edge* on page 275 for more about changing the mode.
3. If you are upgrading an nShield Solo or nShield Edge, run the **enquiry** command-line utility to check that the module is in the pre-maintenance state.
4. Insert the Security World Software installation media .
5. Load the new firmware by running the command:

```
loadrom -m<module_number> E:\firmware\firmware_ver\firmware_file.nff
```

In this command, *<module_number>* is the module number (such as **-m2** for module 2), *E* is the drive letter of your installation media, *firmware_ver* is the firmware version number, and *firmware_file* is the firmware file name.
For example:

```
loadrom -m2 E:\firmware\2-50-16\ncx3p-25.nff
```

The firmware files are signed and encrypted; you can load only the correct version for your module.

6. Put the module in Initialization mode and reset the module.
7. Initialize the module by running the command:

```
initunit
```

8. Put the module in Operational mode and reset the module.
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.
10. Log in to the host as normal.

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* on page 57.

If you are re-initializing the HSM into an existing Security World, see *Adding or restoring an HSM to the Security World* on page 77.

Appendix N: 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.

Note: 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 installations, is available from the NET-SNMP project Web site: http://net-snmp.sourceforge.net/.

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.

Installing and activating the SNMP agent

The SNMP agent can be installed and activated separately. After installing the SNMP components, an activation command can be issued.

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/`.

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 edtiting 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.

Starting the SNMP agent

To register the SNMP agent as a Windows service, enter the following command with administrative privileges:

```
%NFAST_HOME\bin\snmpd -register [params]
```

See *SNMP agent command-line* on page 312 for more information on additional parameters accepted by snmpd.

This installs the agent as a Windows Service but does not start it automatically.

Note: By default, the SNMP agent logs start-up and shut-down to the Event Viewer. More detailed logging can be configured by providing additional parameters when running the SNMP agent either from the command line or when registering as a service.

To unregister the SNMP agent as a windows service, enter the following command:

```
snmpd -unregister
```

The SNMP agent can be started and stopped from the services control panel or from the command prompt using:

```
net start "nCipher SNMP Agent"
net stop "nCipher SNMP Agent
```

Basic configuration

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 nCipher. 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* on page 285.

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?* on page 283.

Note: Make sure you protect access to the configuration files, since these contain information that defines the security parameters of the SNMP system.

The default location for the nShield SNMP configuration files is the *%NFAST_HOME%\etc\snmp* directory.

The **snmp.conf** and **snmpd.conf** files are not created automatically by the installation. Instead, example files (**example.snmp.conf** and **example.snmpd.conf**) are created in that location, which you can copy, rename (to **snmp.conf** and **snmpd.conf**), and edit with your desired configuration settings.

Note: The sample **snmpd.conf** file includes **agentuser** and **agentgroup** directives, however these are inoperative in Windows.

Note: You can override the default search path by setting the environment variable **SNMPCONFPATH** to a colon-separated (":") list of directories for which to search.

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.

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.

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.

Directive	Description
`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* on page 299.
`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* on page 298 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* on page 298 for more details.

Note: There may be a tolerance gap between the `memoryUsageOkThreshold` and the `memoryUsageHighThreshold` values.

Note: The timeouts should be set to values that achieve a balance between recieving up to date information whilst preventing excessive load.

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:

```
%NFAST_HOME%\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 con be performed with the `createUser` directive. See *USM users* on page 287. On initialisation 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.

Note: Do not modify the persistant `snmpd.conf` file while the agent is running. The file is only read on initialisation 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 configuration file.

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.

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 specification ":161" is not strictly necessary as this is the default port) . It will also listen on TCP port 1161 on all IPv4 interfaces.

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**
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: http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-63-2.pdf.

Note: 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* on page 287. The user can then be referenced in access control directives(s) after which it can be used.

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 user1
```

Or

```
rouser -s usm user1 auth .1
```

- Read-only user with access to the nCipher 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 **GETNEXT**) 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.

VACM configuration

The full flexibility of the VACM, for example allowing access to two or more distinct OID subtrees, or different access requirements for reading and writing, is available using four configuration 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 (**comsec2sec** 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 community 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 community 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 security 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
group 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 subtree, 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:

```
1.3.6.1.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
```

Trap Configuration

The distribution of SNMP supports SNMPv1, SNMPv2 and SNMPv3 traps. Control over these traps is defined with a number of directives:

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 hostname 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 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
```

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
```

Using the SNMP agent with a manager application

Note: The nShield SNMP monitoring agent provides MIB files that can be added to your (third-party) SNMP manager application.

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.

Note: 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).

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
```

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.

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* on page 287 .

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.

Note: Some traps can take up to five minutes to be received.

Note: 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

Memory usage monitoring

The HSM memory usage thresholds and memory usage traps provide a mechanism to monitor 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.

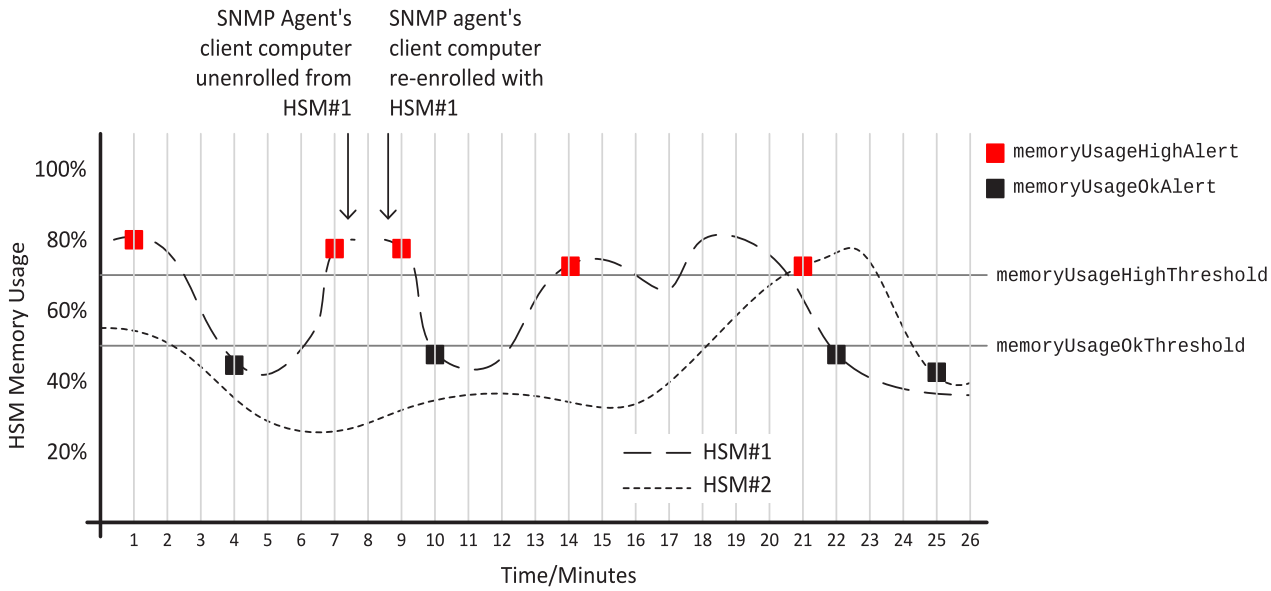
With memory usage monitoring enabled, there will be a **`memoryUsageOkAlert`** trap sent each time the memory usage for an HSM falls below the currently in-use **`memoryUsageOkThreshold`**.

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 (inclusive); 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 **`memoryUsageOkThreshold`** > **`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:



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.
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**.

Node name	R/W	Type	Remarks
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* on page 285).

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.
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 pass phrase 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.

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 pass phrase for an Operator card.
`adminQuorumNV`	R	Gauge32	The quorum required to access nonvolatile memory (NVRAM).
`adminQuorumRTC`	R	Gauge32	The quorum required to update the Real Time Clock.
`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.

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.

Node name	R/W	Type	Remarks
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	
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.

Node name	R/W	Type	Remarks
`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: outofrange	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)* on page 277.
`hashKNETI`	R	MHash	K_{NETI} hash, if present.
`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.

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**).
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.

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.
cardsetTimeout	R	Gauge32	Token time-out period, in seconds, or 0 if none.
cardsetGenTime	R	DisplayString	Generation time of card set.

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.

Node name	R/W	Type	Remarks
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.
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.

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	

Node name	R/W	Type	Remarks
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.
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).

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.

Note: 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 *Basic configuration* on page 284.

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	

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.
`cmdBytes`	R	Counter32	
`cmdErrors`	R	Counter32	Returned as for `cmdCount` all the different error states aggregated into one counter.

Node name	R/W	Type	Remarks
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).
currentTemp	R	DisplayString	Character representation of the current temperature value (SNMP does not provide for a floating-point type). Only available on non-XC variants.
maxTemp	R	DisplayString	Maximum temperature the module has ever reached. Only available on non-XC variants.
nvRAMInUse	R	Gauge32	
volatileRAMInUse	R	Gauge32	
tempSP	R	DisplayString	Only available on XC variants.
currentCPUTemp1	R	DisplayString	Only available on XC variants.
currentCPUTemp2	R	DisplayString	Only available on XC variants.
currentFanSpeed	R	DisplayString	Only available on XC variants.
currentFanDuty	R	DisplayString	Only available on XC variants.
CPUVoltage1	R	DisplayString	Only available on XC variants.
CPUVoltage2	R	DisplayString	Only available on XC variants.
CPUVoltage3	R	DisplayString	Only available on XC variants.
CPUVoltage4	R	DisplayString	Only available on XC variants.
CPUVoltage5	R	DisplayString	Only available on XC variants.
CPUVoltage6	R	DisplayString	Only available on XC variants.
CPUVoltage7	R	DisplayString	Only available on XC variants.
CPUVoltage8	R	DisplayString	
CPUVoltage9	R	DisplayString	
CPUVoltage10	R	DisplayString	
CPUVoltage11	R	DisplayString	

nShield Connect statistics table

The following table gives details of the nodes in the nShield Connect statistics table (`enterprises.nCipher.nC-series.statistics.netHSMStatsTable`):

Node name	R/W	Type	Remarks
`netHSMStatsIndex`	R	Integer	Table index (not module number).
`netHSMUptime`	R	Counter32	Host system uptime.
`netHSMCPUUsage`	R	Gauge32	CPU usage of unit host processor.
`netHSMUserMem`	R	Gauge32	Total user memory of unit.
`netHSMKernelMem`	R	Gauge32	Total kernel memory of unit.
`netHSMCurrentTemp`	R	DisplayString	Internal unit temperature (sensor 1).
`netHSMMaxTemp`	R	DisplayString	Maximum recorded temperature (sensor 1).
`netHSMCurrentTemp2`	R	DisplayString	Internal unit temperature (sensor 2).
`netHSMMaxTemp2`	R	DisplayString	Maximum recorded temperature (sensor 2).
`netHSMVoltage5V`	R	DisplayString	unit 5V power reading.
`netHSMVoltage12V`	R	DisplayString	unit 12V power reading.
`netHSMFan1Speed`	R	Gauge32	Fan 1 speed (RPM).
`netHSMFan2Speed`	R	Gauge32	Fan 2 speed (RPM).
`netHSMFan3Speed`	R	Gauge32	Fan 3 speed (RPM).
`netHSMIPAddress`	R	IpAddress	IP address of unit.
`netHSMDescription`	R	DisplayString	Textual description of module (for example, `Local module #3`).
`netHSMFan4Speed`	R	Gauge32	Fan 4 speed (RPM).
`netHSMVoltage3p3V`	R	DisplayString	3.3V Supply Rail Voltage
`netHSMCurrent3p3V`	R	DisplayString	3.3V Supply Rail Current
`netHSMCurrent5V`	R	DisplayString	5V Supply Rail Current
`netHSMCurrent12V`	R	DisplayString	12V Supply Rail Current
`netHSMVoltage5VSB`	R	DisplayString	5V Supply Rail Voltage (Standby)
`netHSMCurrent5VSB`	R	DisplayString	5V Supply Rail Current (Standby)
`netHSMTamperBattery1`	R	DisplayString	Voltage of Tamper Battery 1
`netHSMTamperBattery2`	R	DisplayString	Voltage of Tamper Battery 2
`netHSMPSUfailure`	R	TruthValue	Power Supply failure status

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.
`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

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

Node name	R/W	Type	Remarks
`connModuleStatsModuleIndex`	R	Integer	Index of the module entry
`connModuleStatsObjectCount`	R	Gauge32	The object count on this module for this connection

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)

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	

SNMP agent command-line

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.
-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, `LOGFILE=log/snmpd.log`).

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.
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.

Note: 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/.

Appendix O: Morse code error messages

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.

The blue Status LED flashes the Morse distress code (SOS: three short pulses, followed by three long pulses, followed by three short pulses). The Morse distress code is followed by one of the error codes listed in the tables shown in this guide.

For nShield Solos running firmware 2.61.2 and above, the error code listed in this chapter is also reported by the **`enquiry`** utility in the **`hardware status field`** of the **`Module`**.

Errors are a rare occurrence. If any module goes into the error state, except as a result of you issuing the **`Fail`** command, contact Support, and give full details of your set up and the error code.

Contact 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 type.

Reading Morse code

The following guidelines are useful when reading Morse 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 repeated series of letters (a Morse code word gap) is 7 times the duration of a dot.

The following table shows the error codes corresponding to numerals.

Numeral	Morse
1	. - - - -
2	. . - - -
3	. . . - -
4	. . . . -
5	. . . . .
6	- . . . .
7	- - . . .
8	- - - . .
9	- - - - .
0	- - - - -

Runtime library errors

Memory failures can occur if the module is exposed to excessive heat. If you experience these errors, check the ventilation around the module. The module generates considerable heat and, if not well ventilated, may be operating at too high a temperature, even if the rest of your server room is at an appropriate temperature.

The runtime library error codes could be caused by firmware bugs or by faulty hardware. If any of these errors is indicated, 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 I --- .-.. ..	SIGSTAK: out of stack space.
O L J --- .-.. .---	SIGFPE: unsupported arithmetic exception (such as division by 0).
O L K --- .-.. -.-	SIGOSERROR: runtime library internal error.
O L N --- .-.. -.	SIGFATALPANIC: error in error handling code.

Codes **OLD**, and **OLE** are more likely to indicate a hardware problem than a firmware problem.

To reset a unit that is in an error state, turn off the unit and then turn it on again.

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.

Note: In the following table, the symbol "#" stands for a given numeral's Morse code representation.

If any of these errors is indicated, contact support.

Code	Meaning
H L .... .-..	Dorris M48T37 NVRAM (or battery) failed
H B .... -...	Debug serial port initialization failed.
H C .... -.-.	Processing thread initialization failed.
HCP .... -.-. .--.	Card poll thread initialisation failed.
H D .... -..	Failure reading unique serial number.
H E	EEPROM failed on initialization.
HF .... ..-.	Starting up crypto offload.

Code	Meaning
H I	Interrupt controller initialization failed.
H M . . . . --	System hardware initialization failed.
H O . . . . ---	Token interface initialization failed.
H R . . . . .-.	Random number generator failed. **Note:** This code may also be generated if an attempt is made to downgrade firmware on a JackEC or Jack Lite to version 2.50.x or older.
HRS . . . . .-. . . .	RNG startup failed.
HRT . . . . .-. -	RNG selftest failed.
HRTP . . . . .-. - . --.	Periodic (scheduled daily) RNG selftest failed.
HRM . . . . .-. --	RNG data matched.
HRZ . . . . .-. --. .	Impossible RNG failure (match after PRNG).
HS	Unexpected error from SCSI controller or host interface initialization failed.
HV . . . . . .-	Environment sensors failed (for example, temperature sensor)
HCV . . . . -.-. . . .-	CPLD wrong version for PCI policing firmware.
H P P . . . . .--. . --.	PCI Interface Policing failure.
HST . . . . . . -	Speed test failed.
H H R . . . .-.	RTC hardware detection failed or random number generator detection failed.
H R H . . . . .-. . . .	RNG hardware failed during operation
KR -.- .-.	RSA selftest failed.
H M *n* . . . . -- #	DSP *n* failed self-test at start up.
H C *n* C A . . . . -.-. # -.-. .-	CPU *n* failed self-test; no memory for cached RAM test.
H C *n* C C . . . . -.-. # -.-. -.-.	CPU *n* failed self-test; CPU ID check failed.
H C *n* C F . . . . -.-. # -.-. . .-.	CPU *n* failed self-test; freeing memory for cached RAM test.
H C *n* C G . . . . -.-. # -.-. --.	CPU *n* failed self-test; setting up cached RAM test.
H C *n* C R . . . . -.-. # -.-. .-.	CPU *n* failed self-test; read error during cached RAM test.
H C *n* C V . . . . -.-. # -.-. . . .-	CPLD version number incorrect (Dorris).

Code						Meaning
H C *n* C W	. . . .	- . - .	#	- . - .	. - -	CPU *n* failed self-test; write error during cached RAM test.
H C *n* K A	. . . .	- . - .	#	- . -	. -	CPU n failed selftest - AES known-answer test.
H C *n* K B	. . . .	- . - .	#	- . -	- . . .	CPU n failed selftest - AES CMAC known-answer test.
H C *n* K C	. . . .	- . - .	#	- . -	- . - .	CPU n failed selftest - ECDSA known-answer test
H C *n* K E	. . . .	- . - .	#	- . -	.	CPU *n* failed self-test; DES known-answer test.
H C *n* K F	. . . .	- . - .	#	- . -	. . - .	CPU *n* failed self-test; Triple-DES known-answer test.
H C *n* K H	. . . .	- . - .	#	- . -	. . . .	CPU *n* failed self-test; SHA-1 known-answer test.
H C *n* K I	. . . .	- . - .	#	- . -	. .	CPU n failed selftest - HMAC-SHA512 known-answer test.
H C *n* K J	. . . .	- . - .	#	- . -	. - - -	CPU n failed selftest - HMAC-SHA256 known-answer test.
H C *n* K M	. . . .	- . - .	#	- . -	- -	CPU *n* failed self-test; HMAC-SHA1 known-answer test.
H C *n* K N	. . . .	- . - .	#	- . -	- .	CPU n failed selftest - HMAC-SHA224 known-answer test.
H C *n* K P	. . . .	- . - .	#	- . -	. - - .	CPU n failed selftest - HMAC-SHA384 known-answer test.
H C *n* K R	. . . .	- . - .	#	- . -	. - .	CPU n failed selftest - RSA known-answer test
H C *n* K S	. . . .	- . - .	#	- . -	. . .	CPU *n* failed self-test; DSA known-answer test.
H C *n* L C	. . . .	- . - .	#	. - . .	- . - .	CPU *n* failed self-test; locking check.
H C *n* P S	. . . .	- . - .	#	. - - .	. . .	CPU *n* failed self-test; test terminated at start.
H C *n* RT	. . . .	- . - .	#	. - .	-	CPU n failed selftest - RTC check.
H C *n* S A	. . . .	- . - .	#	. . .	. - - .	CPU *n* failed self-test; no memory for uncached RAM test.
H C *n* S F	. . . .	- . - .	#	. . .	. . - .	CPU *n* failed self-test; freeing memory for uncached RAM test.

Code					Meaning
H C *n* S R	. . . .	- . - .	#	. . .	. - .	CPU *n* failed self-test; read error during uncached RAM test.
H C *n* S W	. . . .	- . - .	#	. . .	. - -	CPU *n* failed self-test; write error during uncached RAM test.
H C *n* T S	. . . .	- . - .	#	-	. . .	CPU *n* failed self-test; could not start test.

Maintenance mode errors

The following error codes indicate faults encountered when a module is in the maintenance mode.

Code			Meaning	Action
I D	. .	- . .	Copies of metadata do not match when trying to run image.	Contact Support.
I H	. .	. . . .	Bad metadata: hash mismatch.	Repeat firmware upgrade.
I I	. .	. .	Execution image does not match metadata.	Contact Support.
I L	. .	. - . .	Bad metadata: either bad length or bad metadata when running loadboot application.	Repeat firmware upgrade.
I M	. .	- -	Bad metadata: malformed ImageMetaData.	Repeat firmware upgrade.
I P	. .	. - - .	Bad metadata: bad padding.	Repeat firmware upgrade.
I R	. .	. - .	Bad metadata: extra bytes at end.	Repeat firmware upgrade.
I S	. .	. . .	Image entry point not found.	Contact Support.
I U	. .	. . -	Bad metadata: ROM blank.	Repeat firmware upgrade.
I X	. .	- . . -	Bad metadata: malformed header.	Repeat firmware upgrade.
J H	. - - -	. . . .	Both copies of metadata invalid.	Contact Support.
H Z E	. . . .	- - . .	.	Monitor checksum failed.	Contact Support.
K F E	- . -	. . - .	.	Flash sector erase failed.	Repeat firmware upgrade.
K F P	- . -	. . - .	. - - .	Flash sector program failed.	Repeat firmware upgrade.
M M D	- -	- -	- . .	No memory for download buffer.	Contact Support.

Note: For instructions on upgrading module firmware, see the appendix in the User Guide for your module type.

Operational mode errors

The following runtime library error codes could be caused by either bugs in the firmware or faulty hardware.

Code	Meaning	Action
D - . .	Fail command received.	Reset module by turning it off and then on again.
T -	Temperature of the module has exceeded the maximum allowable.	Restart your host computer, and improve module cooling.
G G G -- . -- . -- .	Failure when performing `ClearUnit` or `Fail` command.	Contact Support.

Note: To improve the cooling of your PCIe module, increase the distance between PCIe cards, and increase the airflow through your host computer.

Solo XC tamper event errors

The following error codes indicate a hard tamper event has occurred on a Solo XC module. The Solo XC will become non-operational if tamper event error is indicated.

Note: If a tamper event error occurs the Solo XC module must be destroyed or returned to nCipher.

Code	Meaning	Action
TT - -	Hard temperature tamper	Contact Support
VV . . . - . . . -	Hard voltage tamper	Contact Support
T .	Soft temperature tamper	Contact Support
V . . . -	Soft voltage tamper	Contact Support
B - . . .	Low battery voltage, <2.5V	Contact Support
HI2C . . . . --- - . - .	I2C Failure	Contact Support
WD0 . -- - . . -----	Watchdog 0 failure	Contact Support
WD1 . -- - . . . ----	Watchdog 1 failure	Contact Support
WD2 . -- - . . . . ---	Watchdog 2 failure	Contact Support
WD3 . -- - . . . . . --	Watchdog 3 failure	Contact Support

Appendix P: 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, nCipher 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.

Appendix Q: Application Performance Tuning

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 a Solo, Solo+, Connect or Connect+, around 40 outstanding jobs per HSM is a good target when using an application that is coded directly against the nCore API. When using higher-level APIs such as PKCS#11 or CNG, your application may benefit from increasing the job count further, e.g. to 60 or more outstanding jobs per HSM.

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.

Client Configuration

If your application is coded directly against nCore, you have a choice of sending multiple jobs asynchronously from a single client connection to the hardserver, or having multiple threads each with their own client connection to the hardserver with a single job sent synchronously in each. You can use the **`--threads`** parameter to the **`ncperftest`** utility to experiment with the performance impact of having more threads/connections with fewer jobs outstanding in each, or having fewer or just one thread/connection with more jobs outstanding in that connection.

When using higher-level APIs such as PKCS#11 or CNG, all cryptographic operations are synchronous, so larger numbers of threads must be used to increase the job count and make full use of HSM resources. These APIs automatically create a hardserver connection for each thread. If many HSMs are being used, a great many threads may be required to achieve best throughput. You can adjust the thread counts in the performance test tools for these APIs (e.g. **`cksigtest`** for PKCS#11 and **`cngsoak`** for CNG) to gauge how much concurrency is required for best throughput in your application.

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`** - Windows, Linux
- **`mtmalloc`** - Solaris

- **`MALLOCOPTIONS`** environment variable - AIX
- **`MallocNextGen`** - HP-UX.

On all systems except HP-UX, the **`hardserver`** is configured to use a scalable memory allocator by default, so only the client application requires special configuration. On HP-UX, the **`hardserver`** will automatically use the HP **`MallocNextGen`** memory allocator if it is installed. **`MallocNextGen`** is an optional package which can be downloaded free of charge from the HP website:

https://h20392.www2.hpe.com/portal/swdepot/displayProductInfo.do?productNumber=MallocNextGen

It is recommended that **`MallocNextGen`** be installed for best performance on HP-UX.

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. This has been observed to be especially the case on HP-UX, where you can execute your application using the **`rtsched`** utility to change the scheduling algorithm from the default.

File Descriptor Limits

On Linux/UNIX systems, large numbers of threads each with their own hardserver connection will require your application to 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.

Appendix R: Product returns

If you need to return your nShield product, contact Support for instructions:

https://help.ncipher.com

Glossary

Authorized Card List

Controls the use of Remote Administration cards. 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.

Access Control List (ACL)

An Access Control List is a set of information contained within a key that specifies what operations can be performed with the associated key object and what authorization is required to perform each of those operations.

Administrator Card Set (ACS)

Part of the Security World architecture, an Administrator Card Set (ACS) is a set of smart cards used to control access to Security World configuration, as well as recovery and replacement operations.

The Administrator Cards containing share in the logical tokens that protect the Security World keys, including $\mathbf{K}_{NSO}$, the key-recovery key, and the recovery authorization keys. Each card contains one share from each token. The ACS is created using the well-known module key so that it can be loaded onto any nShield module.

See also *Security World*, *Operator Card Set (OCS)*

Advanced Encryption Standard (AES)

The Advanced Encryption Standard (AES) is a block cipher adopted as an encryption standard by the US government and officially documented as US FIPS PUB 197 (FIPS 197). Originally only used for non-classified data, AES was also approved for use with for classified data in June 2003. Like its predecessor, the Data Encryption Standard (DES), AES has been analyzed extensively and is now widely used around the world.

Although AES is often referred to as *Rijndael* (the cipher having been submitted to the AES selection process under that name by its developers, Joan Daemen and Vincent Rijmen), these are not precisely the same cipher. Technically, Rijndael supports a larger range of block and key sizes (at any multiple of 32 bits, to a minimum of 128 bits and a maximum of 256 bits); AES has a fixed block size of 128 bits and only supports key sizes of 128, 192, or 256 bits.

See also *Data Encryption Standard (DES)* on page 326

CAST

CAST is a symmetric encryption algorithm with a 64-bit block size and a key size of between 40 bits to 128 bits (but only in 8-bit increments).

client identifier: R_{SC}

This notation represents an arbitrary number used to identify a client. In the nCore API, all client identifiers are 20 bytes long.

Data Encryption Standard (DES)

The Data Encryption Standard (DES) is a symmetric cipher approved by NIST for use with US Government messages that are Secure but not Classified. The implementation of DES used in the module has been validated by NIST. DES uses a 64-bit block and a 56-bit key. DES keys are padded to 64 bits with 8 parity bits.

See also *Triple DES* on page 331, *Advanced Encryption Standard (AES)* on page 325

Diffie-Hellman

The Diffie-Hellman algorithm was the first commercially published public key algorithm. The Diffie-Hellman algorithm can only be used for key exchange.

Digital Signature Algorithm (DSA)

Also known as the Digital Signature Standard (DSS), the Digital Signature Algorithm (DSA) is a digital signature mechanism approved by NIST for use with US Government messages that are Secure but not Classified. The implementation of the DSA used by nShield modules has been validated by NIST as complying with FIPS 186.

Digital Signature Standard (DSS)

See *Digital Signature Algorithm (DSA)* on page 326

ECDH

A variant of the Diffie-Hellman anonymous key agreement protocol which uses elliptic curve cryptography.

See also *Diffie-Hellman* on page 326.

ECDSA

Elliptic Curve DSA: a variant of the Digital Signature Algorithm (DSA) which uses elliptic curve cryptography.

See also *Digital Signature Algorithm (DSA)* on page 326, *Diffie-Hellman* on page 326.

encryption: $\{A\}_B$

This notation indicates the result of **A** encrypted with key **B**.

Federal Information Processing Standards (FIPS)

The Federal Information Processing Standards (FIPS) were developed by the United States federal government for use by non-military government agencies and government contractors. FIPS 140 is a

series of publications intended to coordinate the requirements and standards for cryptographic security modules, including both their hardware and software components.

All Security Worlds are compliant with FIPS 140-2. By default, Security Worlds are created to comply with FIPS 140-2 at level 2, but those customers who have a regulatory requirement for compliance with FIPS 140-2 at level 3 can also choose to create a Security World that meets those requirements.

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

hardserver

The hardserver software controls communication between applications and nShield modules, which may be installed locally or remotely. It runs as a service on the host computer. The behavior of the hardserver is controlled by the settings in the hardserver configuration file.

hardware security module (HSM)

A hardware security module (commonly referred to as an HSM) is a hardware device used to hold cryptographic keys and software securely.

hash: H(X)

This notation indicates a fixed length result that can be obtained from a variable length input and that can be used to identify the input without revealing any other information about it. The nShield module uses the Secure Hash Algorithm (SHA-1) for its internal security.

identifier hash: $H_{ID}(X)$

An identifier hash is a hash that uniquely identifies a given object (for example, a key) without revealing the data within that object. The module calculates the identity hash of an object by hashing together the object type and the key material. The identity hash has the following properties:

H_{ID} is not modified by any operations on the key (for example, altering the ACL, the application data field, or other modes and flags)

H_{ID} is the same for both public and private halves of a key pair.

Unique data is added to the hash so that a H_{ID} is most unlikely to be the same as any other hash value that might be derived from the key material.

key blob

A key blob is a key object with its ACL and application data encrypted by a module key, a logical token, or a recovery key. Key blobs are used for the long-term storage of keys. Blobs are cryptographically secure; they can be stored on the host computer's hard disk and are only readable by units that have access to the same module key.

See also *Access Control List (ACL)*.

key object: K_A

This is a key object to be kept securely by the module. A key object may be a private key, a public counterpart to a private key, a key for a symmetric cipher (MAC or some other symmetric algorithm), or an arbitrary block of data. Applications can use this last type to allow the module to protect any other data items in the same way that it protects cryptographic keys. Each key object is stored with an ACL and a 20-byte data block that the application can use to hold any relevant information.

KeyID: ID_{KA}

When a key object KA is loaded within the module's RAM, it is given a short identifier or handle that is notated as $\mathbf{ID}_{KA}$. This is a transient identifier, not to be confused with the key hash **HID(KA)**.

logical token: K_T

A logical token is a key used to protect key blobs. A logical token is generated on the nShield module and never revealed, except as shares.

MAC: MAC_{KC}

This notation indicates a MAC (Message Authentication Code) created using key **KC**.

module

See *hardware security module (HSM)*.

module key: K_M

A module key is a cryptographic key generated by each nShield module at the time of initialization and stored within the module. It is used to wrap key blobs and key fragments for tokens. Module keys can be shared across several modules to create a larger Security World.

All modules include two module keys:

- module key zero $\mathbf{K}_{M0}$, a module key generated when the module is initialized and never revealed outside the module.
- **null**, or well-known module key $\mathbf{K}_{MWK}$.

You can program extra module keys into a module.

See also: *Security World*, *hardware security module (HSM)*.

module signing key: K_{ML}

The module signing key is the module's public key. It is used to issue certificates signed by the module. Each module generates its own unique $\mathbf{K}_{ML}$ and $\mathbf{K}_{ML}^{-1}$ values when it is initialized. The private half of this key pair, $\mathbf{K}_{ML}^{-1}$, is never revealed outside the module.

nShield master feature enable key K_{SA}

Certain features of the module firmware are available as options. These features must be purchased separately from nCipher. To use a feature on a specific module, you require a certificate from nCipher

signed by K_{SA}. These certificates include the electronic serial number for the module.

nShield Remote Administration Card

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. You must use nShield Remote Administration Cards with Remote Administration.

nShield Security Officer's key: K_{NSO}^{-1}

The notation K_{NSO}^{-1} indicates the Security Officer's signing key. This key is usually a key to a public-key signature algorithm.

nShield Trusted Verification Device

A smart card reader that allows the card holder to securely confirm the Electronic Serial Number (ESN) of the HSM to which they want to connect, using the display of the device. nCipher supplies and the nShield Trusted Verification Device and recommends its use with Remote Administration.

null module key: K_{MWK}

The null module key is used to create tokens that can be loaded onto any module. Such tokens are required for recovery schemes. The null module key is a Triple DES key of a value 01010101. As this value is well known, this module key does not have any security. Key blobs cannot be made directly under the null module key. To make a blob under a token protected by the null module key, the key must have the ACL entry **AllowNullKMToken**.

Operator Card Set (OCS

Part of the Security World architecture, an Operator Card Set (OCS is a set of smart cards containing shares of the logical tokens that is used to control access to application keys within a Security World. OCSs are protected using the Security World key, and therefore they cannot be used outside the Security World.

See also: *Security World*, *Administrator Card Set (ACS)*.

Recovery key: K_{RA}

The recovery key is the public key of the key recovery agent.

Remote access solution

The remote access solution, such as SSH or a remote desktop application, which is used as standard by your organization. Enables you to to carry out Security World administrative tasks from a different location to that of an nShield Connect or nShield Solo.

For example, the remote access solution is used to run security world utilities remotely and to enter passphrases.

Note: nCipher does not provide this software.

Remote Administration

An optional Security World feature that enables Remote Administration card holders to present their cards to an HSM located elsewhere. For example, the card holder may be in an office, while the HSM is in a data center. Remote Administration supports the ACS, as well as persistent and non-persistent OCS cards, and allows all smart card operations to be carried out, apart from loading feature certificates.

nShield Remote Administration Client

A GUI or command-line interface that enables you to select an HSM located elsewhere from a list provided by the Remote Administration Service, and associate a card reader attached to your computer with the HSM. Resides on your local Windows or Linux-based computer.

Remote Administration Service

Enables secure communications between an nShield Remote Administration Card and the hardserver that is connected to the appropriate HSM. Listens for incoming connection requests from nShield Remote Administration Clients. Supplies a list of available HSMs to the nShield Remote Administration Client and maintains an association between the relevant card reader and the HSM.

Dynamic Slot

Virtual card slots that can be associated with a card reader connected to a remote computer. Remote Administration Slots are in addition to the local slot of an HSM and any soft card slot that may be available. HSMs have to be configured to support between zero (default) and 16 Remote Administration Slots.

Rijndael

See *Advanced Encryption Standard (AES)* on page 325

salt: X

The random value, or salt, is used in some commands to discourage brute force searching for keys.

Security World

The Security World technology provides an infrastructure for secure lifecycle management of keys. A Security World consists of at least one hardware security module, some cryptographic key and certificate data encrypted by a Security World key and stored on at least one host computer, a set of Administrator Cards used to control access to Security World configuration, recovery and replacement operations, and optionally one or more sets of Operator Cards used to control access to application keys.

See also *Administrator Card Set (ACS)*, *Operator Card Set (OCS)*.

Security World key: K_{MSW}

The Security World key is the module key that is present on all modules in a Security World. Each Security World has a unique Security World key. This key is generated randomly when the Security World is created, and it is stored as a key blob protected by the ACS.

share: K_{Ti}

The notation K_{Ti} indicates a share of a logical token. Shares can be stored on smart cards or software tokens. Each share is encrypted under a separate share key.

share key: K_{Si}

A share key is a key used to protect an individual share in a token. Share keys are created from a Security World key, a pass phrase, and a salt value.

Standard nShield Cards

Smart cards used in the local slot of an HSM. Standard nShield cards are not supported for use with Remote Administration.

Standard card reader

A smart card reader for ISO/IEC 7816 compliant smart cards. nCipher recommends that standard smart card readers are only used with the nShield Remote Administration Client command-line utility, not the GUI.

Triple DES

Triple DES is a highly secure variant of the Data Encryption Standard (DES) algorithm in which the message is encrypted three times.

See also *Data Encryption Standard (DES)* on page 326, *Advanced Encryption Standard (AES)* on page 325.