

Redline Networks E|X Web I/O Accelerator  
**Supplemental Documentation**  
**Product Model: 3650-FIPS**  
**E|X Version 3.1**



**REDLINE**  
**NETWORKS**

Copyright © 2003, 2004 Redline Networks, Inc.

Redline Networks and The Redline Symbol are registered trademarks of Redline Networks, Inc. Deploy and Enjoy, E|X and T|X are trademarks of Redline Networks, Inc. All rights reserved. All other products and services mentioned in this publication are the trademarks, service marks, registered trademarks, or registered servicemarks of their respective owners.

Document Version: 2.1

Redline Networks  
655 Campbell Technology Parkway #250  
Campbell, CA 95008

+1 408.369.3800

www.RedlineNetworks.com

## Conventions

The following conventions were used in this manual:

Notation	Example	Meaning and Use
Courier typeface	<code>.ini</code> file	Code Listings, names of files, symbols, and directories, are shown in courier typeface.
Bold Courier typeface	<b>install</b>	In a command line, keywords are shown in bold, non-italic, Courier typeface. Enter them exactly as shown.
Italics	<i>Note:</i>	Notes about the subject are shown with a header in italics.
Bold Italics	<b><i>Important:</i></b>	Important information about the subject is shown with the header in bold Italics. This information should not be ignored.

When computer output listings are shown, an effort has been made not to break up the lines when at all possible. This is to improve the clarity of the printout; for this reason, some listings will be indented, and others will start at the left edge of the column.

# Table of Contents

Conventions .....	ii
<b>Table of Contents .....</b>	<b>iii</b>
<b>Chapter 1. Overview.....</b>	<b>1</b>
Basic Configuration .....	1
Changes to the Standard Configuration .....	2
<b>Chapter 2. The nCipher Utilities.....</b>	<b>5</b>
Help Options: .....	6
Detailed Utility Usage .....	7
checkmod.....	7
cryptest.....	8
des_kat .....	9
enquiry .....	10
floodtest .....	16
fwcheck.....	18
hardserver .....	19
initunit.....	20
killrecov .....	21
kmfile-dump .....	26
kptest.....	27
loadrom .....	29
mkaclx .....	30
modstate .....	31
ncdate .....	32
ncversions .....	33
new-world .....	34
nfkminfo .....	38
nfkmverify .....	43
nopclearfail .....	46
nvram-sw .....	47
pollbare .....	49
pubkey-find.....	50
racs.....	52
randchk .....	53
sigtest .....	54
slotinfo .....	56
stattree.....	57
with-nfast.....	63



# Chapter 1. Overview

---

The E|X Series Web I/O Accelerator FIPS (Federal Information Processing Standard) option consists of an nCipher 1600 PCI module installed in the Redline appliance, and the utilities needed to control it. This allows the E|X Series Web I/O Accelerator to achieve FIPS I-140 level 2 and level 3 compliance. FIPS 140 is the Federal Information Processing Standard that outlines security requirements for cryptographic modules.

To initialize the FIPS option, you need to initialize the nCipher module and create a new security world. This chapter outlines how to accomplish that.

## Basic Configuration

Use nCipher for the following steps to configure the security world and generate keys. This also makes it work with rl\_mux.

- Step 1.** Change the mode of the nCipher Module to 'I' (Initialization mode).
- Step 2.** Reset the module by pressing the “Clear button.” The module will perform some self tests.
- Step 3.** Confirm the mode of nCipher module #1 using “ncipher enquiry” command from the E|X command line. It should show module #1 mode as “pre-initialization”.
- Step 4.** Create a new security world using the command:

```
ex%ncipher new-world -m 1 -s 0 -i -Q 1/1
```

**Note:** The security world can only be created in “pre-initialization” mode.

- Step 5.** Load and add the PCI module to the security world.
- Step 6.** Change the state of the nCipher module to 'O' (Operational state) using the hardware switch on the backplane of the E|X 3650.
- Step 7.** Reset the module again by pressing “Clear button”.
- Step 8.** Confirm the mode of the module by typing the command:

```
tx%ncipher nfkminfo
```

The system will reply with:

```
Currently state will be initialised.
state 0x1725000 Initialised Usable Recovery !PIN Recovery...
n_modules 1
...
```

- Step 9.** Generate a key protected by the module (the Admin key). If the key is protected by a token then an Operator Card Set (OCS) needs to be created using the `createocs` command as shown below. The user responses are shown in **bold**.

```
ex%ncipher generatekey embed
```

```
nCipher KM key generation/import utility
Protected by ? (token/module) [token] module
```

```
Key name ? [] test
Key size ? (512/768/1024/2048) [1024]1024
Site domain name ? [] red.com
Org. unit ? [] RLN
Organization ? [] RLN
City/Locality ? [] Campbell
State/Province ? [] CA
Country Code ? [] US
Email address ? [] bmartino@redlinenetworks.com
Save key file as: test
Recovery feature ? (y/n) [y]y
key generation/import parameter(s):
protect Protected by Module
plainname Key name test
size Key size 1024
x509dnscommon Site domain name red.com
x509orgunit Org. unit
x509org Organization
x509locality City/Locality
x509province State/Province
x509country Country Code
x509email Email address
embedsavefile Save key file as test
recovery Recovery feature 1
Generating fresh key ...
Key generated and stored.
This creates test, test_req, test_selfcert files.
```

**Step 10.** To see the list of key and certificate files

```
ex%list file
```

**Step 11.** Bind the key/cert files with the SSL VIP using

```
ex%set cluster 1 listen keyfile test
ex%set cluster 1 listen certfile test_selfcert
```

One key can be bound to more than one VIP.

## Changes to the Standard Configuration

The standard configuration has been changed on appliances modified to implement FIPS. The following Command Line commands have been removed or modified.

- All of the **gen** commands have been removed:
  - **gen csr**
  - **gen ssc**
  - **gen cac**
  - **gen key**
- The Admin ssl key /cert interface has changed:
  - **set admin webui ssl keyfile:** Defines the key to be used for admin login
  - **set admin webui ssl certfile:** Defines the certificate to be used for admin login.
  - **set admin webui ssl keypass:** Sets the pass-phrase for private key protection.

“The nCipher Utilities” on page 5 lists the nCipher commands that are available with the addition of the nCipher card into the Redline FIPS. To access these commands, you need to precede the command with the word “ncipher”, as follows:

```
ex%ncipher cardpp
```

the generic version of this command is:

```
"ncipher <ncipher command name>"
```



## Chapter 2. The nCipher Utilities

The nCipher utility files shown in Table 2-1 are supplied in the bin subdirectory of your nCipher installation. The abbreviations used in the Type column indicate the following:

- **KM:** The file requires a key management module.
- **SW:** Software is used to manage security worlds and are described in the appropriate chapter.
- **FW:** The firmware requires the module firmware release to be 1.71.11 or later.
- **FW2:** The firmware requires the module firmware release to be 1.75.0 or later.

Utilities that are not marked internal, PK, KM, or SW can be run on all nCipher modules.

**Table 2-1 nCipher Utilities**

Utility	Type	Description	See Page
cardpp	SW FW	This utility adds, changes, removes, or verifies a pass phrase.	
checkmod		This utility checks modulo exponentiations.	7
createocs	SW	This utility creates or erases Operator Card Sets. <b>Note:</b> Operator Card Sets can <i>ONLY</i> be erased while the security world used to create them is <i>ACTIVE</i> on your server.	
crypttest	KM	This utility tests all defined symmetric cryptographic mechanisms.	8
des_kat	KM	This utility performs DES known-answer tests.	9
enquiry		This utility returns information about the nCipher server and connected modules.	10
floodtest		This utility performs hardware speed testing.	16
fwcheck	FW	This utility verifies the firmware version.	18
generatekey	SW	This utility generates or imports keys.	
hardserver		This is the nCipher server program. It is installed as a service and runs automatically.	19
initunit	KM	This utility initializes a module.	20
killrecov	KM	This utility disables the Replace Operator Card Set feature for a security world.	21
kmfile-dump	KM	This utility displays key-management information from a security world's key-management data file.	26
kptest		This utility tests the consistency of encryption and decryption, or of signature and verification, with the RSA and DSA algorithms.	27
loadrom		This utility loads new firmware into a module.	29
mkaclx	SW2	This utility generates keys on a one-off basis.	30
moddate		This utility displays the signed module state.	31

**Table 2-1 nCipher Utilities**

Utility	Type	Description	See Page
ncdate		This utility sets, updates and views the time on a module's real-time clock.	32
ncversions		This utility displays the versions of installed nCipher support software components	32
new-world	SW FW	This utility creates a security world. It replaces sw-init.	34
nfkminfo	SW	This utility displays information about a security world, cards, and keys.	38
nfkverify		The <b>nfkverify</b> utility verifies key generation certificates.	43
nopclearfail		This utility clears a module, puts a module into the error state, or retries a failed module.	46
nvr-am-sw	FW2	This utility modifies and displays information about NVRAM areas.	47
pollbare	FW	This example utility returns information about state changes.	49
pubkey-find	FW	This example utility returns information the key, certificate or certificate request in a file.	50
racs	SW FW	This utility replaces your existing Administrator Card Set (ACS) with a new ACS, of the same size and security policy as the original ACS.	52
randchk		This utility runs a universal statistical test on random numbers returned by the module.	53
rocs	SW	This utility replaces an Operator Card Set.	
sigtest	KM	This utility measures module speed using RSA or DSA signatures or signature verifications.	54
slotinfo	KM	This utility returns information about tokens in a module.	56
stattree		This utility returns statistics gathered by the nCipher server and modules.	57
with-nfast	KM	This utility loads keys onto a module before an application is run in another session.	63

**Help Options:**

If a utility has the standard help options, these are not included in the synopsis or in the description. The standard help options are as follows:

- **-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

## Detailed Utility Usage

This section provides details on the usage of each of the utilities.

### checkmod

The **checkmod** utility checks modulo exponentiations performed on the module against test data.

#### Usage:

```
/opt/nfast/bin/checkmod [testfile [testfile...]]
```

In this command, *testfile* is the name of a test file. The test file consists of lines that contain values for the modexp command: variables A, P, and N as well as the result R. Several example test data files are provided in the `/opt/nfast/testdata` directory. The name of each file starts with `testdata`.

#### Output:

**checkmod** counts the number of successful commands. For example:

```
/opt/nfast/bin/checkmod /opt/nfast/testdata/testdata-s1024-
n100 File /opt/nfast/testdata/testdata-s1024-n100:
100 completed OK
```

If any command fails or if the value returned does not match the value supplied in the test file, **checkmod** prints an error message and then terminates.

#### Options:

This option is required if you are working in FIPS 140-2 level 3 mode.

**cryptest**

The **cryptest** utility tests all defined symmetric cryptographic mechanisms. **cryptest** can also test encryption and signature mechanisms. Although defined, some mechanisms may not be supported on a particular module because of licensing restrictions.

If the mechanism is supported, **cryptest** reports the sizes for which it successfully encrypts and decrypts a message. If a mechanism is defined but not supported by the module, **cryptest** reports Unknown Mechanism.

**Usage:**

```
/opt/nfast/bin/cryptest [-e|--encryption] [-s|--signature] [-E|--channel-encrypt]
[-S|--channel-decrypt] [-m|--max-size=BYTES] [-b|--max-channel-update=BYTES]
```

**Test Options:**

- **-e, --encryption;** Tests the encryption mechanisms
- **-s, --signature;** Tests the signature mechanisms
- **-E, --channel-encrypt;** Specifies the use of channel commands to encrypt or sign
- **-S, --channel-decrypt;** Specifies the use of channel commands to decrypt or verify
- **-m, --max-size=BYTES;** Limits the size of the block to be encrypted to the number of bytes specified in *BYTES*
- **-b, --max-channel-update=BYTES;** Uses an update size for channels that is less than or equal to the number of bytes specified in *BYTES*.

If neither the **--encryption** nor the **--signature** option is given, **cryptest** tests both encryption and signature mechanisms.

**Output**

The **cryptest** utility returns output similar to this:

```
KeyType_DES:
Mech_Any: encrypt 0 1 7 8 9 15 16 17 20 36 64 256 1025 2048 4096 8192 16384 65536
sign 0 1 7 8 9 15 16 17 20 36 64 256 1025 2048 4096 8192 16384 65536
Mech_DESmCBCi64pPKCS5: encrypt 0 1 7 8 9 15 16 17 20 36 64 256 1025 2048 4096 8192
16384 65536
Mech_DESmECBpPKCS5: encrypt 0 1 7 8 9 15 16 17 20 36 64 256 1025 2048 4096 8192
16384 65536
Mech_DESmECBpNONE: encrypt 0 8 16 64 256 2048 4096 8192 16384 65536
Mech_DESmCBCpNONE: encrypt 0 8 16 64 256 2048 4096 8192 16384 65536
Mech_DESmCMACi64pPKCS5: sign 0 1 7 8 9 15 16 17 20 36 64 256 1025 2048 4096 8192
16384 65536
...
```

This utility resolves situations where key counting for the specified key is disabled because the NVRAM file area is not present. This could occur, for example, because the NVRAM file has been erased by module initialization or because a module has been replaced. Key counting can be enabled again by regenerating the required NVRAM file area. A quorum of the ACS is required to authorize regeneration of the NVRAM file.

### **des\_kat**

The **des\_kat** utility performs DES known-answer tests and indicates if any of them fail.

#### **Usage:**

```
/opt/nfast/bin/des_kat
```

#### **Output:**

The **des\_kat** utility should return:

```
DES known answer tests for nFast version 1.x.x  
Tests passed
```

If any tests fail, **des\_kat** returns the number of the failed test. Please inform Support at nCipher.

**enquiry**

The **enquiry** utility returns information about the nCipher server and the modules connected to it.

**Usage:**

```
/opt/nfast/bin/enquiry -m|--module=MODULE
```

In this command, **--module=MODULE** is the module number of the module about which you want to receive information. If you do not specify a module, **enquiry** returns information about all modules.

**Output:**

The **enquiry** utility returns the following data for the server and each module:

**Table 2-2 The enquiry Utility Return Values**

Return	Description
enquiry reply flags	Failed indicates that the module has failed. This failure may be because the module is in the Error state or because there is a problem on the SCSI or PCI bus.
enquiry reply level	This is the version of enquiry that the module supports. For release 1.40 and later, this level is Four or higher.
serial number	This is the electronic serial number of the module. Please quote this number when contacting Support at nCipher.
mode	These are the options that will be reported: <ul style="list-style-type: none"> <li>• error</li> <li>• operational</li> <li>• initialization</li> <li>• maintenance</li> <li>• pre-initialization</li> <li>• pre-maintenance</li> <li>• uninitialised</li> </ul>
version	This is the version of the nCipher server or firmware.
speed index	This is the estimated number of 1024-bit modular exponentiations that the module can perform in 1 second.
rec. queue	This is the recommended minimum and maximum number of jobs in the job queue to keep the module fully loaded.

**Table 2-2 The enquiry Utility Return Values**

Return	Description
level one flags	One or more of the following flags is set: <ul style="list-style-type: none"> <li>• The Hardware flag is always set</li> <li>• The HasTokens flag is set if the module has a smart card interface</li> <li>• The MaintenanceMode flag is set if the module is in Maintenance state</li> <li>• The InitialisationMode flag is set if the module is in Initialization state</li> <li>• The PreMaintInitMode flag is set if the module is in Pre-Maintenance or Pre-Initialization state</li> </ul>
version string	This indicates the version of the nCipher server or firmware.
checked in	This indicates the date the firmware was last modified.
level two flags	There should be no level two flags set. If any are set, contact Support at nCipher.
max. write size	This is for nCipher use only—currently, it is 8192.
level three flags	The KeyStorage flag is set if you have a key-management module.
level four flags	One or more of the following flags can be set: <ul style="list-style-type: none"> <li>• The OrderlyClearUnit flag is set for firmware release 1.40 and later.</li> <li>• The HasRTC flag is set if the module has a Real-Time Clock.</li> <li>• The HasNVRAM flag is set if the module has non-volatile memory.</li> <li>• The HasNSOPermsCmd flag is set if the module supports the SetNSOPerms nCore API command.</li> <li>• The ServerHasPollCmds flag is set if the server or any module supports the PollModuleState and PollSlotTest API commands.</li> <li>• The FastPollSlotList flag is set for a particular module if PollSlotList on that module does not require module interaction. This is the case if both the hardserver and the module support the relevant extensions.</li> </ul> If you purchased a developer kit, you can refer to the relevant developer documentation for information on nCore API commands.

**Table 2-2 The enquiry Utility Return Values**

Return	Description
module type code	(Reply level five or later) The numeric value of the module type. This can be: <ul style="list-style-type: none"> <li>• 0 for the server</li> <li>• 2 for models nC1001S-xxx, nC3021S-xxx and nC3031S-xxx</li> <li>• 4 for models nC1001P-xxx and nC3021P-xxx</li> <li>• 5 for models nC3022W-xxx and nC3032W-xxx</li> <li>• 6 for models nC1002P-xxx and nC3022P-xxx</li> </ul>
product name	(Reply level five or later) The product name. For the server, this is "nFast server".
device name	(Reply level five or later) The ModuleID, bus, and slot for the device, as reported in log messages. For example #1 SCSI bus 0 id 2 means ModuleID 1 on SCSI bus 0 with SCSI ID 2. The device name is blank in the information on the server and for installations where the server software is earlier than version 1.60.5.

**Example 1: Standard Output:**

Critical information is highlighted with explanatory text in italics.

```

Server:
enquiry reply flags  none [contact nCipher Support should this report anything else]
enquiry reply level  Five
serial number        a 12-digit, alpha-numeric serial number will appear here for each nCipher module attached]

mode                 operational
version              1.71.11 [This will match the version of the "nFast" package installed]
speed index          147 [This is the sum total of SSL transactions per second your attached nCipher devices can process; nCipher devices are scalable so this number will increase with each addition, according to each device's "speed"]

rec. queue           37..102 [This is sum total of threads recommended for your application to send requests to the nCipher devices]

level one flags      Hardware HasTokens
version string        1.71.11, 1.71.11 built on Mar 21 2001 16:01:59 [The first number in a version string represents the version of "nFast" package installed. The second number in a version string represents the firmware version installed on your nCipher device(s).]

checked in           000000003ab8cc96 Thu Mar 22 00:45:26 2001
level two flags      none
max. write size      8192
level three flags    KeyStorage [This confirms you are using a key management device]
level four flags     OrderlyClearUnit HasRTC HasNVRAM HasNSOPermsCmd ServerHasPollCmds HasSEE

module type code     0
product name         nFast server
device name

Module #1:
enquiry reply flags  none
    
```

```

enquiry reply level  Five
serial number       [A 12-digit, alpha-numeric serial number will appear here]
mode                operational
version            1.71.11 [This is your module's firmware version]
speed index        147 [this is the number of SSL transactions per second this single nCipher
                    devices can process]
rec. queue         9..17 [This is sum total of threads recommended for your application to send
                    requests to the nCipher devices]

level one flags    Hardware HasTokens
version string     1.71.11 built on Mar 21 2001 16:01:59
checked in        000000003ab8cc96 Thu Mar 22 00:45:26 2001
level two flags    none
max. write size    8192
level three flags  KeyStorage [This confirms you are using a key management device]
level four flags  OrderlyClearUnit HasRTC HasNVRAM HasNSOPermsCmd ServerHa-
                    sPollCmds HasSEE

module type code   5
product name       nC1002W/nC3022W/nC4032W
device name        #1 SCSI bus 0 id 1

```

**Example 2: Output when there is no nCipher device attached, or there is a problem with one or more attached nCipher devices:**

**NFastApp\_Connect failed: ServerNotRunning**

This indicates there is a problem with your nCipher devices, or that there are no nCipher devices attached.

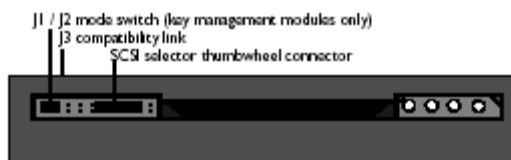
- Check your server. For each attached nCipher device, check the nCipher Blue LED to determine if it is in a normal mode. Recently, we added a special section to our web site which provides active depictions of the nCipher heartbeat, our blue LED via this link:

<http://www.ncipher.com/support/leds/index.php>

- If the nCipher device(s) LEDs appear to be in a normal mode, proceed as with issues reported by the Install script. Power down your server to check the connections of your nCipher hardware to the server:

SCSI devices:

- (External SCSI only) Turn off power and check power cord
- Check the SCSI cable connections to the device and server and ensure there are no bent pins on either side of the cable
- Confirm that the terminator is on securely
- (Solaris and AIX only) ensure there is a jumper on J3:



Rear view of an nCipher SCSI module

- Check your SCSI controller, as it may not be suitable for Wide SCSI, necessitating the addition of a jumper on J4

PCI devices:

- Remove and reconnect the PCI card to your motherboard, ensuring it is secure in its slot.
- Alternatively, swap its slot with another PCI card on your motherboard that is known to be working.

**Example 3: Output for a SCSI device that can be detected by the hardserver, but cannot be used due to a SCSI communication problem:**

The output will look like this:

```
Server:
enquiry reply flags none
enquiry reply level Four
serial number
mode operational
version 1.52.5
speed index 0
rec. queue 2..50
level one flags none
version string 1.52.5,
checked in 00000000380adc4c Mon Oct 18 04:37:32 1999
level two flags none
max. write size 8192
level three flags none
level four flags none

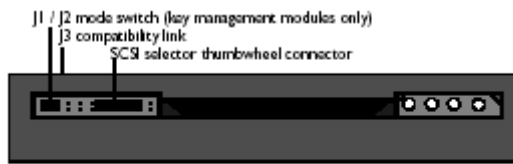
Module #1:
enquiry reply flags Failed
enquiry reply level Zero
```

Note the lack of data in those items highlighted in italics compared to the normal output in #1 above. Also note that, as opposed to Example 2, there is data being reported for the hardserver (“server”), and that the module can be seen but not communicated with. This indicates that a SCSI problem is affecting communication.

- Check your server. For each attached nCipher device, check the nCipher Blue LED to determine if it is in a normal mode. Recently, we added a special section to our web site which provides active depictions of the nCipher heartbeat, our blue LED via this link: <http://www.ncipher.com/support/leds/index.php>
- If the nCipher device(s) LEDs appear to be in a normal mode, proceed as with issues reported by the Install script. Power down your server to check the connections of your nCipher hardware to the server:

**SCSI devices:**

- (External SCSI only) Turn off power and check the power cord.
- Check the SCSI cable connections to the device and server and ensure there are no bent pins on either side of the cable.
- Confirm that the terminator is on securely.
- (Solaris and AIX only) ensure there is a jumper on J3:



Rear view of an nCipher SCSI module

- Check your SCSI controller, as it may not be suitable for Wide SCSI, necessitating the addition of a jumper on J4.

**PCI devices:**

- Remove and reconnect the PCI card to your motherboard, ensuring it is secure in its slot.
- Alternatively, swap its slot with another PCI card on your motherboard which is known to be working.

**floodtest**

The **floodtest** utility performs hardware speed testing using modular exponentiation, with the Chinese Remainder Theorem.

**Usage:**

```
/opt/nfast/bin/floodtest [--crt|--no-crt] [-Q|--query] [-l|--job-size=BITS]
[-t|--stop-after=TIME] [-j|--outstanding-jobs=COUNT] [-n|--jobs-count=COUNT]
[[[-K|--skew-check=SKEW ]| [-T|--min-check=COUNT ]] [-C|--check-start=TIME ]]
[--overprint][--B|--unbuffered-stdout][--o|--output=FILE]
[-r|--report-interval=TIME}
```

**Program Options:**

- **--crt:** Uses CRT optimization (this is the default).
- **--no-crt:** Does not use CRT optimization.
- **-Q, --query:** Uses Query mode (spinlock) rather than Wait mode.
- **-l, --job-size=BITS:** Sets the size of exponentiation modulus to the number of bits specified in *BITS*. The default is 1024. The value of *BITS* must be between 64 and 4096. *BITS* must be a multiple of 64

*Note: The module may run out of memory if you run **floodtest** with sizes greater than 2048, especially if the module has any keys or tokens loaded.*

- **-j, --outstanding-jobs=COUNT:** Tries to keep *jobs* number of jobs outstanding at a time. The default value of *COUNT* is the maximum number of jobs recommended for the hard-server, plus one.
- **-t, --stop-after=TIME:** Specifies the maximum time to run the test. Use the suffix s to specify seconds, m minutes, h hours and d days. The default value of *TIME* is infinity.
- **-n, --jobs-count=COUNT:** Specifies the maximum number of jobs to run. The default value of *COUNT* is infinity.

**Automatic Checking Options**

- **-T, --min-check=COUNT:** Performs threshold checking, starting after either 15 seconds or the time specified by **--check-start**. Subsequently, when **floodtest** writes an output line, it quits with an error message if the overall average number of modular exponentiations each second drops below *COUNT*.
- **-K, --skew-check=SKEW:** Performs skew checking, after either 15 seconds or the time specified by **--check-start**. The **floodtest** utility records the overall average number of modular exponentiations each second as *rec\_ave*. Subsequently, when **floodtest** writes an output line, it quits with an error message if the average is outside the range: *rec\_ave* ± *SKEW*.

*Note: The **--min-check** and **--skew-check** options are mutually exclusive.*

- **-C, --check-start=TIME:** Specifies the time in seconds at which threshold or skew checking starts. The default value of *TIME* is 15.

**Output Options:**

- **--overprint:** Prints results all on one line, using \r rather than \n.
- **-B, --unbuffered-stdout:** Turns off buffering for stdout (if the **--output** option is specified, **--unbuffered-stdout** also turns off buffering for the output stream)

- **-o, --output=FILE:** Specifies that results be written to *FILE* in addition to stdout.
- **-r, --report-interval=TIME:** Displays, at the specified interval of *TIME* seconds, the total number of exponentiations achieved and the rate at which they are performed. The default value of *TIME* is 1.

**Output:**

The **floodtest** utility returns output similar to this:

```
hardware, speed index 296, using rec. max outstanding + 1 = 46
1, 148 59.2, 151 overall
2, 410 140.32, 206 overall
3, 677 190.992, 226 overall
4, 944 221.395, 267 overall
5, 1190 231.237, 256.5 overall
...
```

The first column shows the number of seconds. The second shows the total number of exponentiations performed. The third column shows the number of exponentiations achieved this second. The last column shows a moving average of the number of exponentiations achieved each second.

## fwcheck

The **fwcheck** utility verifies a firmware image by using one of several files that contain verification data.

### Usage:

```
/opt/nfast/bin/fwcheck [-v|--verbose] [-m|--module=MODULE] [-C|--challenge=HEX]  
NFF-file
```

### Help Options

- **-h, --help**; Displays help for **fwcheck**.
- **-V, --version**; Displays the version number of **fwcheck**.
- **-u, --usage**; Displays a brief usage summary for **fwcheck**.
- **-v, --verbose**; Displays all possible output for **fwcheck**.

### Checking options

- **-m, --module=MODULE**; Checks the module number *MODULE*.
- **-C, --challenge=HEX**; Uses *HEX* as the challenge. If not specified, the challenge is a random value.

The **--challenge** option supplies a random value that is used in the challenge-response protocol. If this is not supplied, **fwcheck** creates a value by hashing various system environment values together with random bytes generated by the module. When used with the **--verbose** option, it displays challenges and responses. This might be useful, for example, to compare the behavior of two modules.

The **fwcheck** utility can be used with the following types of files:

- *fw\_platform.nff* the regular application firmware file as supplied to loadrom
- *fw\_platform.ftv* performs a “pseudo-random sequence generation” test on the application

In these file names, *platform* is the version number of the module, beginning with nC. The pseudo-random sequence generation test asks the module to generate a fixed pseudo-random byte sequence that is a number of megabytes long. This is generated according to an algorithm kept secret inside the firmware and not present in the **fwcheck** application, which simply computes the hash of this sequence and compares it to a value in the .ftv file.

**hardserver**

The **hardserver** is the communications service between applications and nCipher modules.

**Usage:**

```
hardserver
```

The **hardserver** utility is the nCipher server program. It is started automatically.

**Stopping the nCipher Server:**

To stop the nCipher Server, type the command:

```
/opt/nfast/bin/init.d-nfast stop
```

**Restarting the nCipher server:**

To restart the nCipher Server, type the command:

```
/opt/nfast/bin/init.d-nfast start
```

### initunit

The **initunit** utility initializes a unit with a random module key and a known KNSO. This utility makes a key-management module usable but does not enable any key archival or recovery.

**Usage:**

```
/opt/nfast/bin/initunit -m|--module MODULE
```

**Module Options:**

- **-m, --module=MODULE;** Selects the module to reinitialize. If you do not specify a module, **initunit** initializes all modules that are in a pre-initialization state.

**Output:**

If **initunit** is successful, it returns a message similar to this for each module that has been initialised:

```
Initializing Unit #  
Setting dummy HKNSO  
Module Key Info:  
HKNSO #####  
HKM #####
```

Otherwise, it returns an error message. You should contact nCipher Support in this case.

**killrecov**

The **killrecov** utility is the nCipher Security World retrospective Operator Card Set recovery disabling tool.

**Usage:**

```
killrecov
```

**Warning:** The **killrecov** utility disables the replace Operator Card Set feature for the current security world. If you disable this feature, you cannot replace lost or damaged Operator Card Sets and therefore will not be able to access keys protected by such cards. You cannot re-enable this feature without reinitializing your security world and discarding all your existing keys.

You can determine whether your security world has the recovery feature enabled by using the **nfkminfo**. The **killrecov** utility removes recovery information from an Administrator Card Set. In order to ensure that Operator Card Set recovery becomes impossible, you must process sufficient Administrator Cards such that no combination of the required number of cards (K) contains only unprocessed cards. Therefore, you must process at least  $N+1-K$  cards. For example, if you have a 5-of-7 Administrator Card Set, you must process at least  $7+1-5 = 3$  cards. You can process all the cards in the Administrator Card Set.

Use of **killrecov** ensures that application keys cannot be loaded without the appropriate Operator Cards. However, once loaded, keys that were recoverable before you ran **killrecov** may have their protection properties varied if authorized by the Administrator Card Set. To prevent this, *in any security world in which you have disabled key recovery with keyrecov*, every time you start a program or utility that requires you to insert a card from the Administrator Card Set you must always press the Clear button on the module *before* running the program or utility. Pressing the Clear button ensures that no application keys are loaded when you insert an Administrator Card.

The **killrecov** utility updates the security world information on the host computer. If you have shared the security world across several computers, you must either:

- Copy the modified kmdata directory to each computer
- Run **killrecov** on each computer.

**Note:** You only need to process each card in the Administrator Card Set once. It is not necessary to process any cards in order to update the host data. Because **killrecov** erases information and does not reassemble the secrets, you do not need to supply the cards' pass phrases.

*Never insert your Administrator Cards into anything except an nShield module connected to a fully trusted computer.*

**Usage:**

**Step 1.** Start one module in the pre-initialization state.

**Step 2.** Run **killrecov** using the command:

```
/opt/nfast/bin/killrecov
```

If no module is in the correct state, **killrecov** displays a message similar to this:

```
-----  
nCipher security world retrospective recovery disabling tool.  
Module #1 Usable - cannot use to modify Admin Cards  
Module #2 Foreign - cannot use to modify Admin Cards  
killrecov: no suitable modules  
-----
```

If there is at least one module in the pre-initialization or factory state, **killrecov** displays a message similar to this:

```
-----  
nCipher security world retrospective recovery disabling tool.  
Module #1 PreInitMode - can use (will be erased)  
Module #2 Foreign - cannot use to modify Admin Cards  
Warning: If you continue, the recovery feature will be perma-  
nently disabled and recovery will not be possible in the  
future. This operation is not reversible!  
Warning: If you continue, Module #1 will be completely erased;  
you will need to reinitialize it if you wish to use it after  
this.  
Warning: As ever, do not insert your administrator cards into  
anything except an nCipher HSM connected to a fully trusted  
host.  
To ensure recovery is properly and securely disabled, you must  
use this utility to modify at least # Administrator Cards, so  
that no quorum (# cards) of unmodified cards remains.  
Will use Module #1 Slot #0.  
Proceed (say 'yes' to continue)?  
-----
```

If **killrecov** cannot find recovery information in the security world, it displays the following additional lines:

```
-----  
Recovery information no longer present in security world host  
data - but Administrator Cards may still need modifying.  
-----
```

If you believe that your Administrator Card Set may have been created with recovery, continue processing the cards.

The **killrecov** utility uses the first module in the factory, pre-initialization, or initialization state. If you want to use a different module, type no, and press Enter. Then reset the modules, and run **killrecov** again. You do not need to run **killrecov** on each module.

**Step 3.** Type yes, and press Enter.

The **killrecov** utility displays the message:

```
-----  
Insert Administrator Card to be modified or checked and press  
RETURN, or type 'q' or 'quit' and press RETURN when you are  
finished.  
-----
```

**Step 4.** Either:

- Insert a card from the Administrator Card Set for the current security world, and press Enter.
- or
- Press Q and then Enter to quit.

If you quit before modifying sufficient Administrator Cards, **killrecov** displays the message:

```
-----
security world host updated to reflect lack of recovery.

Warning: Recovery is not completely disabled unless at least s
of your n Administrator Cards are modified to remove the
recovery data. Only t cards have been modified or checked in
this run.

Warning: If you ever used the Replace Administrator Card Set
feature, and did not erase a full quorum of the old Adminis-
trator Cards, then the old Administrator Cards might still be
used with old Security World host data to perform recovery.
-----
```

If you insert a card from a different Administrator Card Set, **killrecov** displays the message:

```
-----
This is not an Administrator Card from the current security
world.
-----
```

If you insert a card from the Administrator Card Set that does not have recovery data, **killrecov** displays the message:

```
-----
This is Administrator Card #i.

There is no recovery information on this Administrator Card -
it has already been modified to disable its use for recovery.-
-----
```

If you insert a card from the Administrator Card Set that still has recovery data, **killrecov** displays the message:

```
-----
This is Administrator Card #i.
-----
```

The **killrecov** utility deletes the recovery information from this card and displays the message:

```
-----
Deleted recovery key material from Administrator Card #i.
-----
```

**Note:** If for any reason **killrecov** cannot delete the information, it displays an error message. Such a message usually indicates that the card is damaged. Replace the Administrator Card Set after you have completed processing the cards.

After **killrecov** has read and modified the card as needed, it displays the message:

```
-----  
Remove the card and press the HSM Clear button. Check that the  
Status LED flashes differently for a second or two.  
-----
```

**Step 5.** Remove the card, and press the Clear button.

You must clear the module between processing each Administrator Card. Doing so ensures that an attacker cannot reassemble the secrets protected by the Administrator Card Set and use them to authorize other actions.

The **killrecov** utility displays the message:

```
-----  
Insert Administrator Card to be modified or checked and press  
RETURN, or type 'q' or 'quit' and press RETURN when you are  
finished.  
-----
```

**Step 6.** Repeat Steps 4 and 5 for subsequent cards.

After you have modified a sufficient number of cards from the Administrator Card Set, **killrecov** displays the message:

```
-----  
Sufficient Administrator Cards have been modified, but you may  
choose to modify or check the remaining cards too. Insert  
Administrator Card to be modified or checked and press RETURN,  
or type 'q' or 'quit' and press RETURN when you are finished.  
-----
```

This message indicates that there are no longer enough cards in the Administrator Card Set with the necessary information to reassemble the keys required for recovery. It is impossible to recover Operator Card Sets using this Administrator Card Set. However, you can still process the remaining cards.

**Step 7.** Either:

- Press Q and then Enter to quit
- or
- Repeat steps 4 and 5 for the remaining cards.

After you have modified all the cards from the Administrator Card Set, **killrecov** displays the message and exits:

```
-----  
All Administrator Cards have been modified.  
Security world host data has already been updated.  
Recovery now disabled for this security world.  
WARNING - If you ever used the Replace Administrator Card Set  
feature, and did not erase a full quorum of the old Adminis-  
trator Cards, then the old Administrator Cards might still be  
used with old Security World host data to perform recovery.  
-----
```

The **killrecov** utility has reinitialised the module to the factory state. In order to continue to use the module as part of your security world, you must add it to the security world again.

**Step 8.** Add the module to the security world, using either the **new-world** command or Key-Safe.

**Note:** After you have used the **killrecov** utility, each time that you start an operation involving the use of your Administrator Card Set, you **must always** press the module's Clear button **before** starting the program or utility.

### kmfile-dump

The **kmfile-dump** utility displays detailed key-management information from a security world's key-management data file. You can also display information about the status of your security world with the **nfkminfo** command-line utility.

#### Usage:

```
kmfile-dump [-b|--binary] [-p|--plain] [-v|--verbose] file  
[file ...]
```

In this command, *file* is the name of the file from which data is dumped. The key-management data files from which you can display data are located in the `kmdata` directory. They are:

- The module file
- The security world file
- The card file
- The cards file
- The key file.

#### Program Options:

- **-b, --binary**; Displays all entries in binary
- **-p, --plain**; Uses plain format for binary output with no offsets or ASCII
- **-v, --verbose**; Shows binary dumps of key blobs

**kptest**

The **kptest** utility tests the consistency of encryption and decryption, or signature and verification, with the RSA and DSA algorithms. It can also test key generation by regenerating the key pair used every given number of checks. **kptest** is not intended as a speed test.

**Usage:**

```
/opt/nfast/bin/kptest [-s|--sign-verify] [-e|--encrypt-decrypt]
[-k|--key-regenerate=CHECKS] [-i|--plain-size=SIZE] [-m|--module=MODULE]
[-t|--stop-after=TIME] [-n|--jobs-count=COUNT] [-S|--sig-type=TYPE]
[-l|--keysize= BITS] [-M|--mech=MECH] [-p|--plain=TYPE]
[[[-K|--skew-check=SKEW ]|[-T|--min-check=COUNT ]] [-C|--check-start=TIME ]]
[--overprint][--b|--unbuffered-stdout][--o|--output=FILE]
[--r|--report-interval=TIME]
```

**Program Options:**

- **-s, --sign-verify**; Test the sign/verify operation. This is the default for DSA and KCDSA.
- **-e, --encrypt-decrypt**; Test the encrypt/decrypt operation. This is the default for RSA.
- **-i, --plain-size=SIZE**; Use plaintext of *SIZE* bytes.
- **-k, --key-regenerate=CHECKS**; Regenerate the key every *CHECKS* number of checks.
- **-m, --module=MODULE**; Use module number *MODULE*. The default is 1.
- **-t, --stop-after=TIME**; Specifies the maximum time to run the test. Use the suffix *s* to specify seconds, *m* minutes, *h* hours and *d* days. The default value of *TIME* is infinity.
- **-n, --jobs-count=COUNT**; Specifies the maximum number of jobs to run. The default value of *COUNT* is infinity.

**Key Options:**

- **-S, --sig-type=TYPE**; Selects the signature type to use. Valid values are RSA, DSA and KCDSA. The default is RSA.
- **-l, --key-size=BITS**; Sets the key size in bits. The default is 1024.
- **-M, --mech=MECH**; Specifies the mechanism to use. Valid values are RSAPKCS1, DSA and KCDSAHAS160.
- **-p, --plain=TYPE**; Uses the plaintext type *TYPE*. Valid values are Bignum, Hash and Bytes.

**Automatic checking options**

- **-T, --min-check=COUNT**; Performs threshold checking, starting after either 15 seconds or the time specified by **--check-start**. Subsequently, when **floodtest** writes an output line, it quits with an error message if the overall average number of modular exponentiations each second drops below *COUNT*.
- **-K, --skew-check=SKEW**; Performs skew checking, after either 15 seconds or the time specified by **--check-start**. **floodtest** records the overall average number of modular exponentiations each second as average. Subsequently, when **floodtest** writes an output line, it quits with an error message if the average is outside the range  $\text{average} \pm \text{SKEW}$ .

*Note: The --min-check and --skew-check options are mutually exclusive.*

- **-C, --check-start=TIME**; Specifies the time in seconds at which threshold or skew checking starts. The default value of *TIME* is 15.

**Output Options:**

- **--overprint;** Prints results all on one line, using \r rather than \n.
- **-b, --unbuffered-stdout;** Turns off buffering for stdout (if the -o option is specified, -b also turns off buffering for the output stream)
- **-o, --output=FILE;** Specifies that results be written to *FILE* in addition to stdout.
- **-r, --report-interval=TIME;** Displays, at the specified interval of *TIME* seconds, the total number of exponentiations achieved and the rate at which they are performed. The default value of *TIME* is 1.

**Output:**

The **kptest** utility displays a message similar to this:

```
-----  
Using default RSA key type  
Using default keysize=1024  
Using default Bytes plaintext type  
Using default plaintext size=160  
Using default RSAPKCS1 mechanism  
Making 1024-bit RSAPrivate key on module #2  
1 : 40  
2 : 97  
3 : 155  
...  
-----
```

Subsequent lines show the time the test has been running and the total number of operations performed. To use this command, you must be logged in as Administrator.

**loadrom**

The **loadrom** utility loads new firmware into a module. Firmware is supplied as an encrypted and signed file. You can also use **loadrom** to obtain information about the firmware. For firmware prior to version 1.60.5, more limited information can be displayed.

**Usage:**

```
/opt/nfast/bin/loadrom [-v|--view] [-n|--notypecheck] [-m|--module=MODULE]
[-b|--maxblocksize=SIZE] NFF_file
```

In this command *NFF\_file* is the name of the file that contains the firmware. This has the extension .nff.

**Help Options:**

- **-h, --help**; Displays help for **loadrom**
- **-V, --version**; Displays the version number of **loadrom**
- **-u, --usage**; Displays a brief usage summary for **loadrom**

**Programming Options:**

- **-v, --view only**; Displays information about the .nff file and does not load it.
- **-m, --module=MODULE**; Checks the firmware on the module *MODULE*.
- **-b, --maxblocksize=SIZE**; Sets the maximum block size in bytes for module programming.
- **-n, --notypecheck**; Omits the module type check.

**Output:**

If you select the **--view** option, **loadrom** displays output similar to this:

```
-----
File : ../../module/firmware/fw_scsi.nff
Firmware : SCSI version 1.65.1+
for : nFast/SCSI
Filetype : NFast3
Programming chunks: 6
Confidentiality key:
SCSI/PCI mkI confidentiality key
Confidentiality mech:
DES3OldFwConf
Signatures:
#0: SCSI mkI integrity key
-----
```

For firmware prior to version 1.60.5, **loadrom -view** can recognize the signature key hashes and displays a corresponding text string (for example, SCSI mkI integrity key or PCI mkII KM integrity key).

During installation and when upgrading module firmware you can compare the output of **loadrom -view** with the information provided by the **enquiry** utility to ensure that the correct version of the firmware is present on the module.

**mkacIx**

The **mkacIx** utility generates a key with specified attributes on a oneoff basis. It is useful for testing.

**Usage:**

```
mkacIx[-k|--keygen-cert] [-K|--no-keygen-cert] [-C|--cardset-protected]
[-M|--module-protected] [-r|--recovery] [-R|--no-recovery] [-q|--quiet]
[-v|--verbose] [-a|--see-app-key=IDENT[:MECH]] [-t|--type=TYPE] [-b|--bits=BITS]
[-g|--group-size=BITS] [-O|--deny-oppermissions=OPPERMS] [-m|--module=MODULE]
[-s|--slot=SLOT] [-N|--name=NAME] [-T|--timeout=SECS] [-U|--use-limit=N] IDENT
[--confirm]
```

In this command, *IDENT* is the name required for the generated key.

**Module Selection Options:**

- **-m, --module=MODULE;** Specifies the module to generate the key. The default is the first available module.
- **-s, --slot=SLOT;** Specifies the slot to use to read card sets, if required. The default is 0.

**Key Generation Parameters:**

- **-t, --type=KEYTYPE;** Specifies the type of the generated key. The default is RSA.
- **-b, --bits=BITS;** Specifies the length in bits of the key to generate. The default depends on the key type.
- **-g, --group-size=BITS;** Specifies the group size for Diffie-Hellman keys in bits. The default depends on the key type.
- **-k, --keygen-cert;** Specifies that a key-generation certificate is stored. This is the default.
- **-K, --no-keygen-cert;** Specifies that no key-generation certificate is stored.
- **-O, --deny-opperms=OPFLAGS;** Specifies that the OpPermissions specified in *OPFLAGS* are disabled. *OPFLAGS* must be a comma separated list of OpPermissions.

**Key Protection Options:**

- **-C, --cardset-protected;** Specifies that a card-set protected key is generated.
- **-M, --module-protected;** Specifies that a module protected key is generated.
- **-r, --recovery;** Specifies that the generated key is recoverable. This is the default.
- **-R, --no-recovery;** Specifies that the generated key is not recoverable.
- **-a, --see-app-key=IDENT[:MECH];** Specifies that use of the generated key is restricted to SEE signed by the SEE integrity key *IDENT*, optionally with the mechanism *MECH*.
- **-T, --timeout=SECS;** Specifies the time limit in seconds for main-use operations.
- **-U, --use-limit=N;** Specifies the per-authorization limit for main-use operations.

**Other Options:**

- **-q, --quiet;** Produces fewer messages on successful runs.
- **-v, --verbose;** Produces more messages on successful runs.
- **-N, --name=NAME;** Specifies the name of the key. The default is that the key has no name.
- **--confirm;** Shows the command and requests confirmation.

**modstate**

The **modstate** utility displays the signed module state.

**Usage:**

```
/opt/nfast/bin/modstate [-m|--module MODULE] [--kml|--klf]
```

**Module Options:**

- **-m, --module=*MODULE***; Specifies the module number. The default is 1.
- **--kml**; Specifies `SignerType_KML`. This is the default.
- **--klf**; Specifies `47SignerType_KLF`.

## **ncdate**

The **ncdate** example utility views, sets, and adjusts the real-time clock in a module.

### **Usage:**

```
/opt/nfast/bin/ncdate [[-d|--display]] [-m|--module=MODULE]]
/opt/nfast/bin/ncdate --adjust [-m|--module=MODULE] hh:mm:ss [yyyy:mm:dd]
/opt/nfast/bin/ncdate --set [-m|--module=MODULE] hh:mm:ss [yyyy:mm:dd]
```

### **Action Selection Options:**

- **-d, --display;** Displays the current module time.
- **-m, --module=MODULE;** Specifies the module to use. The default is 1.
- **--set;** Sets the module time to the time specified
- **--adjust;** Adjusts the module time to the time specified.

To show the current time and date on the current module:

```
/opt/nfast/bin/ncdate
```

To set the time and optionally the date on the module *MODULE* to the given time and date:

```
/opt/nfast/bin/ncdate -m=MODULE -set hh:mm:ss [yyyy:mm:dd]
```

To update the time and optionally the date on the module *MODULE* to the given time and date:

```
/opt/nfast/bin/ncdate -m=MODULE -adjust hh:mm:ss [yyyy:mm:dd]
```

Using the **--adjust** option makes the module compute the drift rate of the internal clock based on the interval between the present time and the time when the clock was last set or adjusted. The module remembers this rate and uses it to correct future readings. It is therefore important that the time is obtained from the same source as it was previously.

## **ncversions**

The **ncversions** utility displays the versions of installed nCipher support software components.

### ***Usage:***

```
/opt/nfast/bin/ncversions
```

**new-world**

The **new-world** utility creates a security world that supports. **new-world** lets you specify different thresholds for different operations on the security world, such as:

- Adding a module
- Replacing an Operator Card Set
- Authorizing real-time clock operations
- Authorizing nonvolatile memory operations.

The **new-world** utility can also be used to add a module to an existing security world.

**Note:** If you create a security world with **new-world**, this security world is created with only one module. If you use other modules, then you must add them to the security world with **new-world**.

To use the **new-world** utility, you must be logged in to the host computer as Administrator.

**Usage:**

```
/opt/nfast/bin/new-world -i|initialize [-S|--no-remoteshare-cert]
[-o|--overwrite] [-F|--strict-fips-140-level-2-level-3] [-R|--no-recovery]
[-s|--slot=SLOT][--m|--module=MODULE]
-Q|--acs-quorum=K/N FEATURES /opt/nfast/bin/new-world
-l|program [-S|--no-remoteshare-cert]
[-s|--slot=SLOT][--m|--module=MODULE]
/opt/nfast/bin/new-world -e|--factory -m|--module=MODULE
```

**Help Options:**

- **-h, --help-features;** Displays the syntax for security world features

**Module Selection Options:**

- **-m, --module=*MODULE*;** Specifies the module to use. **new-world** initializes only one module at a time. If you have multiple modules, you must run **new-world** once for every module. The default is 1.
- **-s, --slot=*SLOT*;** Specifies the slot to use to read or write Administrator cards. The default is 1.

**Action Selection Options:**

- **-i, --initialize;** Tells **new-world** to initialize a new security world and program it into the module specified in `--module=MODULE`, replacing any existing kmdata directory.
- **-h, --help;** Displays help for **new-world**
- **-v, --version;** Displays the version number of **new-world**
- **-u, --usage;** Displays a brief usage summary for **new-world**
- **--program;** Tells **new-world** to load the existing security world from the kmdata directory into a module specified in `--module=MODULE`.
- **--factory;** Tells **new-world** to erase a module, restoring it to factory state.

**Note:** You must not include more than one of the `--initialize`, `--program`, and `--factory` options. If you do not specify any of these options and a kmdata directory already exists, **new-world** loads the security world to which the directory belongs. If

you do not specify any of these options and no kmdata directory exists, **new-world** creates a new security world.

#### ***New Security World Options:***

- **in --acs-quorum= $K/N$** ;  $K$  is the maximum number of smart cards required from the Administrator Card Set to authorize a feature. You can specify lower thresholds for a particular feature. Thresholds must be less than or equal to the total number of cards in the set.
- **In --acs-quorum= $K/N$** ;  $N$  is the total number of smart cards to be used in the Administrator Card Set. This value must be less than or equal to 64.

**Note:** You should not create an Administrator Card Set for which the required number of cards is equal to the total number of cards because you will not be able to replace the Administrator Card Set if even a single card is lost or damaged. The option **--acs-quorum= $K/N$**  only takes effect if you are creating a new security world.

- **-F, --strict-fips-140-2-level-3**; Creates 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. The **--strict-fips-140-2-level-3** option only has any effect if you are creating a new security world.

**Note:** This option provides compliance with the roles and services of the FIPS 140-2 level 3 standard, but nForce modules are only compliant with FIPS 140-2 level 2. It is included for those customers who have a regulatory requirement for compliance. If your installation includes an nForce module then it cannot satisfy a regulatory requirement for FIPS 140-2 level 3 compliance.

- **-o, --overwrite**; Tells **new-world** to allow the use of smart cards that already contain data. Any existing data will be erased. If a value for this flag is not specified, **new-world** will prompt you if a card contains data.
- **-R, --no-recovery**; Tells **new-world** to disable Operator Card Set recovery. The effect of setting this flag is the same as for specifying the feature **!r**.

By default, **new-world** creates key-recovery material that is protected by the cryptographic keys on the Administrator Card Set. This option does not give nCipher or any other third party access to your keys. Keys can only be recovered by using the Administrator Cards. nCipher recommends that you leave Operator Card Set recovery enabled. If you do not enable this feature now, you can never enable recovery for keys in this security world.

**Warning:** If you set the **--no-recovery flag**, you cannot replace lost or damaged Operator Card Sets and therefore cannot access the keys that are protected by cards from such sets. This feature cannot be enabled later without reinitializing your security world and discarding all your existing keys.

#### ***Features***

When initializing a security world, features may be specified as arguments after the module number. By default, all features are turned off, except **r** for Operator Card Set recovery. By default, Operator Card Set recovery is on.

Features are specified as a comma-separated list of terms. Each term consists of a feature name, optionally preceded by either a dash (“-”) or an exclamation point (“!”) to turn off the feature. They can optionally be followed by an equal sign (“=”) and a value that sets the threshold for this feature.

However, the “!” character, when used to turn off a given feature, is interpreted by some shells (for example, the C shell and its derivatives, bash and zsh) as requesting a history expansion. In these cases, you must escape the “!” character by preceding it with a “\” character—for example:

```
new-world 2 rtc,nv,\!r
```

Additionally, any feature entered on the command line with a leading hyphen (-) can be interpreted as an option. Attempting to pass a feature as a nonexistent option in this way returns an error (for example, unknown option). Prevent errors of this kind by using the standard POSIX double hyphen (--) marker to indicate that any subsequent features on the command line are not to be interpreted as options. For example:

```
new-world -m 2 -- -r
```

Currently understood feature names are:

- **m**, for module programming (this feature cannot be disabled)
- **r**, for Operator Card Set recovery
- **p**, for pass phrase recovery
- **nv**, for nonvolatile memory allocation
- **rtc**, real-time clock setting

The module programming (**m**) and Operator Card Set recovery (**r**) features are turned on by default; the other options are turned off by default. The **dsee** and **dseeall** options are not applicable unless you have purchased and installed nCipher’s CodeSafe Developer kit.

For example, the following features:

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

create a security world for which:

- A single card from the Administrator Card Set is required to add a new module.
- The default number is required to replace an Operator Card Set.
- Pass phrase recovery is not enabled.
- Two cards are required to allocate nonvolatile memory.
- One card is required to set the real-time clock..

**Note:** Under most conditions, it is advisable to turn on such features as nv, rtc, dsee, and (if desired) p. It is not usually advisable to turn on both dsee and dseeall simultaneously.

### **Output:**

If **new-world** cannot interpret the command line, it displays its usage message and exits. If you attempt to set a threshold for a feature that you have disabled or if you attempt to set a threshold too high, **new-world** displays an error and exits. If the module is not in the pre-initialization state, **new-world** displays an error and exits. Take the following steps in order to put the module into the correct state:

- *Internal SCSI modules*; Fit the initialization link, clear the module, and run **new-world** again
- *External modules*; Hold down the initialization switch, clear the module, and run **new-world** again.

If the module is in the pre-initialization state, **new-world** prompts you for smart cards and pass phrases as required. When **new-world** has initialised the module, restart the module in the operational state.

**nfkminfo**

The **nfkminfo** utility output provides you with immediate diagnostics on your server's nCipher Security World, the modules attached to your server (as they are seen by your security world), and the smart cards inserted into your modules at the time the command is run. This information is useful when your application is reporting problems using nCipher-protected keys, as it will confirm the status of your security architecture.

**“state” output**

The first, most crucial output from **nfkminfo** is the third line; state. You will see these parameters in all current nCipher modules running current nCipher software:

- **initialised** (security world status)
- **Usable** (whether nCipher device attached is authorized for use with the security world)
- **Recovery** (recoverability of key data if Operator Card is lost, damaged, or stolen)
- **PINRecovery** (ability to change Operator Card passphrases if originals are forgotten)

When there is an exclamation point in front of any of these parameters, it indicates that feature is not active:

- **!initialised** (security world is not initialised)
- **!Usable** (nCipher device attached is not authorized for use with the security world)
- **!Recovery** (keys cannot be recovered if Operator Card is lost, damaged, or stolen)
- **!PINRecovery** (Operator Card passphrases cannot be changed if originals are forgotten)

Depending on the type of nCipher device and software installed on your server, there are other parameters that are available: **ExistingClient**, **RTC**, **NVRAM**, **FTO**, **SEEDebug**. These are Secure-Execution-Environment (SEE) related, and are not pertinent here.

**Sample Outputs; Normal Output, One nCipher Device Attached**

```
World
generation 2
state 0x270000 initialised Usable Recovery !ExistingClient
n_modules 2
hkns0 954bef15eb32fe1f236ae55c9122fb629653cadd
hkm 530f60d27bc8cdb3d06bc747ccc3000729f91d00
hkmwk 1d572201be533ebc89f30fdd8f3fac6ca3395bf0
hkrc 1dd8fc6c8bef38bc1a838b9bf147d242762132c4
hkra 36f6861f57fc9e39973d47d73fbd1e4375783592
ex.client none

Module #1
generation 2
state 0x2 Usable
n_slots 2
esn [12-digit alpha-numeric serial number will appear here]
hkml 5bec1c30722762e2c4513c68a12d23b29c1b1793

Module #1 Slot #0 IC 1
generation 1
phystype SmartCard
```

```

slotlistflags 0x2
state      0x5 Operator
flags     0x10000 Passphrase
shareno   2
error     OK

```

```

Cardset
name      [Individual Operator Card Set name will appear here]
k-out-of-n 1/2
flags     NotPersistent
timeout   none
card names "" ""
hkltu     8b161e79163448531ee7888cc17e5670c59682f4

```

```

Module #1 Slot #1 IC 0
generation 1
phystype   SoftToken
slotlistflags 0x0
state      0x2 Empty
flags     0x0
shareno   0
error     OK
No Cardset

```

```
No Pre-Loaded Objects
```

**Sample Outputs; Output Where The nCipher Device Is Not Authorized For Use With The Security World**

```

World
generation 2
state      0x7250000 initialised !Usable Recovery !PINRecovery !ExistingClient
RTC NVRAM !FTO SEEDebug
n_modules 1
hkns0     8c4cd02ca66c731f7f569dfb4892441bdbfa09c0
hkm       811750192149e716a060a6a6c58d506e83279eac
hkmswk    1d572201be533ebc89f30fdd8f3fac6ca3395bf0
hkrs      58e1448ff49d59023d7d158a3bebaa4a7dee1619
hkrs      8785c0507669d5e9386679224d83c6e10d939a0a
ex.client none

```

```

Module #1
generation 2
state      0xa InitMode
flags     0x0 !ShareTarget
n_slots   2
esn       [12-digit alpha-numeric serial number will appear here]
hkml      76844a16d268b39426e86d94184b9dceda6660f

```

```

Module #1 Slot #0 IC 5
generation 1
phystype   SmartCard
slotlistflags 0x2

```

```

state      0x5 Operator
flags      0x0
shareno    1 (^If you give an entire Operator Card Set a name it will appear here')
shares     LTU
error      OK
Cardset
name       [Individual Operator Card name will appear here]
k-out-of-n 1/2
flags      Persistent
timeout    none
card names [Both card and card set names will appear here.]
hkltu     73993f5cdf514dba6dc414751ec12f732cf96793

Module #1 Slot #1 IC 0
generation 1
phystype   SoftToken
slotlistflags 0x0
state      0x2 Empty
flags      0x0
shareno    0
shares
error      OK
No Cardset

No Pre-Loaded Objects

```

In these cases, the nCipher device not authorized for use by the security world must be added to the security world by the command:

```
new-world -l -s 0 -m 1
```

**Sample Outputs; Output where One nCipher Device is Authorized for use with the Security World, but the Other Device is Not**

```

World
generation 2
state      0x270000 initialised Usable Recovery !ExistingClient
n_modules 2
hkns0     954bef15eb32fe1f236ae55c9122fb629653cadd
hkm       530f60d27bc8cdb3d06bc747ccc3000729f91d00
hkmwk     1d572201be533ebc89f30fdd8f3fac6ca3395bf0
hkrc      1dd8fc6c8bef38bc1a838b9bf147d242762132c4
hkra      36f6861f57fc9e39973d47d73fbd1e4375783592
ex.client none

Module #1
generation 2
state      0x2 Usable
n_slots    2
esn       [The 12-digit alpha-numeric serial number will appear here]
hkml      5bec1c30722762e2c4513c68a12d23b29c1b1793

Module #1 Slot #0 IC 1

```

```

generation 1
phystype SmartCard
slotlistflags 0x2
state 0x5 Operator
flags 0x10000 Passphrase
shareno 2
error OK
Cardset
name [The Individual Operator Card name will appear here]
k-out-of-n 1/2
flags NotPersistent
timeout none
card names "" ""
hkltu 8b161e79163448531ee7888cc17e5670c59682f4

Module #1 Slot #1 IC 0
generation 1
phystype SoftToken
slotlistflags 0x0
state 0x2 Empty
flags 0x0
shareno 0
error OK
No Cardset

Module #2
generation 2
state 0x5 Factory
n_slots 2
esn [The 12-digit alpha-numeric serial number will appear here]
hkml 93c4136558cbf0a9550e90e7f3840cbe6f587acc

Module #2 Slot #0 IC 1
generation 1
phystype SmartCard
slotlistflags 0x2
state 0x7 Error
flags 0x0
shareno 0
error TokenAuthFailed
No Cardset

Module #2 Slot #1 IC 0
generation 1
phystype SoftToken
slotlistflags 0x0
state 0x2 Empty
flags 0x0
shareno 0
error OK
No Cardset

No Pre-Loaded Objects

```

In these cases, the nCipher device not authorized for use by the security world must be added to the security world by the command:

```
new-world -l -s 0 -m 2
```

(as it's the second module needing to have the security world loaded)

**nfmverify**

The **nfmverify** utility verifies key generation certificates, allowing you to confirm how a particular security world and key are protected. The utility also verifies and returns some information about the security world and key.

The **nfmverify** utility only works for:

- Keys that were generated by **generatekey** or **nfmcmdadp** from nftcl component version 1.81.0 or later.
- Security worlds generated by **new-world** from sworld component version 1.4.0 or later, and keys generated by application programs and libraries that use the certificate storage features available from sworld component version 1.4.0 or later.
- Keys and security worlds generated on modules using firmware version 1.67.15 or later.

The **nfmverify** utility compares the details in the ACL of the key and those of the card set that currently protects the key; the card set listed in the key blobs.

A key that has been recovered to a different card set shows a discrepancy for every respect where the new cardset differs from the old one. For example, a key recovered from a 2-of-1 card set to a 1-of-1 card set, has a different card-set hash and a different number of cards, so two discrepancies are reported. The discrepancy is between the cardset mentioned in the ACL of the key, and the cardset that the key is currently protected by (i.e. the one mentioned in the key blobs). A key that has been transferred from another security world will show discrepancies and fail to be verified. nCipher recommend that keys are verified in the original security world at the time of generation.

If you need to replace your security world or card set, nCipher recommends that new keys are generated where possible. If you need to transfer a key, verification should be performed immediately prior to transfer as verification after the change of protection to a new security world, or card set, is troublesome.

**Usage:**

```
nfmverify [-f|--force] [-v|--verbose] [-U|--unverifiable] [-m|--module=MODULE]
[appname] [ident]
```

**Help Options:**

None

**Program Options:**

- **-m, --module=MODULE;** Performs checks with module *MODULE*
- **-f, --force;** Forces display of an output report that might be wrong
- **-U, --unverifiable;** Proceeds even if the security world is unverifiable

*Note:* If you need the **--unverifiable** option, there may be some serious problems with your security world.

- **-v, --verbose;** Prints full public keys and generation parameters
- **-C, --certificate;** Checks the original ACL for the key using key generation certificate (Default)
- **-L, --loaded;** Checks the ACL of a loaded key instead of the generation certificate
- **-R, --recovery;** Checks the ACL of the key loaded from the recovery blob.

**Output:**

Returns from **nfmverify** can take a variety of forms, depending on the parameters of the given key generation certificate, security world, and key concerned. Examples of possible output resulting from several different situations are provided below.

- **-h, --help** displays help for **nfmverify**
- **-V, --version** displays the version number of **nfmverify**
- **-u, --usage** displays a brief usage summary for **nfmverify**

Under normal circumstances, issuing a command of the form:

```
nfmverify --verbose --unverifiable myapp o20010621a13h25m02
```

returns output in the form:

```
-----
** [security world] **
1 Administrator Cards
(Currently in Module #1 Slot #0: Card #1)
Cardset recovery ENABLED
Passphrase recovery disabled
Strict FIPS 140-2 level 3 disabled
Generating module ESN 0A42-E645-7A75 currently #1 (in same
incarnation)
** [Application key myapp o20010621a13h25m02] **
[Named 'test Thu, 21 Jun 2001 13:25:02 +0100']
Usable by HOST applications.
Recovery ENABLED.
MODULE-ONLY protection
Type RSAPrivate 1024 bits sppkeygenparams.type= RSAPrivate 2
.params.rsaprivate.flags= none 0x00000000
.lenbits= 0x00000400 1024
.given_e absent
.nchecks absent

Generating module ESN 0A42-E645-7A75 currently #1 (in same
incarnation)
nCore hash 23a901f3329aa9e29cd79d3bb7b32d549b725fc3
public_half.type= RSAPublic 1
.data.rsapublic.e= 4 bytes
00010001

.n= 128 bytes
8a6ab219 183de558 48c8379e 840895ff 0ba64bae 392848c6 c0aeb7f9
d10b046d
4a214b70 4878b518 8e599c69 1cd61db0 bab4f852 425c70f5 b9c000e5
4ceda15f
c062b5dd 01852380 f70275a1 870a6947 68ef59f0 db5d2e84 d6ae8dc1
7542e94d
adedece8 cb3c9fb6 98fab8af 52c94137 a76ab7dd 38648134 0df55ca8
2f45e8b7

Verification successful, check details above.
-----
```

Output of this form indicates successful verification of the relevant key generation certificate.

The following examples indicate forms of output that could be returned if you try to verify the generation certificate of a key generated in a security world that was created with an insufficiently up-to-date version of nCipher support software. In such a case, issuing a command of the form:

```
nfkmverify --verbose myapp spong
```

returns output of the form:

```
PROBLEM: no world generation certificates
PROBLEM: application key myapp spong: no key generation signature
2 issues found, NOT VERIFIED
```

Adding the **--unverifiable** tag to the same command:

```
nfkmverify --verbose --unverifiable myapp spong
```

returns output in the form:

```
-----
PROBLEM: application key myapp spong: no key generation signature1
issues found, NOT VERIFIED
-----
```

Then, also adding the **--force** tag to this same command:

```
nfkmverify --force --verbose --unverifiable myapp spong
```

returns output in the form:

```
-----
PROBLEM: application key myapp spong: no key generation signature
PROBLEMS BUT FORCING POSSIBLY-WRONG OUTPUT

** [security world] **
UNVERIFIED security world !
proceeding anyway as requested
** [Application key myapp spong] **
[Not named]
Usable by HOST applications.
Recovery ENABLED.
MODULE-ONLY protection
1 issues found, NOT VERIFIED
```

## nopclearfail

The **nopclearfail** utility clears a module, puts a module into the error state, or retries a failed module.

*Note:* In order to recover a module from the error state, you must turn the power to the module off and then on. For internal modules, this may mean shutting down and then restarting your computer.

Retrying a module causes the nCipher server to attempt reconnecting to a module to which the connection has previously failed. Retrying a module may cure some bus errors. You cannot attempt to reconnect to a module if it is in the error state.

### Usage:

```
/opt/nfast/bin/nopclearfail [[-n|--no-op]]|[-c|--clear]]|[-f|--fail]]|[-r|--retry]]  
[[--a|--all]]| [-m|--module=MODULE]]
```

### Action Selection Options:

- **-n, --no-op;** Sends the NoOp command
- **-c, --clear;** Sends the ClearUnit command To use this option, you must be logged in
- **-f, --fail;** Sends the Fail command. To use this option, you must be logged in
- **-r, --retry;** Sends the RetryFailedModule command. To use this option, you must be logged in

### Module Selection Options:

- **--all;** Sends the command to all modules
- **--module=MODULE;** Sends the command to module *MODULE*

### Output:

If you select the **-no-op**, **--retry**, or **--clear** option, **nopclearfail** returns the following for each selected module:

```
Module 1, command NoOp: OK
```

If you select the **--fail** option and are logged in, the Fail command causes the module to enter the error state. The status LED will flash SOS D ( ... --- ... -. ). In order to reset the module, you must turn the power to the module off and then on again. If you are not logged in as such a user and you select the fail option, the Fail command will fail and the module will itself continue as normal.

**nvrnm-sw**

The **nvrnm-sw** utility views and modifies NVRAM areas.

**Usage:**

```
/opt/nfast/bin/nvrnm-sw --alloc [-m|--module=MODULE] [-s|--slot=SLOT]
[-b|--bytes=BYTES] [-n|--nvrnm-id=ID] [-k|--key=APPNAME,IDENT]

/opt/nfast/bin/nvrnm-sw --delete [-m|--module=MODULE] [-s|--slot=SLOT]
[-n|--nvrnm-id=ID]

/opt/nfast/bin/nvrnm-sw --write [-m|--module=MODULE] [-s|--slot=SLOT]
[-f|--file=FILE]

/opt/nfast/bin/nvrnm-sw --read [-m|--module=MODULE] [-s|--slot=SLOT]
[-f|--file=FILE]

/opt/nfast/bin/nvrnm-sw --delete-noadmin [-m|--module=MODULE] [-s|--slot=SLOT]
[-n|--nvrnm-id=ID]

/opt/nfast/bin/nvrnm-sw --acl [-m|--module=MODULE] [-s|--slot=SLOT]
[-n|--nvrnm-id=ID]

/opt/nfast/bin/nvrnm-sw --list [-m|--module=MODULE]
```

**Help Options:**

- **-h, --help;** Displays help for **nvrnm-sw**
- **-V, --version;** Displays the version number of **nvrnm-sw**
- **-u, --usage;** Displays a brief usage summary for **nvrnm-sw**

**Action Selection Options:**

- **-a, --alloc;** Allocates a new NVRAM area on the module specified in **--module=MODULE**. An Administrator Card Set must be inserted in the slot specified in **-slot=SLOT** to perform this action.
- **-d, --delete;** Deletes the NVRAM area on the module specified in **--module=MODULE**. An Administrator Card Set must be inserted in the slot specified in **-slot=SLOT** to perform this action.
- **-w, --write;** Writes data to the NVRAM area on the module specified in **--module=MODULE**. The data is written from the file specified in **--file=FILE**, if present. Otherwise it is written from stdout. If the ACL of the NVRAM area requires it, an Operator Card Set must be inserted in the slot specified in **-slot=SLOT** to perform this action. This action may not be permitted by the ACL of the NVRAM area.
- **-r, --read;** Reads data from the NVRAM area on the module specified in **--module=MODULE**. The data is written to the file specified in **--file=FILE**, if present. Otherwise it is written to stdout. If the ACL of the NVRAM area requires it, an Operator Card Set must be inserted in the slot specified in **-slot=SLOT** to perform this action. This action may not be permitted by the ACL of the NVRAM area.
- **-c, --delete-noadmin;** Deletes the NVRAM area on the module specified in **--module=MODULE** when no Administrator Card Set is required. If the ACL of the NVRAM

area requires it, an Operator Card Set must be inserted in the slot specified in **-slot=SLOT** to perform this action.

- **-i, --acl**; Displays the ACL of the NVRAM area on the module specified in **--module=MODULE**. If the ACL of the NVRAM area requires it, an Operator Card Set must be inserted in the slot specified in **-slot=SLOT** to perform this action.
- **-l, --list**; Lists the entire contents of the NVRAM.

**General Options:**

- **-m, --module=MODULE**; Specifies the module to use. The default is 1.
- **-s, --slot=SLOT**; Specifies the slot to use to read Administrator or Operator cards. The default is 0.
- **-v, --verbose**; Provides verbose output.
- **-x, --hex**; Specifies that hex notation is used for the *ID* value supplied to the **--nvram-id** option.

**Action Specific Options:**

- **-b, --bytes=BYTES**; Specifies the number of bytes to allocate for **--alloc** or to read for **--read**. The default is 100.
- **-n, --nvram-id=ID**; Specifies the identifier of the NVRAM file for the **--delete**, **--delete-noadmin** and **--acl** actions.
- **-f, --file=FILE**; Specifies file to be read or written for the **--read** and **--write** options. If this option is not specified for these actions, stdout is used by default.
- **-k, --key=APPNAME, IDENT**; Specifies a key during the **--alloc** action. This key is required for all subsequent **--read** or **--write** actions on the NVRAM area.

**pollbare**

The **pollbare** example utility returns information about state changes. Its functionality depends on whether the server or any module supports nCore API poll commands.

**Usage:**

```
/opt/nfast/bin/pollbare [-q|--quiet] [-t|--time=TIME] [-m|--module=MODULE]
```

**Options:**

- **-q, --quiet**; Specifies that **pollbare** run only once.
- **-t, --time=*TIME***; Specifies an update every *TIME* number of seconds. If you do not specify a value, the default is 1.
- **-m, --module=*MODULE***; Specifies that **pollbare** be applied only to a given module *MODULE*. If you do not specify a module, the default is all modules.

**pubkey-find**

The **pubkey-find** utility returns information about the contents of a .pem file, which may be a private key, a certificate or a certificate request.

**Usage:**

```
/opt/nfast/bin/pubkey-find [--cert|--certreq|--privkey|--auto] [--summary]
[--hash] [--fingerprint] [--thumbprint] [--identify] [--verify] [--info]
[--all|--latest|--earliest] - < CERT-OR-KEY-FILE /opt/nfast/bin/pubkey-find
[--cert|--certreq|--privkey|--auto] [--summary] [--hash]
[--fingerprint] [--thumbprint] [--identify] [--verify] [--info]
[--all|--latest|--earliest] [--nfkmverify-options OPTIONS]CERT-OR-KEY-FILE
```

In this command, *CERT-OR-KEY-FILE* specifies a .pem file.

**Input Format Options:**

- **--cert**; Specifies that the input file is a certificate.
- **--certreq**; Specifies that the input file is a certificate request.
- **--privkey**; Specifies that the input file is a private key.
- **--auto**; Specifies that the type of the input file is not known. **pubkey-find** attempts to identify the input format. This is the default if no input file type option is specified. The input file type should be specified if it is known.

**Output Format Options:**

- **--summary**; Generates summary output in human-readable format. Summary information includes the nCore hash and the identities of any matching key files in the kmdata area. If the file is in private key format but is actually an embed key indicator (or ssleay encapsulated blob), this is also reported.

The information displayed by the **--summary** option is the default if no output format options are specified. Summary information is not printed if another output format option is specified and **--summary** is not.

- **--hash**; Prints the nCore key-pair hash.
- **--fingerprint**; Prints the certificate's MD5 hash (its "fingerprint").
- **--thumbprint**; Prints the certificate's SHA-1 hash (its "thumbprint").
- **--identify**; Prints the key's security world *appname* and *idents*.

The **--hash**, **--fingerprint**, and **--thumbprint** options print not just the key, but the cryptographic checksums of the entire certificate.

**Further Processing Options:**

- **--verify**; Uses **nfkmverify** to verify that the key was securely generated.
- **--info**; Displays extensive general information about key (**nfkminfo -k**).

**Other Options:**

- **--all**; Reports and processes all key files in the kmdata directory that match *CERT-OR-KEY-FILE*. This is the default if none of **--all**, **--latest**, or **--earliest** is specified.
- **--latest**; Reports and processes the most recently modified file.
- **--earliest**; Reports and processes the file that has the earliest modification time.

- **--nfkmverify-option *OPTIONS***; Passes *OPTIONS* to **nfkmverify**. *OPTIONS* must be a Tcl list.

If several kmdata files match, for example if the key has been retargeted, **pubkey-find** processes all of them. The **--latest** and **--earliest** options can be used to select one file only to process. The **--earliest** option is likely to work best with **nfkmverify** (that is, the **-verify** option). The **--latest** option is likely to work best otherwise.

**Output:**

The **pubkey-find** utility returns information similar to the following:

```
-----
$ /opt/nfast/bin/pubkey-find cert.pem
input format cert
nCore hash 7e259da3d7e78d8ad0b9375e5b99d2d4ad0de7a8
no matching key in current security world host data area

$ /opt/nfast/bin/pubkey-find embed.pem
PEM 'key' file really contains only key indicator
input format privkey
nCore hash 750749f62b640132c4b05fa552f49fea3785d01a
name 'plainname'
appname embed
ident 1038030e0e3f21d709817e049eec575a33faa2f2

$ /opt/nfast/bin/pubkey-find embed_req.pem
input format certreq
nCore hash 750749f62b640132c4b05fa552f49fea3785d01a
name 'plainname'
appname embed
ident foo
name 'plainname'
appname embed
ident 1038030e0e3f21d709817e049eec575a33faa2f2

$ /opt/nfast/bin/pubkey-find --fingerprint embed_selfcert.pem
47:3d:8f:f8:2c:7a:64:23:f9:5a:53:c4:1c:39:3b:2c

$ /opt/nfast/bin/pubkey-find --thumbprint embed_selfcert.pem
52:ab:8c:fa:13:a6:ce:43:69:70:87:36:d1:eb:73:71:3e:ba:98:e8
-----
```

## **racs**

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

### **Usage:**

```
:/opt/nfast/bin/racs [-f|--force] [-m|--module=MODULE] [-s|--slot=SLOT]
```

### **Help Options:**

None

### **Other Options:**

- **-f, --force;** Tells **racs** to allow the use of smart cards that already contain data (any existing data will be erased). If a value for this flag is not specified, **racs** prompts you if a card contains data.
- **-m, --module=MODULE;** Specifies the ModuleID to use.
- **-s, --slot=SLOT;** Specifies the nCipher slot number for the smart card reader to use, as identified by **slotinfo**. If your module has a only one smart card reader, *SLOT* is 0.
- **-h, --help;** Displays help for **racs**
- **-V, --version;** Displays the version number of **racs**
- **-u, --usage;** Displays a brief usage summary for **racs**

**randchk**

The **randchk** utility runs an  $n$ -bit, or an  $n$ -to- $m$ -bit, universal statistical test on random numbers returned by the module.

**Usage:**

```
/opt/nfast/bin/randchk [L-min [L-max]]
```

If specified, the value of **L-min** is the number of the lowest bit to be used in the test and **L-max** is the number of the upper bit to be used in the test. These numbers must be between 1 and 16. If you do not enter a number, **randchk** performs a 6-bit test. You can specify a single number, **L-min**, or a range of numbers, **L-min** to **L-max**. If you specify numbers from **L-min** to **L-max** inclusive, **randchk** performs statistical tests for a range of random numbers with these lengths. The second number, **L-max**, must be larger than the first number, **L-min**.

**Output:**

The **randchk** utility displays a message similar to this:

```
Initializing 6-bit Universal Statistical Test
Processing...
46K
6-bit test results: 5.221070 = 0.003365 from mean (0.507348
s.d.'s)
```

To use this command, you must be logged in as Administrator.

**sigtest**

The **sigtest** utility measures the module speed using RSA or DSA signatures or signature verifications.

**Usage:**

```
/opt/nfast/bin/sigtest [[-s|--sign] | [-v|--verify]][-d|--decrypt]]
[-m|--module=MODULE] [-j|--outstanding-jobs=COUNT] [-t|--stop-after=TIME]
[-n|--jobs-count=COUNT] [-S|--sig-type=TYPE] [-l|--key-size=BITS]
[-M|--mech=MECH] [-p|--plain=TYPE]
[[[-K|--skew-check=SKEW ]] [-T|--min-check=COUNT ]] [-C|--check-start=TIME ]]
[--overprint][-B|--unbuffered-stdout][-o|--output=FILE] [-r|--report-inter-
val=TIME}
```

**Program Options:**

- **-s, --sign;** Tests the Sign operation. This is the default if no program option is specified.
- **-v, --verify;** Tests the Verify operation.
- **-d, --decrypt;** Tests the Decrypt operation.
- **-m, --module=MODULE;** Specifies the module number. Multiple modules may be specified, for example, “--module=1 --module=2”. If no module is specified, all module are tested by default.
- **-j, --outstanding-jobs=COUNT;** Sets the maximum number of outstanding jobs. The default value of *COUNT* is the maximum number of jobs recommended for the hardserver, plus one.
- **-t, --stop-after=TIME;** Specifies the maximum time to run the test. Use the suffix s to specify seconds, m minutes, h hours and d days. The default value of *TIME* is infinity, that is, the test runs forever.
- **-n, --jobs-count=COUNT;** Specifies the maximum number of jobs to run. The default value of *COUNT* is infinity, that is, there is no limit to the number of jobs run.

**Key Options:**

- **-S, --sig-type=TYPE;** Selects the signature type to use. Valid values are RSA, DSA and KCDSA. The default is RSA.
- **-l, --key-size=BITS;** Sets the key size in bits. The default is 1024.
- **-M, --mech=MECH;** Specifies the mechanism to use. Valid values are RSAPKCS1, DSA and KCDSAHAS160.
- **-p, --plain=TYPE;** Uses the plaintext type *TYPE*. Valid values are Bignum, Hash and Bytes.

*Note:* The KCDSA algorithm and the KCDSAHAS160 mechanism are only available if you have enable the KCDSA feature.

**Automatic Checking Options:**

- **-T, --min-check=COUNT;** Performs threshold checking, starting after either 15 seconds or the time specified by **--min-check**. Subsequently, when **floodtest** writes an output line, it quits with an error message if the overall average number of modular exponentiations each second drops below *COUNT*.
- **-K --skew-check=SKEW;** Performs skew checking, after either 15 seconds or the time specified by **--min-check**. **floodtest** records the overall average number of modular expo-

mentiations each second as average. Subsequently, when **floodtest** writes an output line, it quits with an error message if the average is outside the range  $\text{average} \pm \text{SKEW}$ .

**Note:** The **--min-check** and **--skew-check** options are mutually exclusive.

- **-C, --check-start=TIME;** Specifies the time in seconds at which threshold or skew checking starts. The default value of *TIME* is 15.

**Output Options:**

- **--overprint;** Prints results all on one line, using `\r` rather than `\n`.
- **-B, --unbuffered-stdout;** Turns off buffering for stdout (if the `-output` option is specified, `--unbuffered-stdout` also turns off buffering for the output stream)
- **-o, --output=FILE;** Specifies that results be written to *FILE* in addition to stdout.
- **-r, --report-interval=TIME;** Displays, at the specified interval of *TIME* seconds, the total number of exponentiations achieved and the rate at which they are performed. The default value of *TIME* is 1.

**Output:**

The **sigtest** utility produces output similar to this:

```
number of modules: 1
Making 1024-bit RSA Private key on module #1...
1, 148 59.2, 151 overall
2, 410 140.32, 206 overall
3, 677 190.992, 226 overall
4, 944 221.395, 267 overall
5, 1190 231.237, 256.5 overall...
```

The first column shows the number of seconds. The second shows the total number of exponentiations performed. The third column shows the number of exponentiations achieved this second. The last column shows a moving average of the number of exponentiations achieved each second.

## slotinfo

The **slotinfo** utility returns information about the tokens that are present in a module. **slotinfo** can also be used to format a token.

### Usage:

```
/opt/nfast/bin/slotinfo -m|--module=MODULE [-s|--slot=SLOT]  
/opt/nfast/bin/slotinfo -f|--format -m|--module=MODULE -s|--slot=SLOT
```

### Module Selection:

- **-f, --format;** Tells **slotinfo** to format the token that is currently in the slot. You must specify a slot number if you want to format a token. The token is formatted without a challenge response key.
- **-m, --module=MODULE;** Sets the module number of the module to be used. You must specify a module number.
- **-s, --slot=SLOT;** Sets the number of the slot to be used. If this flag is set, **slotinfo** returns information about the files on the token. If, however, you do not specify **--slot**, **slotinfo** returns information about all slots. The **slotinfo** utility does not update the security world host data. If you are using an nCipher security world:
  - Use createocs to format Operator Cards
  - Use createocs to erase cards.

### Output:

The **slotinfo** utility `--module=1` returns information in the format:

```
Slot Type Token IC Flags Details  
#0 Smartcard present 1 A  
#1 Software Tkn - 0
```

```
slotinfo --module=1 --slot=0 returns information in the format:  
Module 1 slot 0:  
Authentication key: 00000000-00000000-00000000-00000000-00000000  
No data on token  
3698 bytes free  
Formatting token in module 1 slot 0:  
Formatted token OK
```

**stattree**

The **stattree** utility returns the statistics gathered by the nCipher server and modules.

**Usage:**

```
/opt/nfast/bin/stattree
```

**Output:**

Running the **stattree** utility displays a snapshot of all 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 module.

Statistics are displayed in the form of a tree. At each node in the tree, either a set of statistics or a list of sub-categories is displayed. Each node has a label which consists of one of the following:

- A tag that identifies its contents
- A number that corresponds to an instance in the category, for example, a module identifier or a client connection identifier. Times are listed in seconds. Other numbers are integers, which are either real number 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 4
-ObjectsCreated 4
-ObjectsDestroyed 0
+ModuleEnvStats:
-MemTotal 14389248
-MemAllocKernel 86016
-MemAllocUser 0
-CurrentTempC 39.50
-MaxTempC 39.50
-MinTempC 39.50
```

*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.

**Node tags**

Statistics for each module. Requires firmware release 1.50 or later.

- *ModuleJobStats* are statistics related to the nCipher commands handled by this module
- *ModuleSCSIStats*, *nCipher SCSI modules only*, are statistics related to the SCSI connection between this module and the host computer

**Categories**

**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 modules. Commands processed internally by the server are not included here. The Uptime statistic gives the total running time of the server so far.

**Connections:** Statistics for connections between clients and the hardserver. There is one node for each currently active connection. Each node has an instance number that matches the log message generated by the server when that client connected. For example, when the hardserver message is “Information: New client #24 connected”, the client’s statistics appear under node #24 in the **stattree** output.

**PerModule:** Statistics kept by the modules. There is one instance node for each module, numbered using the standard module numbering. The statistics provided by each module depend on the module type and firmware version.

**ModuleJobStats:** Statistics for the commands (jobs) processed by a module. Appears under the Permodule category ModuleSCSIStats Statistics from the module’s SCSI interface. Appears only on SCSI-interfaced modules.

**ModuleObjStats:** Statistics for the module’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 module’s operating environment.

**Statistics IDs:**

**Uptime:** The length of time (in seconds) since a module 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 a module. Contains the number of commands currently being processed.

**ReplyCount:** The total number of replies returned from server to client, or from module to server.

**CmdBytes:** The total length of all the command blocks sent for processing.

**ReplyBytes:** The total length of all the reply block received after completion.

**CmdMarshalErrors:** The number of times a command block was not understood when it was received. A non-zero 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 module that the module 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 non-zero 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 module that the module 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 a module after it has failed. The hardserver provides a “Notice” message when this occurs. The message does not indicate that the attempt was successful.

**QOutstanding:** The number of commands waiting for a module to become available on the specified client connection. When a module 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 modules. When a module accepts a command from a client, QOutstanding decreases by 1 and this number increases by 1. Commands that are processed purely by the server are never included in this count.

**HostWriteCount:** The number of write operations (used to submit new commands) that have been received by the module 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 write data from the host was rejected by the module. A non-zero value may indicate that data is being corrupted in transfer, or that the hardserver/device driver has got out of sync with the module’s interface.

**HostWriteBadData:** Not currently reported by the module. Attempts to write bad data to the module are reflected in *HostWriteErrors*.

**HostWriteOverruns:** Not currently reported by the module. Write overruns are reflected in *HostWriteErrors*.

**HostWriteNoMemory:** Not currently reported by the module. Write failures due to lack of memory are reflected in *HostWriteErrors*.

**HostReadCount:** The number of times a read operation to the module was attempted. The module can defer a read if it has no replies at the time, but expects some to be available later. Typically the module 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 *ModuleSCSIStats* or *ModulePCISStats* counts this as one operation.

**HostReadErrors:** The number of times a read to a module failed because the parameters supplied with the read were incorrect. A non-zero value here typically indicates some problem with the host interface or device driver.

**HostReadEmpty:** The number of times a read from the module returned no data because there were no commands waiting for completion. In general, this only happens a small number of times during module startup or reset. It can also happen if *PauseForNotifications* is disabled.

**HostReadUnderruns:** Not currently reported by the module.

**HostReadDeferred:** The number of times a read operation to the module was suspended because it was waiting for more replies to become available. When the module is working at full capacity, a sizeable proportion of the total reads are likely to be deferred.

**HostReadTerminated:** The number of times a module had to cancel a read operation which has been deferred. This normally happens only if the clear key is pressed while the module is executing commands. Otherwise it might indicate a device driver, interface, or firmware problem.

**PFNIssued:** The number of PauseForNotifications commands accepted by the module 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 will not occur.

**PFNRejected:** The number of PauseForNotifications commands rejected by the module when received from the hardserver. This can happen during module 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 module. Normally, this is one less than the *PFNIssued* figure, since there is normally one such command outstanding.

**ANIssued:** The number of Asynchronous Notification messages issued by the module to the hardserver. These messages indicate such things as the clear key being pressed and the module 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 module. The fast channel facility is unsupported on current modules. This number should always be zero.

**ChanJobsCompleted:** The number of fast channel jobs completed by the module. The fast channel facility is unsupported on current modules. This number should always be zero.

**CPULoadPercent:** The current processing load on the module, represented as a number between 0 and 100. Because a module 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, modules 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 modules, the total number of interrupts received from the host. On current modules, 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 zero on current modules.

**HostDebugIRQs:** On PCI modules, the number of 'debug' interrupts received. This is used only for driver testing, and should be zero in any production environment.

**HostUnhandledIRQs:** On PCI modules, the number of unidentified interrupts from the host. If this is non-zero, a driver or PCI bus problem is likely.

**HostReadReconnect:** On PCI modules, 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.

**SCSIConnections:** The number of times a SCSI module has been successfully selected as a target.

**SCSICommands:** The total number of SCSI commands (including Read, Write, and Inquiry) that have been issued to the module.

**SCSIInquiries:** The number of SCSI Inquiry commands that have been sent to the module. A host typically sends a SCSI Inquiry command searching the SCSI bus for devices, for example, at startup.

**SCSIDisconnects:** The number of SCSI bus disconnects issued to the host by the module. A SCSI disconnect is issued whenever a read is deferred.

**SCSIReconnects:** The number of reconnections attempted by the module after a SCSI disconnect. This should be the same as SCSIDisconnects, or one less if the bus is currently disconnected.

**SCSILUN0Use:** The number of times SCSI LUN 0 was specified when the host connected to the module. A host is normally configured to use LUN 0 for Write commands.

**SCSILUN1Use:** The number of times SCSI LUN 1 was specified when the host connected to the module. Normally a host will be configured to use LUN 1 for Read commands, in order to allow writes to take place (on LUN 0) when a read is in operation. If this is zero, it is possible that the host has a SCSI interface which does not support multiple LUNs correctly. This will give performance problems - see the nFast troubleshooting guide.

**SCSICmdErrors:** The number of times an error was sent in response to a SCSI command by the module.

**SCSIBusResets:** The number of SCSI bus reset conditions issued by the host. If this occurs other than at start-up, it may indicate a serious error condition has been detected by the SCSI driver.

**SCSICtrlErrors:** The number of times the SCSI controller in the module reported various sorts of error. If non-zero, indicates either a SCSI cabling and termination problem, or a faulty module.

**SCSITagQUse:** The number of times SCSI Tagged Queueing was used when the host selected the module as target. If the host supports tagged queueing correctly, it does not need to use multiple LUNs for reads and writes. (The module offers both tagged queueing and multiple LUN support; it is up to the host to choose either or both of these as options when giving SCSI commands).

**SCSIReconFailures:** The number of times a SCSIReconnect operation ended in failure. If this is non-zero, SCSI bus cabling and termination could be at fault, or possibly the host's SCSI adapter or driver.

**SCSIWideNeg:** The number of times the SCSI 'Wide' option was negotiated between host and module. This is non-zero if both sides are Wide SCSI devices and are configured to allow wide data transfers.

**SCSISyncNeg:** The number of times Synchronous SCSI data transfer was negotiated between host and module. This is non-zero if both sides are Synchronous SCSI devices and are configured to allow synchronous data transfers.

**ObjectsCreated:** The number of times a new object has been put into the object store. This appears under the module's ModuleObjStats node.

**ObjectsDestroyed:** The number of items in the module's object store that have been deleted and their corresponding memory released.

**ObjectCount:** The current number of objects (keys, logical tokens, buffers) in the object store. This is equal to ObjectsCreated minus ObjectsDestroyed. An 'empty' module contains a small number of objects that are always present.

**CurrentTempC:** The current temperature (in degrees Celsius) of the module main circuit board. First-generation modules do not have a temperature sensor and do not return temperature statistics.

**MaxTempC:** The maximum temperature recorded by the module's temperature sensor. This is stored in non-volatile memory, which is cleared only when the unit is initialised. First-generation modules do not have a temperature sensor and do not return temperature statistics.

**MinTempC:** The minimum temperature recorded by the module's temperature sensor. This is stored in non-volatile memory, which is cleared only when the unit is initialised. First-generation modules 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 module. This is the installed RAM size, minus various fixed overheads.

**MemAllocKernel:** The total amount of RAM allocated for kernel use in a module. 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 module.

**Options:**

- **--is-machine;** Specifies SEE machine signing mode.
- **--machine-key=HASH;** Specifies the keyhash of the SEE machine for which this signature is good.
- **--ncipher-jym=HASH;** This is similar to --machine-key, but also checks that HASH is a known nCipher machine signing hash.
- **--machine-key-ident=IDENT;** Retrieves the hash of key *IDENT* then behaves like --machine-key=HASH. Only one machine key specification option can be specified.
- **--non-interactive;** Sets non-interactive mode. The program fails instead of requesting that required cardsets are loaded.
- **--show-metadata;** Shows image metadata before signing.
- **--verbose;** Increases the verbosity level. This option can be repeated.
- **--quiet;** Decreases the verbosity level. This option can be repeated.

**with-nfast**

The **with-nfast** utility enables keys to be loaded onto a module before an application is run. The application can be run immediately in the same session, or the first session can be paused and the application can be run in a separate session. Additional keys can be loaded while a session is running.

For applications that do not support K-of-N Operator Card Sets, the **with-nfast** utility allows K-of-N Operator Card Sets and module-protected keys to be loaded before the application is run. Some command-line options may cause **with-nfast** to function interactively by prompting for Operator Cards.

*Note:* Currently, **with-nfast** can be used only with command-line utilities.

**Usage:**

```
/opt/nfast/bin/with-nfast [-m|--module=MODULE] [-c|--cs-name=OCS-NAME]
[-C|--cs-hash=OCS-HASH] [-M|--module-prot] [-r|--multiple-cardsets]
[-i|--interactive] [-k|--key-name=KEY-NAME] [-A|--appname=APPNAME]
[-K|--key-ident=KEY-ID] [--admin=TOKEN] [-t|--tokens-only]
[-f|--preload-file=FILE] [program [argument ...]] | pause | exit
```

**Module Selection Options:**

- **-m, --module=MODULE;** Selects the module to be used. Repeating **--module=MODULE** enables a number of selected modules to be used. By default, **with-nfast** loads on all usable modules.

**Card Set Selection Options:**

- **-c, --cs-name=OCS-NAME;** Loads an Operator Card Set or Sets whose name or names match the given name or pattern. Card set patterns or names are treated as global patterns if they contain any of the characters \*, ?, or [..
- **-C, --cs-hash=OCS-HASH;** Loads the Operator Card Set with the given ID.
- **-M, --module-prot;** Loads or offers module-protected keys. You can use **--module-prot** together with other options to load additional keys or tokens.
- **-r, --multiple-cardsets;** Interactively loads multiple Operator Card Sets. If more than one Operator Card Set is to be loaded (in addition to any module-protected keys) and neither the **--cs-name** nor **--cs-hash** flag is set, then you must include **--multiple-cardsets**.

*Note:* If you do not specify any of **--cs-name**, **--cs-hash**, or **--multiple-cardsets**, **with-nfast** loads Operator Card Sets interactively. If you do not specify **--multiple-cardsets**, **with-nfast** loads only one Operator Card Set. By default, if no card set selection options are used, **with-nfast** loads the Operator Card Sets that are present in the selected modules.

**Key Selection Options:**

- **-i, --interactive;** Allows you to specify the keys to load interactively. This option is not currently implemented.
- **-k, --key-name=KEY-NAME;** Loads all keys whose names include or match **KEY-NAME**. If **KEY-NAME** contains any of the characters \*, ? or [, it is treated as a glob wildcard pattern. You can set multiple instances of **--key-name** to load keys that match more than one pattern. Use the **nfkminfo** command-line utility to identify keys and cards that are used by particular applications.

- **-A, --appname=APPNAME**; Specifies the application to be used by a subsequent **-K** option. You can set multiple instances of the **--appname** flag to specify more than one application. Examples of *APPNAME* include:
  - custom
  - embed
  - hwcrhk
  - iis34
  - netscape
  - simple
  - ssleay
- **-K, --key-ident=KEY-ID**; Loads the key with the specified ID and the application previously specified by the **--appname** option. You can set multiple instances of the **--key-ident** flag to load more than one key.

Use the **nfkminfo** command-line utility in order to identify keys that are used by particular applications.

**Note:** If you use **--key-name** or **--key-ident** to specify keys and do not explicitly select Operator Card Sets with **--cs-name**, **--cs-hash**, or **--module-prot**, then **with-nfast** loads only the Operator Card Sets that are required by those keys. By default, if no key selection options are used, **with-nfast** loads all keys on the loaded Operator Card Set, and, if **--module-prot** is specified, all module-protected keys.

- **--admin=TOKEN**; Loads the administrator keys and tokens specified in *TOKEN*. Specify all to load all administrator keys and tokens.
- **-t, --tokens-only**; Loads tokens only and no keys.

**Note:** If both **--tokens-only** and **--module** are specified, nothing is loaded into the modules. This is useful if you want to allow an application to access objects loaded by another, still running instance of **with-nfast**.

#### Other Options:

- **-f, --preload-file=FILE**; Specifies the file to be used to share the keys. If no file is specified, the value of the environment variable `NFAST_NFKM_TOKENSFILE` is used (if this variable has been set). If no file is specified and `NFAST_NFKM_TOKENSFILE` has not been set, the default file `/opt/nfast/kmdata/preload/default` is used.

If the preload file does not exist, **with-nfast** creates it, making it readable and writable only by the current user. (If **with-nfast** uses the default preload file, it creates the directory `.../kmdata/preload`, if necessary, making it also readable and writable only by the current user.) The **with-nfast** program does not change the permissions of a file that already exists. If multiple users require access to preloaded keys, they must do one of the following:

- Instruct **with-nfast** to use distinct existing preload files, by using either the **--preload-file** option, or the `NFAST_NFKM_TOKENSFILE` environment variable
- Set up a file configured with appropriate permissions for all of the users who will share it, and instruct **with-nfast** to use that by using either the **--preload-file** option, or the `NFAST_NFKM_TOKENSFILE` environment variable.

Users who can read the preload file can use all keys loaded by all applications that use the preload file, not just the preloaded keys listed in that file. Hence, it is important to restrict access to a preload file to only those users who need it

**Note:** The file used to share keys contains binary information maintained by the **with-nfast** utility.

**Note:** When using **with-nfast** to load non-persistent tokens, you must ensure that you have enough remaining Operator Cards in order to load the token onto the final module. Usually, this means you need one or more cards than you have modules, depending on the circumstances. For example, if you have four modules and K is 3, then N must be at least 6 so that there will be at least three cards remaining to load the token onto the final module.

- **program;** This option runs an application.
- **argument;** This option specifies any parameters that are required by *program*. More than one instance of *argument* may be required, depending on the application. For further information, refer to the relevant documentation for the application.
- **pause;** Pauses the session. You can use this feature to load keys in one session and use them in another.
- **exit;** Exit causes **with-nfast** to terminate after loading keys or cards (or both). This functionality can be useful in order to add keys or cards to the card set loaded and held by a still running instance of **with-nfast**.

#### **Environment Variables used with-nfast**

The environment variable NFAST\_NFKM\_TOKENSFILE holds the name of the file used to share ClientID and KeyIDs of the loaded objects. **with-nfast** sets this variable to a default value, `/opt/nfast/kmdata/preload/default`, so you do not normally need to set it. The environment variable NFAST\_NFKM\_TOKENSSELECT overrides the *default* at the end of the default value of NFAST\_NFKM\_TOKENSFILE.

Use the NFAST\_NFKM\_TOKENSFILE and NFAST\_NFKM\_TOKENSSELECT environment variables to ensure that the file containing the ClientID and KeyIDs of the loaded objects can be accessed by both **with-nfast** and the application. If you have set the value of NFAST\_HOME to a value other than `/opt/nfast/`, this overrides the `/opt/nfast/` at the beginning of the default value of NFAST\_NFKM\_TOKENSFILE.

#### **Output:**

If you want to load keys from a number of Operator Card Sets in module 2 and use them immediately in an application, for example, with **nfkminfo**, enter the following:

```
/opt/nfast/bin/with-nfast -r -m 2 nfkminfo
```

If there is no card in the specified module, **with-nfast** displays the following message:

```
-----
Loading cardset(s) on module #2.
Currently inserted card(s) are:
Slot#0:
Enter slot number to load from, or change cards and press
return, or type 'd' to load no cards on this module.
-----
```

Insert a card in the smart card reader, and **with-nfast** displays:

```
-----  
Loading cardset(s) on module #2. Currently inserted card(s)  
are:  
Slot#0: Operator Card 'alpha' #1  
Enter slot number to load from, or change cards and press  
return, or type 'd' to load no cards on this module.  
-----
```

Input 0 for the slot number, and **with-nfast** displays:

```
-----  
Loading cardset 'alpha' in module #2 ...  
Loaded cardset 'alpha' in module #2.  
Loading ssleay 'www.ncipher.com' key(RSAPrivate) done.  
Loading hwcrhk rsa-g key(RSAPrivate) done.  
Loading simple 'for recovery' key(RSAPrivate) done.  
Loading simple 'tg card recovery' key(RSAPrivate) done.  
Loading simple foo key(DSAPrivate) done.  
Loading pkcs11 'Private Key Object Label' key(RSAPrivate)extra1(Random) done.  
Loading simple 't2' key(RSAPrivate) done.  
Loading simple 'nofipstest' key(DHPrivate) done.  
Loading hwcrhk rsa-drhibbert8 key(RSAPrivate) done.  
When the card set has loaded, with-nfast displays:  
Loading cardset(s) on module #2. Currently inserted card(s) are:  
Slot#0: Operator Card 'alpha' #1  
Enter slot number to load from, or change cards and press return, or type 'd' if  
done loading cards on this module.  
-----
```

Change the card, and press and **with-nfast** displays:

```
-----  
Loading cardset(s) on module #2. Currently inserted card(s) are:  
Slot#0: Operator Card 'alpha' #3  
Enter slot number to load from, or change cards and press return, or type 'd' if  
done loading cards on this module.  
-----
```

Input 0 for the slot number, and **with-nfast** displays:

```
-----  
Loading cardset 'alpha' in module #2 ...  
...  
-----
```

Continue inserting cards into the module until all cards are loaded. Then press Enter followed by d to indicate that you are done.

The **with-nfast** utility then runs **nfkminfo**, which lists the pre-loaded objects.

**Note:** Objects remain loaded only so long as at least one application using them maintains a connection to the hardserver. In the case of the example above, the object will be unloaded after **nfkminfo** and **with-nfast** finish.

If you want to load a particular key set (here, *rsa-g*) for a specified application (here, *hwcrhk*) onto all available modules for subsequent use in another session, input:

```
/opt/nfast/bin/with-nfast -A hwcrhk -K rsa-g pause
```

The system replies with:

```
-----
Loading tokens and/or keys on Module#1, ESN xxxx-xxxx-xxxx
Loading cardset 'alpha' in module #1 ...
1 card(s) required. Insert a card (or say 'd').
Loaded cardset 'alpha' in module #1.
Loading hwcrhk rsa-g key(RSAPrivate) done.
Loading tokens and/or keys on Module#2, ESN xxxx-xxxx-xxxx
Loading cardset 'alpha' in module #2 ...
1 card(s) required. Insert a card (or say 'd').
Loaded cardset 'alpha' in module #2.
Loading hwcrhk rsa-g key(RSAPrivate) done.
Cards/keys loaded.
-----
```

The keys are loaded onto the two available modules, and the ClientID and KeyIDs of the loaded objects are stored for the subsequent session to access. In order to use the loaded keys with another application, do the following:

1. Run **with-nfast** with the required options and the pause option to load the keys in one window.
- 2: In a second window, run **with-nfast** with the command to start the second application.
- 3: When each module is listed, enter d. You do not need to perform any other tasks with **with-nfast** in this window.

For example, to load keys for generatekey that are already loaded for another application, input the following in a second window:

```
/opt/nfast/bin/with-nfast generatekey hwcrhk
```

The system replies with:

```
-----
Loading tokens and/or keys on Module#1, ESN xxxx-xxxx-xxxx
Loading tokens and/or keys on Module#2, ESN xxxx-xxxx-xxxx
Executing generatekey ...
-----
```

In this example, the keys and cards will remain loaded at least until **with-nfast** pause is terminated in the first window.

