



nShield Security World

nShield Connect v12.81 User Guide (Linux)

12 July 2024

Table of Contents

1. Introduction	1
1.1. Read this guide if	1
1.2. Terminology	1
1.3. Model numbers	1
1.4. Security World Software	2
1.4.1. Software architecture	2
1.4.2. Default directories	4
1.4.3. Utility help options	5
1.5. Further information	5
1.6. Security advisories	5
1.7. Recycling and disposal information	6
2. Security Worlds	7
2.1. Security	8
2.1.1. Smart cards	8
2.1.2. Remote Operator	9
2.1.3. Remote Administration	10
2.1.4. Security World and an Connect	10
2.1.5. NIST SP800-131A	11
2.1.6. FIPS 140-2 compliance	12
2.1.7. Common Criteria compliance	12
2.2. Platform independence	13
2.3. Application independence	13
2.4. Flexibility	14
2.4.1. Using the Security World key: module-protected keys	14
2.4.2. Using Operator Card Sets: OCS-protected keys	15
2.4.3. Using pass phrases for extra security	18
2.4.4. Using softcard-protected keys	19
2.4.5. NVRAM key storage	20
2.5. Scalability	21
2.5.1. Load-sharing	21
2.6. Robustness	22
2.6.1. Backup and recovery	22
2.6.2. Replacing a hardware security module	23
2.6.3. Replacing the Administrator Card Set	23
2.6.4. Replacing an Operator Card Set or recovering keys to softcards	24
2.7. Audit Logging	25
2.8. KeySafe and Security Worlds	26

2.9. Applications and Security Worlds	27
2.10. The nShield PKCS #11 library and Security Worlds	27
2.11. Risks	27
3. The nShield Connect user interface	29
3.1. Front panel controls	29
3.2. Display screen and controls	29
3.2.1. Menu screens	30
3.2.2. Dialogs	31
3.2.3. Information display	31
3.3. Using the front panel controls	32
3.3.1. Start-up information	32
3.3.2. Administrative control of the unit	32
3.3.3. Viewing the current status of the unit	34
3.3.4. Viewing the mode of the unit	34
3.4. Using a keyboard to control the unit	34
4. Physical security of the nShield Connect	36
4.1. Tamper event	36
4.1.1. Connect lid is closed	36
4.1.2. Connect lid is open	37
4.2. Physical security checks	38
4.3. Replacing the fan tray module and PSU	40
4.3.1. Replacing the fan tray module	41
4.3.2. Replacing the PSU	42
4.3.3. Battery life when storing the Connect	42
4.4. Disabling tamper detection functionality	42
4.4.1. Reactivating disabled tamper functionality	43
5. Software installation	44
5.1. After software installation	44
6. Client Software and module configuration	46
6.1. About user privileges	46
6.2. About client configuration	46
6.2.1. Remote file system (RFS)	47
6.2.2. HSM configuration	47
6.2.3. Client configuration	48
6.3. Basic HSM and remote file system (RFS) configuration	48
6.3.1. Configuring the Ethernet interfaces	49
6.3.2. Optionally configure hardserver interfaces	49
6.3.3. Configuring the Remote File System (RFS)	50
6.3.4. Configuring log file storage	52

6.3.5. Setting the time and date	53
6.3.6. Keyboard layout	54
6.4. Configuring the Connect to use the client	54
6.4.1. Remote configuration of additional clients	58
6.5. Configuring client computers to use the Connect	60
6.5.1. Enrolling the client with the client configuration file	60
6.5.2. Enrolling the client from the command line	61
6.5.3. Client configuration utilities	62
6.6. Configuring NTP in the Connect	64
6.6.1. Using the NTP configuration tool	64
6.6.2. Restarting the Connect	65
6.7. Configuring Remote Syslog	66
6.8. Setting up client cooperation	67
6.8.1. Useful utilities	69
6.8.2. Setting environmental variables	71
6.8.3. Logging and debugging	71
6.8.4. Configuring Java support for KeySafe	72
6.9. Routing	72
6.9.1. Testing routes	72
6.9.2. Tracing network routes	74
6.10. Configuring an Connect using the Serial Console	76
6.10.1. Serial port configuration	76
6.10.2. Creating a serial console session	77
6.10.3. Enabling and disabling the serial console	78
6.10.4. Serial console commands	78
6.11. Enabling optional features	80
6.11.1. Available optional features	80
6.11.2. Ordering additional features	85
6.11.3. Enabling features	86
6.11.4. Using multiple modules	87
6.12. Stopping and restarting the hardserver	89
6.13. Resetting and testing the Connect	89
6.13.1. Default configuration	89
6.13.2. Factory state	89
6.13.3. Testing the installation	90
7. Security World Remote Administration	92
7.1. Remote Administration components	92
7.1.1. Remote Administration software	92
7.1.2. Security World programs and utilities	93

7.1.3. nShield Remote Administration smart cards	93
7.2. Authorized Card List	95
7.3. Remote Administration Client	96
7.4. Remote Administration Service	96
7.5. nShield Trusted Verification Device	97
7.6. Software installation	98
7.6.1. The Remote Administration Service with the Connect or Connect XC	98
7.6.2. Remote Administration Service bundle	99
7.6.3. Remote Administration Client	99
7.6.4. TVD	99
7.7. System configuration	99
7.7.1. Remote Administration Service port	99
7.7.2. Stopping and restarting the Remote Administration Service	100
7.7.3. Firewall settings	100
7.8. Configuring auto push	100
7.9. Configuring Dynamic slots	101
7.10. Privileged client	102
7.11. Using nethsmadmin to reboot an Connect	102
7.11.1. Enabling and disabling remote reboot	103
7.11.2. Enabling and disabling remote reboot using the module configuration file	103
7.11.3. Enabling and disabling remote reboot using the front panel of the Connect	103
7.11.4. Enabling and disabling remote mode change	103
7.11.5. Enabling and disabling remote upgrade	104
7.12. Adjusting card removal detection timers to account for network characteristics	105
7.13. Using Remote Administration with applications requiring cards in slot 0	105
7.14. Authorized Card List	106
7.14.1. Adding cards to the Authorized Card List	107
7.15. Using Remote Administration	107
7.15.1. Presenting nShield Remote Administration smart cards using the Remote Administration Client	107
7.15.2. Configuring the Connect with configuration files	108
7.15.3. Remote Administration Configuration file sections	109
8. Creating and managing a Security World	112
8.1. Creating a Security World	112
8.1.1. The creation process	112
8.1.2. Security World Files	113
8.1.3. Security World options	115

8.1.4. Creating a Security World using the Connect front panel	120
8.1.5. Creating a Security World using new-world	123
8.1.6. After you have created a Security World	131
8.2. Displaying information about your Security World	131
8.2.1. Displaying information about a Security World with nfkminfo	132
8.2.2. Displaying information about a Security World with kmfile-dump	132
8.3. Adding or restoring an HSM to the Security World	133
8.3.1. Adding an HSM to a Security World using the Connect front panel	134
8.3.2. Adding an HSM to a Security World with new-world	134
8.4. Security World migration	135
8.4.1. Pre-requisites for migrating keys	136
8.4.2. Restrictions on migrating keys	136
8.4.3. About the migration utility	137
8.4.4. Migrating keys	138
8.4.5. Migrating keys process	139
8.4.6. Verifying the integrity of the migrated keys	140
8.4.7. Troubleshooting	140
8.5. Migrating KMDATA	142
8.6. Erasing a module from a Security World	143
8.6.1. Erasing a module from the unit front panel	144
8.6.2. Erasing a module with new-world	144
8.6.3. Erasing a module with KeySafe	144
8.6.4. Erasing a module with initunit	145
8.7. Replacing an existing Security World	145
8.8. Deleting a Security World	146
8.8.1. Deleting the Security World using the Connect front panel	147
9. Managing card sets and softcards	148
9.1. Creating Operator Card Sets (OCSs)	149
9.1.1. Persistent Operator Card Sets	149
9.1.2. Time-outs	150
9.1.3. FIPS 140-2 Level 3-compliant Security Worlds	150
9.1.4. Creating an Operator Card Set using the nShield Connect front panel	150
9.1.5. Creating an Operator Card Set using the command line	151
9.1.6. Creating an Operator Card Set with KeySafe	154
9.2. Creating softcards	156
9.2.1. Creating a softcard with ppmk	157
9.2.2. Creating softcards with KeySafe	157
9.3. Erasing cards and softcards	158
9.3.1. FIPS 140-2 Level 3-compliant Security Worlds	159

9.3.2. Erasing card sets using the Connect front panel	159
9.3.3. Erasing cards with KeySafe	160
9.3.4. Erasing cards using the command line	160
9.3.5. Erasing softcards	161
9.4. Viewing cards and softcards	162
9.4.1. Viewing card sets using the Connect front panel	162
9.4.2. Viewing card sets with KeySafe	163
9.4.3. Viewing card sets using the command line	164
9.4.4. Viewing softcards	164
9.4.5. Verifying the pass phrase of a card or softcard	166
9.5. Changing card and softcard pass phrase	167
9.5.1. Changing known pass phrase	168
9.5.2. Changing unknown or lost pass phrase	171
9.6. Replacing Operator Card Sets	172
9.6.1. Replacing OCSs from the unit front panel	174
9.6.2. Replacing OCSs with KeySafe	174
9.6.3. Replacing OCSs or softcards with rocs	176
9.7. Replacing the Administrator Card Set	184
9.7.1. Replacing an Administrator Card Set using the Connect front panel	185
9.7.2. Replacing an ACS with KeySafe	186
9.7.3. Replacing an Administrator Card Set with racs	188
10. Application interfaces	189
10.1. nCipherKM JCA/JCE CSP	189
10.1.1. Installing the nCipherKM JCA/JCE CSP	190
10.1.2. Named Modules in Java 11	195
10.1.3. keytool	195
10.1.4. Using keys	196
10.1.5. System properties	197
10.1.6. Compatibility	199
10.2. nShield PKCS #11 library	200
10.2.1. Choosing functions	201
10.2.2. PKCS #11 library with Security Assurance Mechanism	203
10.2.3. Using the nShield PKCS #11 library	204
10.2.4. nShield PKCS #11 library environment variables	207
10.2.5. Checking the installation of the nShield PKCS #11 library	219
10.2.6. How the nShield PKCS #11 library protects keys	221
10.3. nShield native and custom applications	222
10.4. CodeSafe applications	222
10.5. Remotely loading and updating SEE machines	223

11. Remote Operator	226
11.1. About Remote Operator	226
11.2. Configuring Remote Operator	226
11.2.1. Overview of configuring Remote Operator	227
11.2.2. Configuring HSMs for Remote Operator	227
11.2.3. Configuring slot import and export	228
11.2.4. Using Remote Operator with applications requiring cards in slot 0	232
11.2.5. Using Remote Operator on Remapped Slots	233
11.2.6. Configuration Example for Using Remote Administration and Remote Operator Concurrently	234
11.2.7. Using Remote Operator with Remote Administration with Older Versions of the Software	235
11.3. Creating OCSs and keys for Remote Operator	235
11.3.1. Creating OCSs for use with Remote Operator	236
11.3.2. Loading Remote Operator Card Sets	236
11.3.3. Generating keys for use with Remote Operator	237
11.3.4. Configuring the application	237
11.3.5. Managing Remote Operator slots using the unit front panel	238
12. Working with keys	239
12.1. Common Criteria CMTS Mode Assigned Keys	239
12.2. Generating keys	240
12.2.1. Generating keys using the command line	240
12.2.2. Generating keys with KeySafe	242
12.2.3. Generating NVRAM-stored keys	243
12.3. Importing keys	244
12.3.1. Importing keys from the command line	244
12.3.2. Importing keys with KeySafe	245
12.4. Listing supported applications with generatekey	246
12.5. Retargeting keys with generatekey	246
12.6. Viewing keys	247
12.6.1. Viewing information about keys on the unit front panel	247
12.6.2. Viewing keys with KeySafe	248
12.6.3. Viewing keys using the command line	248
12.7. Verifying Key Generation Certificates with nfkmverify	250
12.7.1. Usage	251
12.8. Discarding keys	252
12.9. Restoring keys	252
13. Using KeySafe	253
13.1. Setting up KeySafe	253

13.2. Starting KeySafe	254
13.3. About the KeySafe window	254
13.3.1. Sidebar	254
13.3.2. Menu buttons	254
13.3.3. Menus	255
13.3.4. Module Status tree	256
13.3.5. Main panel area	259
13.4. Errors	260
14. Supplied utilities	261
14.1. Utilities for general operations	261
14.2. Hardware utilities	264
14.3. Test analysis tools	264
14.4. Security World utilities	264
14.5. CodeSafe utilities	267
14.6. PKCS #11	268
14.7. nShield Connect utilities	269
14.8. Developer-specific utilities	271
14.9. Utilities that require a privileged connection	272
15. Preload Utility	273
15.1. Overview	273
15.2. Using Preload	273
15.2.1. Preload Commands	274
15.2.2. Preload file location	274
15.2.3. Preload Command Line Arguments	274
15.2.4. Pattern Matching	276
15.3. Preload File	276
15.4. Softcard Support	277
15.4.1. No Cardset Keys	278
15.5. FIPS Auth	278
15.6. Admin Keys	278
15.6.1. Listing	278
15.6.2. Loading	278
15.7. High Availability	279
15.7.1. Prerequisites for high availability mode	280
15.7.2. Differences from legacy behaviour	280
15.7.3. Conditions for Management/Reloading	280
15.7.4. Merged Keys in the Preload File	280
15.7.5. Polling Interval	281
15.7.6. Key timeouts and use limits	281

15.7.7. Multiple Preload instances in high availability mode	282
15.7.8. FIPS Auth in High Availability mode	284
15.7.9. PKCS #11 and JCE	284
15.7.10. Unsupported options	285
15.8. Logging	285
15.9. Using preloaded objects - Worked example	286
16. Environment variables	288
17. Logging, debugging, and diagnostics	291
17.1. Logging and debugging	291
17.1.1. Environment variables to control logging	291
17.1.2. Logging and debugging information for PKCS #11	295
17.1.3. Debugging information for Java	296
17.2. Diagnostics and system information	297
17.2.1. nfdiag: diagnostics utility	298
17.2.2. nfkminfo: information utility	300
17.2.3. perfcheck: performance measurement checking tool	310
17.2.4. stattree: information utility	314
17.3. How data is affected when a module loses power and restarts	320
18. nShield Connect and client configuration files	321
18.1. Location of client configuration files	321
18.2. Location of module configuration files	321
18.3. Structure of configuration files	322
18.4. Sections only in module configuration files	324
18.4.1. nethsm_settings	324
18.4.2. nethsm_eth	324
18.4.3. nethsm_eth_ipv6	325
18.4.4. nethsm_gateway	325
18.4.5. nethsm_gateway_ipv6	325
18.4.6. nethsm_bond	325
18.4.7. nethsm_route	327
18.4.8. nethsm_route_ipv6	327
18.4.9. nethsm_eth1_enable	328
18.4.10. nethsm_bond_enable	328
18.4.11. nethsm_enable	328
18.4.12. cosmod	328
18.4.13. hs_clients	329
18.4.14. rfs_client	329
18.4.15. sys_log	330
18.4.16. remote_sys_log	330

18.4.17. config_op	330
18.4.18. ui_lockout	331
18.5. Sections in both module and client configuration files	332
18.5.1. server_settings	332
18.5.2. server_remotecomms	335
18.5.3. server_remotecomms_ipv6	336
18.5.4. server_startup	337
18.5.5. load_seemachine	338
18.5.6. slot_imports	339
18.5.7. slot_exports	339
18.5.8. dynamic_slots	340
18.5.9. slot_mapping	340
18.5.10. dynamic_slot_timeouts	340
18.6. Sections only in client configuration files	341
18.6.1. module_settings	341
18.6.2. server_performance	341
18.6.3. nethsm_imports	342
18.6.4. rfs_sync_client	342
18.6.5. remote_file_system	343
18.6.6. remote_administration_slot_server_startup	344
19. Cryptographic algorithms	345
19.1. Symmetric algorithms	345
19.2. Asymmetric algorithms	345
19.3. FIPS information	346
19.4. Compatibility with v1 and v2 Security Worlds	347
20. Audit Logging	348
20.1. Architecture	348
20.2. Audit Logging Implementation	349
20.2.1. Audit log Entry	349
20.2.2. Signature Block	349
20.2.3. Certifier Block	350
20.2.4. Audit Log Verification process	350
20.3. Configuring Audit Logging	351
20.3.1. Confirming Audit Logging configuration	352
20.3.2. Disabling Audit Logging	354
20.3.3. Configuring syslog	354
20.4. Log distribution	355
20.5. Configuring log distribution	356
20.5.1. Configuring syslog message infrastructure	357

20.6. Log format	357
20.6.1. CEF format	357
20.6.2. CEF extensions	359
20.6.3. Infrastructure extensions	361
20.6.4. Message and reboot counters	361
20.6.5. Certifier Block extensions	361
20.6.6. Signature Block extensions	362
20.6.7. Example Audit Logging messages	363
20.7. Commands Audited	365
20.7.1. Key usage logging	365
20.7.2. Commands generating Audit Log messages	366
20.7.3. Key commands	366
20.7.4. Logical Token and Share Commands	368
20.7.5. Administrative Commands	369
20.7.6. Dynamic Slot Commands	370
20.7.7. Heartbeat	370
20.7.8. Post Reboot Logging	371
20.7.9. Tracing Key Usage	373
20.8. Audit Log Verification	374
20.8.1. Running the example verification program	374
20.8.2. Program Architecture	379
20.8.3. Extended Verification	380
21. Key generation options and parameters	381
21.1. Key application type (APPNAME)	381
21.2. Key properties (NAME=VALUE)	382
21.3. Available key properties by action/application	386
22. Checking and changing the mode on an nShield Connect	390
22.1. nShield Connect front panel controls	390
22.2. Available modes	390
22.3. Identifying the current mode	390
22.3.1. Checking the mode at the nShield Connect	390
22.3.2. Checking the mode using enquiry	391
22.3.3. Checking the mode by using KeySafe	392
22.4. Changing the mode	393
22.4.1. Changing the mode using the front panel controls of an nShield Connect	393
22.4.2. Changing the mode using remote mode and nopclearfail	393
23. Upgrading the nShield Connect image file and associated firmware	395
23.1. Version Security Number (VSN)	395

23.2. Key data	395
23.3. Upgrading the nShield Connect image file and firmware using the front panel ..	396
23.4. Remotely enabling dynamic feature certificates including nShield Connect client licenses	397
23.5. Upgrading the nShield Connect image file and firmware from a privileged client	398
23.5.1. Enabling and disabling remote upgrade	400
23.6. After firmware installation	400
24. SNMP monitoring agent	401
24.1. Installing the SNMP agent	402
24.1.1. Default installation settings	402
24.1.2. Do you already have an SNMP agent running?	402
24.1.3. Starting the SNMP agent	402
24.2. Basic configuration	403
24.2.1. Protecting the SNMP installation	403
24.2.2. Configuring the SNMP agent	403
24.2.3. The SNMP agent persistent configuration file	405
24.2.4. Agent Behaviour	406
24.2.5. agentaddress directive	406
24.2.6. agentgroup and agentuser directives	406
24.2.7. System information	407
24.3. USM users	407
24.4. Traditional access control	409
24.5. VACM configuration	411
24.6. Trap Configuration	414
24.6.1. SNMPv1 and SNMPv2 traps	415
24.6.2. SNMPv3 traps	416
24.7. Using the SNMP agent with a manager application	416
24.7.1. Manager configuration	416
24.7.2. MIB module overview	417
24.7.3. MIB functionality	417
24.7.4. Memory usage monitoring	419
24.7.5. Administration sub-tree overview	420
24.7.6. Statistics sub-tree overview	429
24.8. SNMP agent command-line	435
24.8.1. SNMP agent (snmpd) switches	435
24.8.2. Using the SNMP command-line utilities	436
25. Morse code error messages	438
25.1. Reading Morse code	438

25.2. Runtime library errors	439
25.3. Hardware driver errors	439
25.4. Maintenance mode errors	442
25.5. Operational mode errors	443
26. Application Performance Tuning	445
26.1. Job Count	445
26.2. Client Configuration	445
26.3. Highly Multi-threaded Client Applications	445
26.4. File Descriptor Limits	446
27. Merged Keys Concept	447
28. Product returns	449

1. Introduction

1.1. Read this guide if ...

Read this guide if you need to configure or manage:

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

The Connect is connected to a network by an Ethernet connection. Each HSM is configured to communicate with one or more client computers on the network. You can also configure clients to make use of any other network-connected HSMs on the network, as well as locally connected HSMs.

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

This guide assumes that:

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



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

1.2. Terminology

The Connect is referred to as the *Connect*, the *hardware security module*, or the *HSM*.

1.3. Model numbers

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

Model number	Used for
NH2047	Connect 6000

Model number	Used for
NH2040	Connect 1500
NH2033	Connect 500
NH2068	Connect 6000+
NH2061	Connect 1500+
NH2054	Connect 500+
NH2075-B	Connect XC Base
NH2075-M	Connect XC Medium
NH2075-H	Connect XC High
NH2082	Connect XC SCAP
NH2089-B	Connect XC Base - Serial Console
NH2089-M	Connect XC Mid - Serial Console
NH2089-H	Connect XC High - Serial Console
NH3003-B	Connect CLX Base - Serial Console
NH3003-M	Connect CLX Mid - Serial Console
NH3003-H	Connect CLX High - Serial Console

1.4. Security World Software

The Security World for nShield is a collection of programs and utilities, including the hard-server, supplied by Entrust to install and maintain your nShield security system.

Entrust provides the firmware that runs on the nShield Connect, and software to run on each client computer. The Connect is supplied with the latest version of the HSM firmware installed. For more information about:

- [Upgrading the nShield Connect image file and associated firmware](#)
- Installing and configuring the software on each client computer, see the Installation Guide and [Client Software and module configuration](#).
- The supplied utilities, see [Supplied utilities](#).

1.4.1. Software architecture

The software, firmware and utilities have version numbers and there is also a version num-

ber for the World which refers to the World data that is stored in encrypted form on the client computer, typically in the `NFAST_KMDATA` directory or on the RFS. This data includes information concerning the World itself and also concerning each key that was created within that World. The World version created is determined by the version numbers of the software and firmware used when it was first created, see [Creating and managing a Security World](#).

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

1.4.1.1. Hardserver

The *hardserver* software controls communication between the internal security module and applications on the network.

Separate instances of the hardserver run on the unit and each client that is configured to work with the unit. There is a secure channel, known as the *impath*, between the two software instances, which forms a single secure entity for transferring data between the unit and the clients. See also [Compatibility issues](#).

The unit's hardserver is configured using the front panel on the unit, or by means of uploaded configuration data. Configuration data is stored on the unit and in files in a specially configured file system on each client computer. For more information about using:

- The front panel to configure the unit, see [The nShield Connect user interface](#)
- The specially configured file system to configure the unit and the client, see [Client Software and module configuration](#).

1.4.1.2. Remote file system (RFS)

Each unit uses a *remote file system* (RFS). You can configure the RFS on any computer, but it is normally located on the first client that is configured. The RFS contains:

- The master configuration information for the unit
- The Security World files
- The key data.

Do not copy the master configuration to file systems on other clients. You can copy Security World files and key data to other clients to allow you to manage the unit from more than one client. To make it available to the unit, copy to the RFS the data for Security Worlds, cards or keys that you create on a client that does not contain the RFS.

1.4.1.3. Security World Software

The Security World Software is a collection of programs and utilities that you install on the client and use to maintain the nShield security system. The Security World Software includes the following:

- The appropriate installer for the client platform
- The client hardserver
- A set of utilities for configuring the Connect
- A set of utilities and the KeySafe application for performing key management tasks on nShield HSMs.

The Connect is referenced and used by a utility or application in the same way as a local (PCI-connected) nShield HSM.

1.4.2. Default directories

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

Directory name	Default path
nShield Installation	<code>/opt/nfast/</code>
Key Management Data	<code>/opt/nfast/kmdata/</code>
Dynamic Feature Certificates	<code>/opt/nfast/femcerts/</code>
Static Feature Certificates	<code>/opt/nfast/kmdata/hsm-ESN/features</code>
Log Files	<code>/opt/nfast/log</code>
User Log Files	<code>/home/<user>/nshieldlogs</code>



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

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

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

1.4.3. Utility help options

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

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

1.5. Further information

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

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

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



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

1.6. Security advisories

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



The Remote Administration feature supports remote firmware upgrade

of nShield Solo, nShield Solo XC, nShield Connect and remote ACS card presentation.

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

1.7. Recycling and disposal information

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

2. Security Worlds

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

A Security World provides you with the following features:

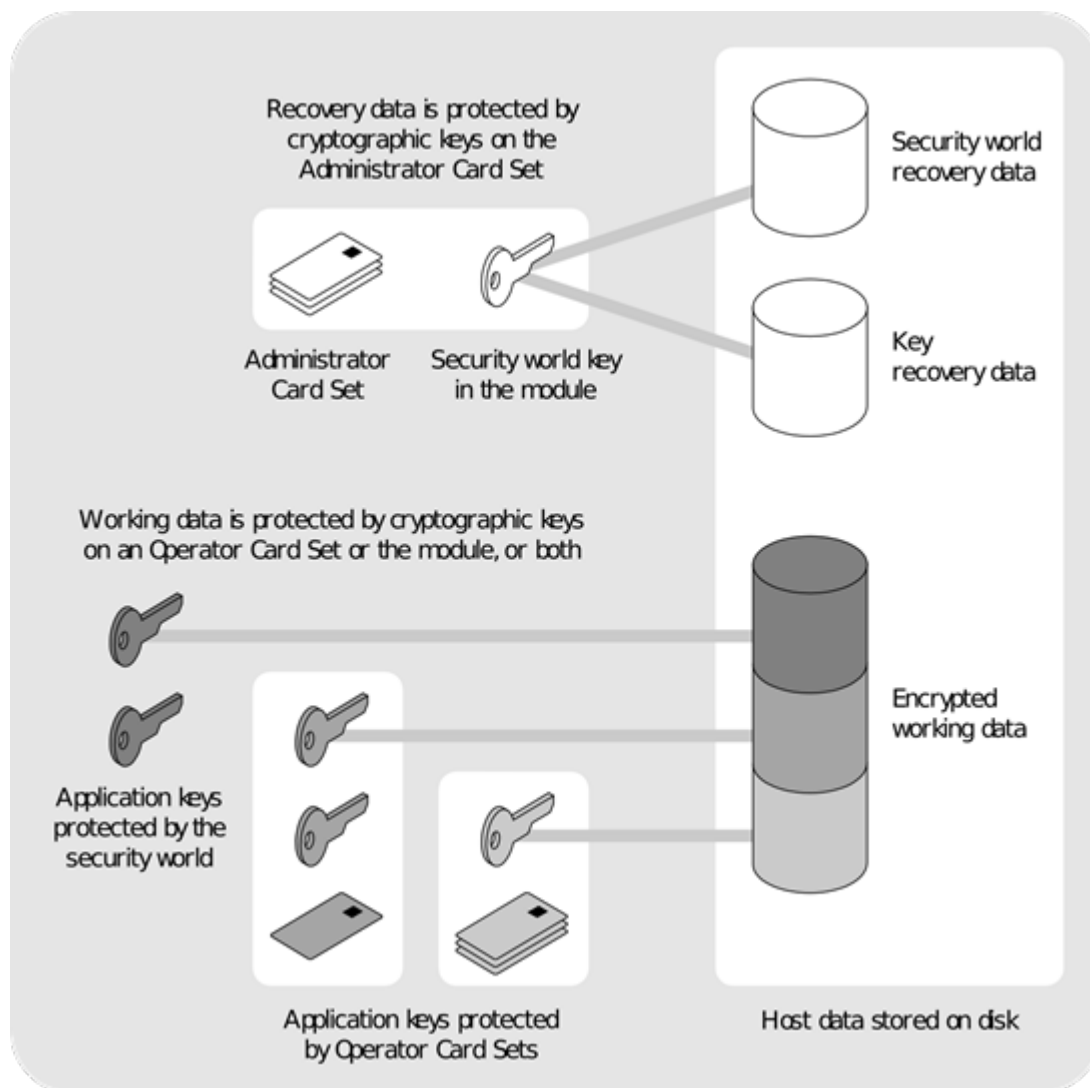
- Security
- Application independence
- Platform independence
- Flexibility
- Scalability
- Robustness
- Audit logging

A Security World comprises:

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

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

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



2.1. Security

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

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



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

2.1.1. Smart cards

The Security World uses:

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



In FIPS 140-2 Level 3 Security Worlds, you require a card from either the ACS or an OCS to authorize most operations, including the creation of keys and OCSs.

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



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

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

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

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

2.1.2. Remote Operator

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



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

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

- The smart card is from a persistent OCS
See [Using persistent Operator Card Sets](#) for more about persistent cards.

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

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

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

See also [Compatibility issues](#).

2.1.3. Remote Administration

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

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

For more information see [Security World Remote Administration](#)

2.1.4. Security World and an Connect

To enable the secure transmission of data between an Connect and each client, the hardserver uses an *impath* (see [Remote Operator](#)). Any Connect on the network can load a Security World securely over the network, and access its keys, wherever the HSM is installed.

Security World and key data is stored on the file system of the Connect, where it is updated whenever card or key operations are performed on the HSM. The data is also automatically transferred to the *remote file system* (RFS). If required, you can also share the data with client computers that use the Security World. For more information, see:

- [Remote file system \(RFS\)](#)
- [Configuring the remote file system \(RFS\)](#).

You can update the Security World on the host using:

- The Connect front panel controls
- The command-line utilities
- The Cryptographic Service Provider wizard
- KeySafe.

You can also use these tools to create keys or cards. If you perform such tasks on a client other than the computer on which the RFS is installed, you must transfer the updated files to the RFS before they are available to the HSM.

2.1.5. NIST SP800-131A

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

2.1.5.1. Compatibility issues

In order to comply with the latest encryption standards, Entrust has adopted an enhanced NIST SP800-131A compliant encryption protocol between Connect HSMs and their clients with Security World software installed. In some cases, this change may have an impact on the compatibility of network-attached HSMs in environments with mixed HSM deployments.

In most cases where versions of Security World software of v11.50 or later are deployed in conjunction with v11.40 software or lower, no action is required. However, there are two cases in which communication cannot be established between the HSM and clients or hosts:

- v11.50 or higher clients communicating with a v11.40 or lower Connect, where the HSM client uses an nToken.
- v11.50 or higher Connect communicating with a Remote File System (RFS) using v11.40 or lower.

Release version	Image versions Connect	Security World Software ¹ v11.40	Security World Software ¹ v11.50 and later
Up to 11.40	Up to image version 0.3.5.	Supported.	For deployments with nTokens, please upgrade the Connect netimage. As a less preferred option, you can downgrade the client-side software.
11.50 or later	Image 0.3.6 or later.	RFS and client software upgrade required.	Supported.

¹ Previously known as nCipher software, or nCSS.

2.1.6. FIPS 140-2 compliance

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

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

2.1.6.1. FIPS 140-2 Level 3 compliance

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

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

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

If you choose to create a Security World that complies with FIPS 140-2 Level 3, the nShield HSM initializes in that mode, conforming with the roles and services, key management, and self-test sections of the FIPS validation certificate.

Before you can create or erase an OCS in a Security World that complies with FIPS 140-2 Level 3, you must authorize the action with a card from the ACS or an OCS from that Security World.

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

2.1.7. Common Criteria compliance

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

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

2.2. Platform independence

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

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



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

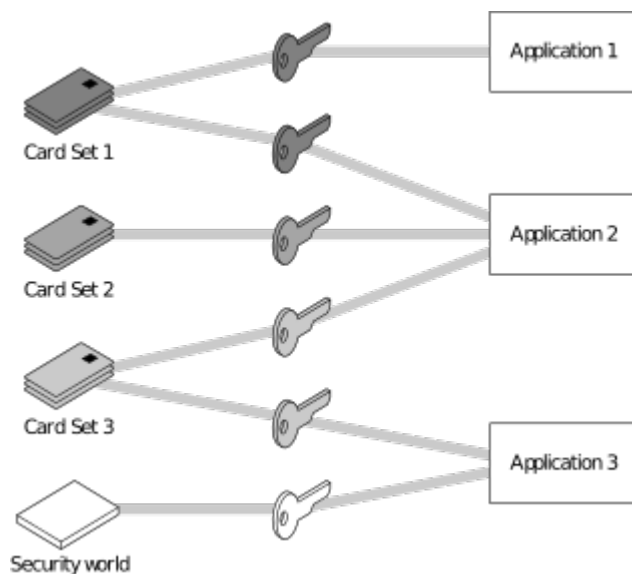
2.3. Application independence

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

You do not need to specify:

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

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



In the image above:

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

2.4. Flexibility

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

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

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

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

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

This level of protection is suitable for high-availability Web servers that you want to recover

immediately if the computer resets.

2.4.2. Using Operator Card Sets: OCS-protected keys

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

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

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

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



Some applications restrict K to 1.

2.4.2.1. Using Operator Card Sets to share keys securely

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

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

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

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

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

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

- K to 1

- N equal to the number of users.

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



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

If a card becomes damaged, you can replace the whole OCS if you have authorization from the ACS belonging to that Security World.



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

2.4.2.2. Using Operator Card Sets for high availability

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

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



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

2.4.2.3. Using persistent Operator Card Sets

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

soon as the card is removed from the card reader, which provides added security.

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

This enables:

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

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

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

- When the application that loaded the OCS closes the connection to the hardware security module
- After a time limit that is specified when the card set is created
- When an application chooses to remove a key
- When the HSM is cleared. See [Manually removing keys from an HSM](#) for more information
- If there is a power loss to the module, for example, due to power outage.



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

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

2.4.2.4. Manually removing keys from an HSM

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

- Run the command `nopclearfail --clear --all`
- Press the Clear button of the hardware security module

- Turn off power to the hardware security module

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

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

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

2.4.3. Using pass phrases for extra security

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

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



Some applications do not support the use of pass phrases.

2.4.3.1. Maximum pass phrase length



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

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

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

Other commands are unaffected.

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

Prior to Security World Software v11.72, we set no absolute limit on the length of pass phrases, although individual applications may not accept pass phrases longer than a spe-

cific number of characters. Likewise, the Security World does not impose restrictions on which characters you can use in a pass phrase, although some applications may not accept certain characters.

Entrust recommends that your password only contains 7-bit ASCII characters:

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

2.4.3.2. Pass phrase penalty timer

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

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



A HSM only has a small number of command processing threads, related to the kind of hardware in use (for example, 9 threads on an nShield Solo). Once all of these are waiting for a penalty to expire, any other submitted commands will be forced to wait. This can mean that even if penalty time isn't large, the total delay experienced by clients may be substantial.

2.4.4. Using softcard-protected keys

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

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

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

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

The pass phrase of a softcard is set when you generate it, and you can use a single softcard

to protect multiple keys. Softcards function as persistent 1/1 logical tokens, and after a soft card is loaded, it remains valid for loading its keys until its **KeyID** is destroyed.

2.4.5. NVRAM key storage

Application keys protected by an Connect are stored in an encrypted format and copied automatically to the remote file system. A hardware security module, such as an Connect, and/or an OCS protects the keys, as described in the preceding sections. You can also store application keys within the nonvolatile memory of a suitable hardware security module.

NVRAM-stored keys are encrypted in exactly the same way as application keys that are protected by the unit. The encrypted application key on the unit is replaced by a file containing the name of the NVRAM file that contains the application key. This file allows applications to use NVRAM-stored keys in the same way as keys stored in the remote file system. You can protect an NVRAM-stored key with either the Security World or an OCS.



NVRAM-stored keys differ from standard application keys only in their storage location. They still require protection by the unit or an OCS.

Use of an NVRAM-stored key is the same as for any other key protected by an Connect Security World.

NVRAM key storage:

- Offers no additional security benefits above those offered by the standard Security World Software mechanisms
- Is available for only a limited number of keys
- Reduces a Security World's ability to offer load-balancing and recovery
- Requires backup and recovery procedures in addition to any that you implement for data stored on the client computer.



Do not store keys in NVRAM unless you must do so to satisfy regulatory requirements.

NVRAM key storage was introduced only for those users who must store keys within the physical boundary of a hardware security module (such as an Connect) to comply with regulatory requirements. NVRAM-stored keys provide no additional security benefits and their use exposes your ACS to increased risk. Storing keys in nonvolatile memory also reduces load-balancing and recovery capabilities. Because of these factors, we recommend you always use standard Security World keys unless explicitly required to use NVRAM-stored keys.

2.5. Scalability

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

To share a Security World across multiple clients:

- Ensure each client has at least one hardware security module configured
- Copy the Security World data to each client
- Load the Security World onto the hardware security modules for each client.

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

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

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



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

If you want to make cards or keys which are normally created from the client available from the front panel of the hardware security module, we recommend that you use client cooperation to automate the copying of files to the device.

2.5.1. Load-sharing

If you have more than one hardware security module configured with a client, your applications (that have been integrated with the Security World Software) can make use of the load-sharing features in the Security World Software to share the cryptography between

them. Two approaches are supported:

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



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

HSM Pool mode is supported on all major APIs except Java (i.e. nCipherKM JCA/JCE CSP). When HSM Pool mode is enabled for an API, the application sees the HSMs in the Security World as a single resource pool. A significant benefit is that when a failed HSM is restored to the Security World or a new HSM is added to the Security World, it is automatically added to the resource pool making it available for cryptographic operations without restarting the application (i.e. failback support). The pool of HSMs can be viewed as a single resource using the command `enquiry --pool`.



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

2.6. Robustness

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

2.6.1. Backup and recovery

The Security World data stored on the file system and remote file system of the hardware security module is encrypted using the Security World key.

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

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

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

these cards in a secure environment.



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

2.6.2. Replacing a hardware security module

If you have a problem with a hardware security module, you can replace it with another hardware security module of the same type by loading the Security World data on the remote file system onto the replacement device. Alternatively, you may be able to erase the Security World from the device that has the problem, return the device to its default state and then reload the Security World on the same device.

If you have more than one hardware security module configured with a client and you use one of the load-sharing modes identified above, then your client application is resilient to the failure of individual hardware security modules. If you use HSM Pool mode, then an Connect can be replaced and returned to the HSM pool without restarting the client application.

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

2.6.3. Replacing the Administrator Card Set

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

- The front panel controls of the Connect
- The KeySafe **Replace Administrator Card Set** option
- **racs** utility (see [Replacing the Administrator Card Set](#)).



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

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

The loss or failure of one of the smart cards in the ACS means that you must replace the

ACS. However, you cannot replace the ACS unless you have:

- The required number of current cards
- Access to their pass phrases.



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

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

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



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



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

To create new copies of the keys protected by the recovery key on an OCS, you can use either:

- The front panel controls of the Connect
- The `rocs` command-line utility.



It is not possible to recover PKCS #11 keys using the front panel controls of the Connect. You must use the `rocs` command-line utility.

2.6.4.1. The security of recovery and replacement data

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

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

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

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

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

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

2.7. Audit Logging

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

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

For further information, see [Audit Logging](#).

2.8. KeySafe and Security Worlds

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



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

When you use KeySafe to create cards or keys, the data is written to the Key Management Data directory on the computer on which you run KeySafe. An Connect can only use this data when it is transferred to the remote file system (if it is on a different computer), from where it is loaded automatically onto the unit. For this reason, you may find it most convenient to run KeySafe on the same computer as the remote file system.

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

KeySafe enables you to:

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

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

In addition to KeySafe, we also supply command-line utilities to manage the Security

World; for more information about the supplied utilities, see [Supplied utilities](#). Current versions of these tools can be used interchangeably with the current version of KeySafe.

You can also perform all the tasks available from KeySafe using the front panel controls of the unit, except for adding, importing and deleting keys.

2.9. Applications and Security Worlds

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

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

For more information about the Entrust range of Integration Guides:

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

2.10. The nShield PKCS #11 library and Security Worlds

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

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

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

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

2.11. Risks

Even the best-designed tools cannot offer security against every risk. Although a Security

World can control which user has access to which keys, it cannot prevent a user from using a key fraudulently. For example, although a Security World can determine if a user is authorized to use a particular key, it cannot determine whether the message that is sent with that key is accurate.

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

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

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



nShield Remote Administration Cards can only be supplied by Entrust.

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



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

3. The nShield Connect user interface

This chapter describes the Connect user interface, including the front panel controls. You are also shown how to use a keyboard to control the unit.

3.1. Front panel controls

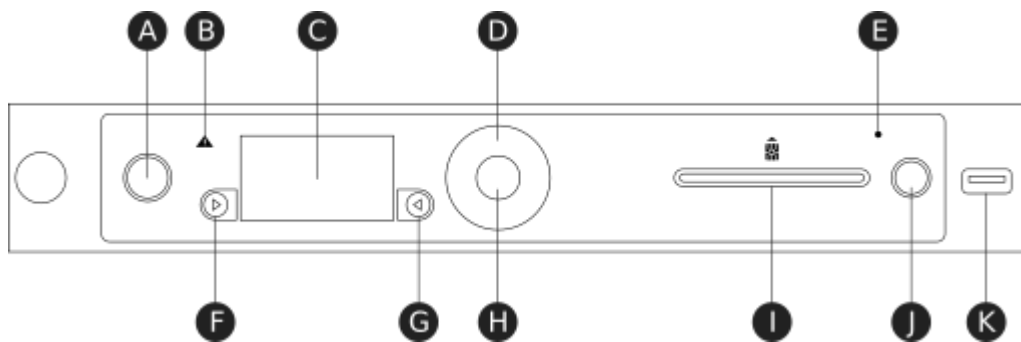
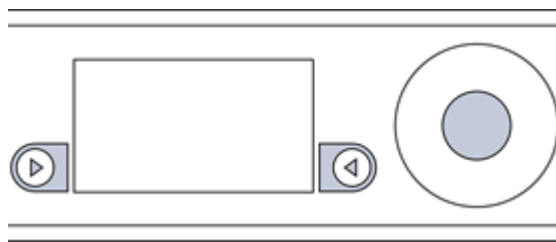


Figure 1. Connect front panel controls

Label	Description
A	Power button
B	Warning LED (orange)
C	Display screen
D	Touch wheel
E	Status indicator LED (blue)
F	Display navigation button (left)
G	Display navigation button (right)
H	Select button
I	Slot for smart cards
J	Clear button
K	USB connector

3.2. Display screen and controls

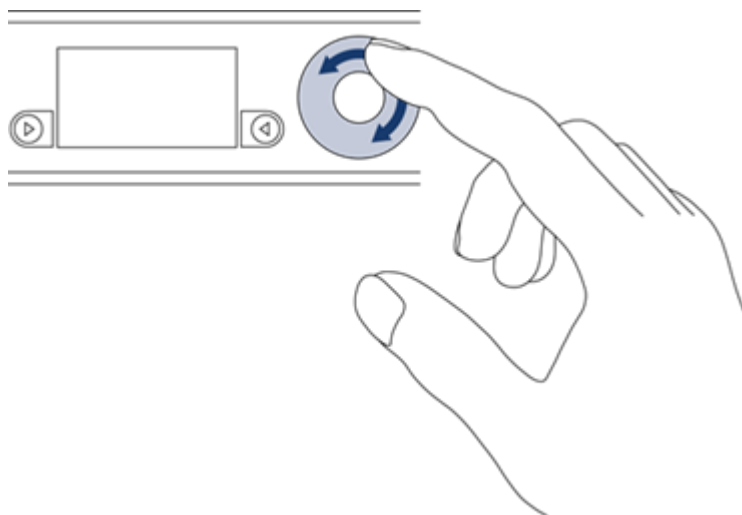


When the unit is powered, the display screen displays a menu or a dialog.

Each menu or dialog includes onscreen navigation labels that appear at the bottom of the display screen, on either side next to the display navigation buttons. Press the button next to the label to perform the action specified by the label.

To go back to the previous dialog or menu screen, use the navigation button to the left of the screen. To confirm a dialog value or select a menu option, either:

- Press the navigation button to the right of the screen.
- Touch the Select button.



Use the touch wheel to changes values or move the cursor on the display screen. To confirm a value, touch the Select button.

3.2.1. Menu screens

You can access menus from the display screen.

Menus are displayed as a list of selectable options. An onscreen arrow points to the currently selectable option. If the menu has more than four options, an arrow indicates the direction in which more options are available.

To select a menu option:

1. Move the indicator arrow up or down with the touch wheel.
2. When the indicator arrow points to the option you want to select, either:
 - Press the navigation button to the right of the screen (labeled onscreen as **SELECT**).
 - Touch the Select button.

At the top right of the display screen, a number sequence indicates the path to the current option. The last digit of the sequence shows the location of the menu you are currently viewing. The top level menu has no numbers, but when you select the System menu, the number **1** is shown.

The preceding digits in the sequence show the position of each option in turn that was selected in previous menu screens to reach the current menu. For example, the sequence **1-2** shows that the indicator is on the second option of the menu that was reached by selecting the first option on the top-level menu.

For a map of the menu screens, see the *Installation Guide*.

3.2.2. Dialogs

For some tasks, a dialog is displayed onscreen. When the dialog opens, the cursor is in the first field. To change and then enter values:

1. Use the touch wheel to change the displayed value of the fields.
2. Touch the Select button to enter the displayed value and move to the next field in the dialog.

Repeat the procedure to enter all necessary values in the dialog.

3.2.3. Information display

When you use a dialog to request information (for example, a log or details of a key), there is often too much information to display onscreen. In such cases, only the first part of the information is displayed.

To view the rest of the information:

- Use the touch wheel to scroll the displayed information in the direction indicated by the onscreen arrows.
- When an **Options** label is displayed, press the right-hand navigation button to see a menu of navigation options. You can normally choose to go to the top, to the bottom, or to a specified line in the display.

The numbers of the lines currently being displayed onscreen are shown at the left of the screen. They are followed in parentheses (()) by the total number of lines available for display.

3.3. Using the front panel controls

You can use the front panel controls to configure the unit and to perform other tasks described in this guide. When the unit is working over the network with another computer (a client computer), you can program and control the unit as if it were part of the client computer.



If the unit is powered down while you are logged in, you are logged out automatically.

3.3.1. Start-up information

When you turn on power to the unit and it has completed its initialization, the lower part of the display screen shows basic start-up information about the unit.

There is a series of start-up information topics available. By default, the first displayed topic is the current **System time**. Use the touch wheel to view the other start-up information topics.

3.3.2. Administrative control of the unit

You can view and control the status of the unit by using the front panel controls and menu options.

Tasks	Action
Understand and control the power status of the unit	<p>Use the Power button to power up the unit.</p> <p>If the Power button is not illuminated, the unit is not powered. The Power button flashes intermittently as the unit powers up. It also flashes when the unit is in standby mode. For more information about the Power button, see the <i>Installation Guide</i>.</p>

Tasks	Action
Control access to the unit	<p>You can control access to the menus on the unit and the Power button on the front panel by using System > System configuration > Login settings.</p> <p>When UI Lockout with OCS has been enabled, you must log in with an authorized Operator Card before you can access the menus. You can still view information about the unit on the start-up screen. When you are logged in, you can log out and leave the unit locked.</p> <p>When UI Lockout without OCS has been enabled, you cannot access the menus, but you can still view information about the Connect on the start-up screen. The only way to disable this setting (apart from returning the HSM to factory state) is to push an updated configuration file to the Connect. See About user privileges and <i>ui_lockout</i> for more information.</p> <p>Power button lockout can be enabled and disabled independently when UI Lockout allows access to the menus.</p>
Unlock the unit	<p>When UI Lockout with OCS has been enabled and you have logged out, the display screen displays the label Login next to the right-hand navigation button. Press the right-hand navigation button, then insert an Operator Card that has been authorized for login, and follow the onscreen instructions.</p>
Log out of the unit	<p>Select Logout.</p> <p>This option is not available if UI Lockout with OCS has not enabled.</p>
Put the unit in standby mode	Press the Power button or select System > Shutdown/Reboot > Shutdown .
Restore the unit to its original configuration	Select System > System configuration > Default config .
Restore the unit to its factory state	Select System > Factory state .
Clear the memory of the internal hardware security module	Use the Clear button or select HSM > HSM reset .
View information about the current state of the internal hardware security module	Select HSM > HSM information .
View information about the current state of the system	See the next section.
Set the Real-Time Clock on the unit	<p>Select Security World mgmt > Admin operations > Set secure RTC.</p> <p>This option requires Administrator Card authorization.</p>

Tasks	Action
Change the mode of the unit	<p>Select HSM > Set HSM mode.</p> <ul style="list-style-type: none"> • Select Operational mode to run the unit normally. • Select initialization mode to configure the unit with software utilities rather than the front panel.

3.3.3. Viewing the current status of the unit

To view information about the current state of the system, from the main menu select **System > System information**. Select an option to view the associated information as follows:

Option	Description
View system log	Displays the system log.
View hardserver log	Displays the module hardserver log.
Display tasks	Displays the tasks that the system is currently performing.
Component versions	Displays the version numbers of the various system software components.
View h/w diagnostics	<p>Displays the following environmental information about the module:</p> <ul style="list-style-type: none"> • The current temperature at the left and right sensors • The minimum and maximum previous temperature at each sensor • The voltage on each power rail • The speed of each fan.
View tamper log	Displays the tamper log.
View unit id	Displays the ID of the unit.

3.3.4. Viewing the mode of the unit

See [Identifying the current mode](#).

3.4. Using a keyboard to control the unit

You can connect a keyboard to the USB connector on the front panel. You can connect either a US or a UK keyboard. To configure the unit for your keyboard type, select **System > System configuration > Keyboard layout** and then choose the keyboard type you require.

When you have connected a keyboard and configured the unit for its use, you can enter numbers and characters directly into the display. You can also control the unit by using the

following keystrokes:

Keystroke	Use
F1	Same as pressing the left-hand navigation button on the front panel.
F2	Same as pressing the right-hand navigation button on the front panel.
F3	Same as touching the Select button.
Esc	Same as pressing the left-hand navigation button on the front panel.
Enter	Where the Select button is active, same as pressing Select: where Button B is active, same as pressing Button B.
Up arrow	Moves the indicator upwards in a menu.
Down arrow	Moves the indicator downwards in a menu.
Tab	Moves the cursor to the next field in a dialog.
Shift-Tab	Moves the cursor to the previous field in a dialog.
PgUp	Displays the previous screen.
PgDn	Displays the next screen.

4. Physical security of the nShield Connect

This chapter provides a brief overview of the physical security measures that have been implemented to protect your Connect. You are also shown how to:

- Check the physical security of your Connect
- Disable and re-enable tamper detection functionality on your Connect.

The tamper detection functionality on the Connect provides additional physical security, over and above that provided by the holographic security seal, and alerts you to tampering in an operational environment. There is a removable lid on top of the Connect, protected by the security seal and tamper switches. To prevent the insertion of objects into the Connect, baffles are placed behind vents.

To optimize their effectiveness, use the physical security measures implemented on the Connect in association with your security policies and procedures. For more information about creating and managing security policies, see the *Security Policy Guide* on the NIST CMVP website.



Currently, the FIPS 140-2 Level 3 boundary is at the internal module. Future software releases may move the FIPS boundary so that it includes the entire Connect chassis.



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

4.1. Tamper event

The Connect offers several layers of tamper protection. The outer boundary of the box is tamper-responsive. When tampered, the unit ceases to provide cryptographic functionality, alerts the operator of the event, and ultimately forces the operator to reset the unit to factory defaults. Movements/vibrations, or replacing the fan tray module or a PSU, does not activate the tamper detection functionality. Tamper-responsiveness can be disabled, if desired. If a tamper event does occur, you can use the Security World data stored on the RFS and the Administrator Card Set to recover the keys and cryptographic data.

4.1.1. Connect lid is closed

If the Connect is powered, a tamper event has occurred, and the lid is closed, the following message is displayed onscreen:

```
** TAMPER DETECTED **  
Consult User Guide.  
Check physical seals.  
Reset to factory state  
or disable tamper  
detection first?  
RESET          DISABLE
```

When you see this message, examine your unit for physical signs of tampering (see [Physical security checks](#)).

If you discover signs of tampering do *not* attempt to put the unit back into operation. The date and time of the tamper event are recorded in the log (see [Logging, debugging, and diagnostics](#)).



The tamper-responsiveness circuitry has a Real Time Clock that is synchronised to the system time of the Connect, however the times associated with events in the tamper log may still have slight offsets to times recorded in other log files.

If there are signs of tampering, and the tamper event occurred:

- During transit from Entrust, contact Support.
- After installation, refer to your security policies and procedures.

For more information about creating and managing security policies, see the *Security Policy Guide*.

If there are no signs of tampering, you can *either*:

- Reset the Connect to a factory state

Or:

- Disable the tamper detection functionality and then reset the Connect to a factory state.



Do not disable tamper detection unless asked to do so by Support

If you chose to disable the tamper protection circuitry when you reset the Connect to a factory state, a cautionary warning is displayed prior to the request for card authorization. For information about re-enabling disabled tamper detection functionality, see [Disabling tamper detection functionality](#).

4.1.2. Connect lid is open

If the Connect is powered, a tamper event has occurred, and the lid is open, the following message is displayed onscreen:

```
** TAMPER DETECTED **  
  
Unit lid is open  
Replace lid or disable  
tamper detection.  
      DISABLE
```

An open lid indicates that the physical security of the unit is compromised. You may want to examine your unit for other physical signs of tampering (see [Physical security checks](#)). Do *not* attempt to put the unit back into operation.

The date and time of the tamper event are recorded in the log files (see [Logging, debugging, and diagnostics](#)). If the tamper event occurred:

- During transit from Entrust, contact Support.
- After installation, refer to your security policies and procedures. For more information about creating and managing security policies, see the *Security Policy Guide* on the NIST CMVP website.

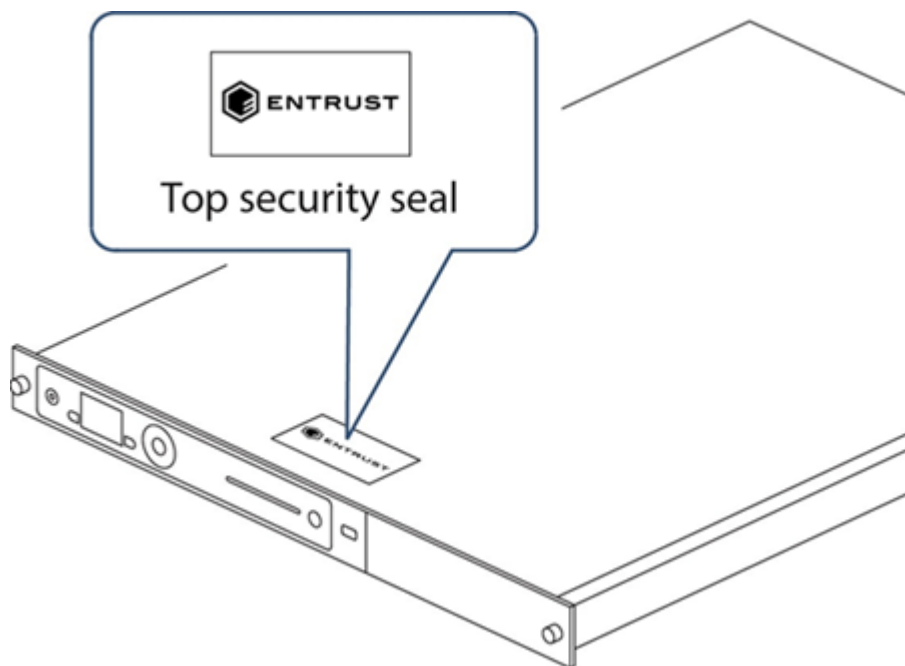
If you choose to replace the lid of the Connect, the onscreen message changes to the tamper alert message that is given when the lid is closed. Closing the lid provides you with the option to reset the unit to a factory state without disabling tamper detection functionality. If the lid remains open, all button presses other than those made using the right-hand navigation button are ignored. Pressing the right-hand navigation button disables tamper responsiveness and resets the Connect to a factory state.

4.2. Physical security checks

Check the physical security of your Connect before installation and at regular intervals afterwards. For an alternative presentation of the physical security checks described here, see the *Physical Security Checklist*. For more information about tamper events, and what actions to take if you discover signs of tampering, see [Tamper event](#).

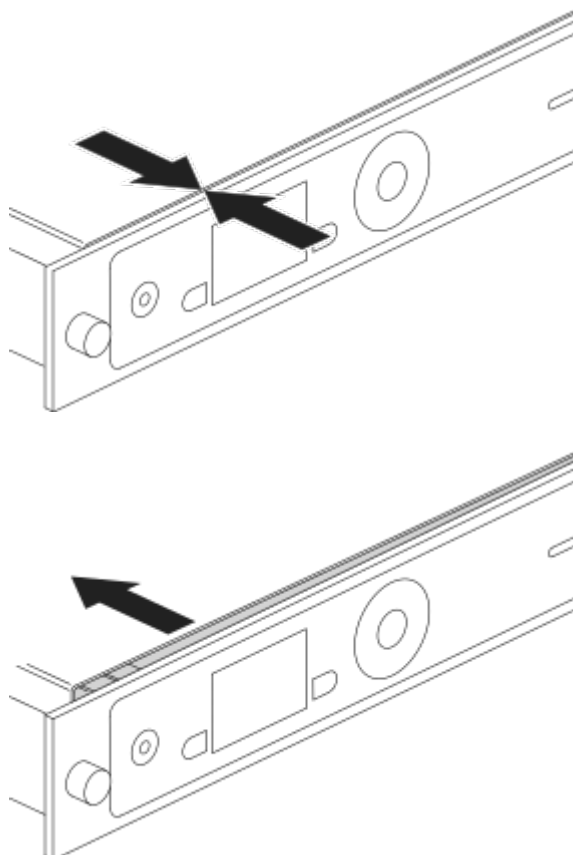
To determine if the security of the Connect is compromised:

1. Check that the physical security seal is authentic and intact. Look for the holographic foil bearing the nCipher logo. Look for cuts, tears and voiding of the seal. The seal is located on the top of the Connect chassis.



For information about the appearance of intact and damaged security seals, see the *Physical Security Checklist*.

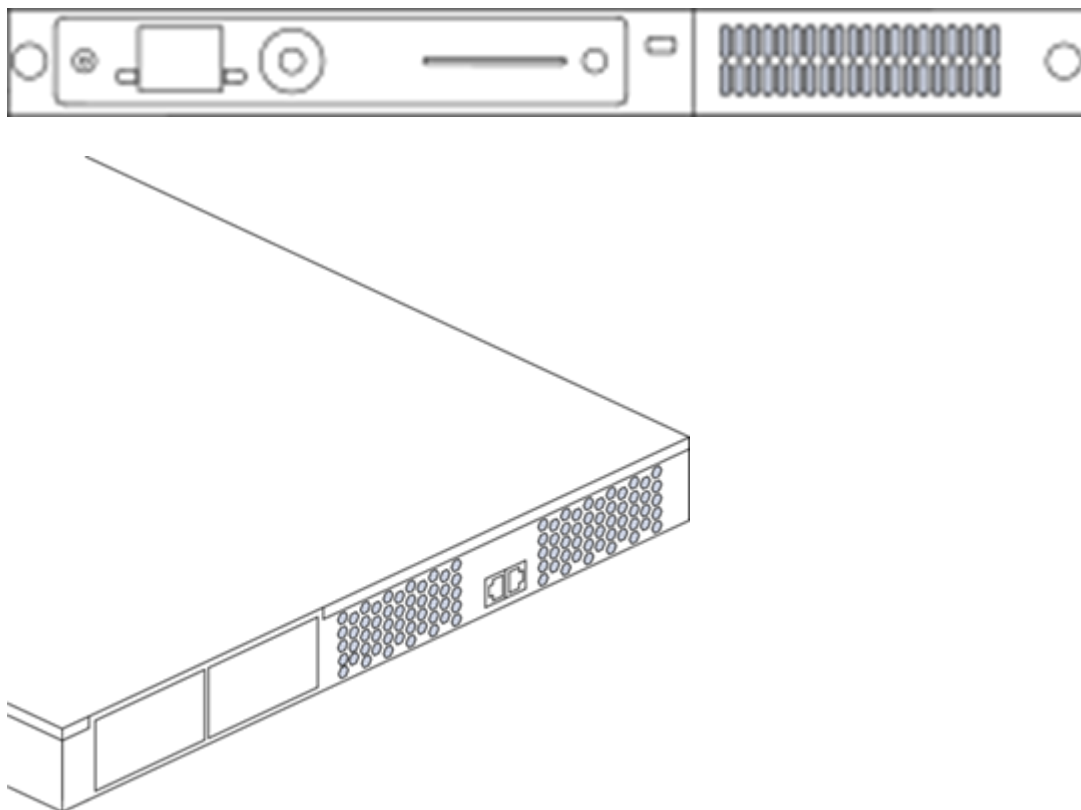
2. Check that the metal lid remains flush with the Connect chassis.



3. Check all surfaces — the top, bottom and sides of the Connect — for signs of physical

damage.

4. Check that there are no signs of physical damage to the vents, including attempts to insert objects into the vents.



4.3. Replacing the fan tray module and PSU

You can replace the fan tray module or a power supply unit (PSU) **without** activating a tamper event as both are outside the security boundary. You can access:

- The PSU(s) from the rear of the Connect.
- The fan tray module through the removable front vent.

Should a problem occur with the fan tray module or a PSU, contact Support **before** taking further action. For more information about replacing the fan tray module or a PSU, see the *Fan Tray Module Installation Sheet* or the *Power Supply Unit Installation Sheet*.



The fan tray module contains back-up batteries providing reserve capacity (a guaranteed minimum of 3 years) for tamper detection functionality even when the Connect is in an unpowered state.

The tamper protection circuitry remains fully operational if the Connect is placed on standby while a replacement operation is performed (whether you are replacing the fan tray module or one of the two PSUs, in the case of dual PSU units).



Provided that the Connect is connected to the mains power supply, the Connect displays an onscreen error message when back-up battery power is low. The Status LED also displays a low power warning. For more information, see the *Installation Guide*.

4.3.1. Replacing the fan tray module

It is not necessary to remove mains power to replace a fan tray module (we recommend that you power down the unit into standby state using the front panel power button). However, if mains power is removed then a replacement fan tray module **must be installed within an hour** to ensure that a tamper event is not activated. If put in standby state the time required to change fan tray module is unlimited. For more information about replacing the fan tray module, see the *Fan Tray Module Installation Sheet*.

4.3.1.1. Fan tray module error messages

If you receive any of the following error messages on the Connect display, accompanied by the orange warning LED, follow the related action in the table below:

Error message	Action
Single fan fail	Contact Support
Many fans fail	Replace fan tray
Battery power low	Consider replacing fan tray during the next scheduled service/maintenance period.
System Shutdown	Replace fan tray
Both fans in a pair had failed	

If the error message is **Single fan fail**, the Connect can continue operating under the specified operating environment. Although you are advised to contact Support, the limited nature of such a failure means you can replace the fan tray module at your convenience.

If the error message is **Many fans fail**, you must replace the fan tray module immediately.

If the error message is **Battery Power low**, this indicates that one or both of the backup batteries located on the fan tray module (required only when the Connect is removed from mains power) is running low.

The **Battery Power low** indication has no detrimental affect on the Connect performance whilst the unit remains powered. Entrust recommend customers should consider replacing the fan tray module during the next service/maintenance.

If two fans fail from a redundant pair, the Connect will display the error message **Many fans have failed** for a few seconds and it will then shutdown. On reboot, the Connect will then display the error messages **System Shutdown** and **Both fans in a pair had failed**. In this situation the fan tray module must be replaced immediately.

4.3.2. Replacing the PSU

If you have a dual PSU Connect, you do not have to remove power to the functioning PSU while replacing a faulty PSU. Tamper detection functionality will operate normally throughout the PSU replacement process. If you decide to remove power from both PSUs, tamper detection functionality will continue to operate normally for at least 3 years, as the fan tray module provides back-up capacity for this circuitry. For more information about replacing the PSU, see the *Power Supply Unit Installation Sheet*.

4.3.2.1. PSU error messages

If a PSU fails, an orange warning LED comes on and an error message is displayed on the Connect display. Although you are advised to contact Support, the unit can continue to operate normally and you can replace the failed PSU at your convenience. There is no need to power down the unit when you replace the failed PSU.

In addition to the orange warning LED, an audible warning is given when a PSU fails on an Connect. The audible warning is turned off when you navigate to the Critical errors screen.

4.3.3. Battery life when storing the Connect

If a Connect has been in storage for an extended period of time the fan tray module may need replacement.

Entrust guarantees a *minimum* battery life of three years, even if the Connect remains in storage and is not connected to the mains power supply during this time.

4.4. Disabling tamper detection functionality

We do *not* recommend disabling tamper detection functionality. However, there may be circumstances in which disabling this functionality is required. For example, you may want to:

- Bring the management of the Connect into line with your existing operational procedures
- Standardize the management task, if you are managing a mixture of older and newer

units.

To disable tamper responsiveness on your Connect:

1. From the main menu select **System > System configuration > Tamper config**.

You are shown the following dialog:

```
Tamper responsiveness
is currently enabled.

This matches nCipher's
recommendation.
Do you want to disable
tamper responsiveness?
```

2. Press the right-hand navigation button to disable tamper responsiveness.
3. You are asked to confirm the change. Press the right-hand navigation button again.

Tamper responsiveness is disabled and the unit is reset to a factory state. To restore the key data and reconnect the Connect to the network you must present a quorum of the ACS.

4.4.1. Reactivating disabled tamper functionality

If your circumstances change, for example, your operational procedures are updated, you may want to re-enable tamper responsiveness.

To re-enable tamper responsiveness on your Connect:

1. From the main menu select **System > System configuration > Tamper config**.
2. You are shown the following dialog:

```
Tamper responsiveness
is currently DISABLED.

nCipher recommends
that tamper should be
enabled. Re-enable
tamper responsiveness?
```

3. Press the right-hand navigation button to enable tamper responsiveness.
4. You are asked to confirm the change. Press the right-hand navigation button again.

Tamper responsiveness is enabled and the unit is reset to a factory state. To restore the key data and reconnect the Connect to the network you must present a quorum of the ACS. Tamper events are logged and never erased.

5. Software installation

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

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

5.1. After software installation



The *Installation Guide* provides brief explanations of how to perform all the post-installation tasks listed in this section. If this is the first time you are installing a unit and the Security World Software, or you are unfamiliar with the process, we recommend following the steps outlined in the *Installation Guide*.

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

1. Ensure that your public firewall is set up correctly. See the *Installation Guide* for your HSM for more information about firewall settings.
2. Perform the necessary basic HSM-client configuration tasks, as described in [Basic HSM and remote file system \(RFS\) configuration](#).
3. Create and configure a Security World, as described in [Creating a Security World](#).
4. Create an OCS, as described in [Creating Operator Card Sets \(OCSs\)](#).
5. Complete additional necessary HSM-client configuration tasks:
 - a. To configure the unit so that it works with the client machine, see [Configuring the nShield Connect to use the client](#).
 - b. To configure client computers so that they work with the unit, see [Configuring client computers to use the nShield Connect](#).



For this release, you must generate a new client configuration file to take advantage of new functionality. To generate a new client configuration file, back up your existing configuration file and run the command `cfg-mkdefault`. This generates a template for the configuration file into which you can copy the settings from your old configuration file.

- c. To enable the TCP sockets for Java applications (including KeySafe), run the com-

mand:

```
config-serverstartup -sp
```

For more information, see [Client configuration utilities](#).

When all additional HSM configuration tasks are completed, you can:

1. Stop and then restart the hardserver, as described in [Stopping and restarting the hard-server](#).
2. Test the installation and configuration. See the *Installation Guide* for your HSM for more information.

6. Client Software and module configuration

This chapter describes how to configure the internal security module of the Connect and the client to communicate with each other, after you have installed the HSM and the Security World Software.

For more information about installing the HSM, see the *Installation Guide*. If you are configuring an HSM and client for the first time, or you want to complete a basic installation quickly, see the *Installation Guide*.



The Connect provides significant performance improvements, and can be deployed successfully with existing nShield products. Customers wishing to take advantage of these performance improvements **must** update their client machines with the latest Security World Software.

6.1. About user privileges

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

There are three levels of user:

- Superuser
- nfast group user
- normal

Typically, normal users can carry out operations involving Security Worlds, cardsets and keys, but not create Security Worlds, keys and cardsets. nfast group users have enhanced access, enabling them to create Security Worlds, cardsets and keys. For example, encrypted copies of keys are held in **kmdata** (`/opt/nfast/kmdata`). Normal users only have read access to the files, whereas nfast group users have read and write access, enabling them to create and use keys. nfast group users can also change the mode of an HSM remotely.

Superuser access (e.g., root) is required for such tasks as software installation, starting and stopping the hardserver and SNMP

6.2. About client configuration



You can add more HSMs to a client and more clients to an HSM at any

time.

The HSM and a client communicate by means of the hardserver, which handles secure transactions between the HSM and applications that run on the client. You must configure:

- Each client hardserver to communicate with the HSM that the client needs to use
- The HSM to communicate with clients that are allowed to use the HSM.

Information about the current configuration of the HSM or a client is stored in configuration files that are stored in specified file systems on the clients and on the HSM. For more information about the contents of configuration files, see [nShield Connect and client configuration files](#).

For information about configuring the HSM by importing an edited configuration file, see [About user privileges](#).

6.2.1. Remote file system (RFS)

Each HSM must have a remote file system (RFS) configured. The RFS contains master copies of all the files that the HSM needs:

- The HSM configuration file
- Feature-enabling certificates
- The encrypted Security World and key data for Security Worlds created on the HSM.

The RFS normally resides on a client computer, but it can be located on any computer that is accessible on the network.

For more information about setting up the remote file system, see [Configuring the Remote File System \(RFS\)](#).

6.2.2. HSM configuration

The data that defines the configuration of the HSM hardserver is stored in a file on the HSM. This file is automatically:

- Updated when the HSM is configured from the front panel
- Exported to the remote file system (RFS) directory.

Each HSM has separate configuration files on the RFS, stored in the directories with names of the form `/opt/nfast/kmdata/hsm-ESN/config` where **ESN** represents the electronic serial number of the HSM from which the files were exported. These directories can contain the

following files:

Option	Description
<code>config</code>	The master configuration file. This contains the current configuration for the HSM. It is always present in the directory.
<code>config-name</code>	An alternative configuration file saved by the system.
<code>config.new</code>	A hand-edited configuration file that can be read by the HSM.

You normally configure the HSM using the front panel controls. However, in some cases (for example, if you need to configure an HSM remotely, or if you are importing a number of clients), you may prefer to edit the exported configuration file and then re-import the file into the HSM. For more information, see:

- [About user privileges](#)
- [nShield Connect and client configuration files.](#)

6.2.3. Client configuration

The data that defines the configuration of the client hardserver is stored in a file on the client's file system.

You must load the configuration file for the configuration to take effect. For information about loading a client configuration remotely, see [Remote configuration of additional clients](#).

You can configure a client to use multiple HSMs. All the HSMs configured for use by a client can fail over if the application that uses them is set up appropriately.

For more information about the contents of the client configuration file, see [nShield Connect and client configuration files](#).



You can also configure the client's hardserver by setting environment variables, as described in [Setting environmental variables](#). Environment variable settings override settings in the client configuration files.

6.3. Basic HSM and remote file system (RFS) configuration

After installing the HSM hardware and software, there are several HSM and RFS configuration tasks you must perform. You perform these RFS tasks before:

- Creating the Security World and an Operator Card Set (OCS)

- Completing the process of configuring the HSM and client to work together.

6.3.1. Configuring the Ethernet interfaces

The HSM communicates with one or more clients. Each client is an Ethernet connected computer that has the Security World Software installed and configured. You must supply Internet Protocol (IP) addresses for the HSM and the client. Contact your system administrator for this information if necessary.

To configure the Ethernet interfaces (IPv4 and IPv6), see the *Connect Installation Guide*.

6.3.2. Optionally configure hardserver interfaces

By default, the hardserver listens on all interfaces. However, you can alter the hardserver settings. Altering the hardserver settings would prove necessary, for example, if you wanted to connect one of the Ethernet interfaces to external hosts.

Ensure that you have configured the Ethernet interfaces on the HSM before attempting to configure the hardserver. See the Installation Guide for more information about configuring the Ethernet interfaces.

You can configure the following options to specify network interfaces on which the hardserver listens:

Option	Description
All interfaces	This option (which is the default) specifies that the hardserver listens on all interfaces.
<i>IP address of interface #1</i>	This option specifies that the hardserver listens only on interface 1. This option only appears if interface 1 has been configured.
<i>IP address of interface #2</i>	This option specifies that the hardserver listens only on interface 2. This option only appears if interface 2 has been configured.
Will not listen	This option specifies that the hardserver does not listen on any interfaces.

To define the interface and port on which the hardserver listens:

1. From the main menu, select **System > System configuration > Hardserver config**. The following screen appears:

```

Hardserver config
Select network I/F
hardserver listens on:
All interfaces
Select TCP port: 9004
CANCEL          FINISH

```

2. Select the network interfaces on which the hardserver is to listen.



For security reasons, do not allow the hardserver to listen on any interface that is to connect to the public Internet.

3. Press the Select button to move to the TCP port field, and set the port on which the hardserver is to listen. The default is 9004.



Make sure that your firewall settings are consistent with your port settings. See the Installation Guide for more about firewall settings.

4. When the network interface and port are correct, press the right-hand navigation button.
5. Press the right-hand navigation button again to continue.
6. You are asked if you wish to reboot the system now or later. Press the right-hand navigation button to reboot now.

6.3.3. Configuring the Remote File System (RFS)

The RFS contains the master copy of the Security World data for backup purposes. The RFS can be located on either a client or another network-accessible computer where the Security World Software is installed. If the RFS is on a client, the same file structure also contains the configuration files for that client.



We recommend that you regularly back up the entire contents of the RFS. The **kmdata** directory is required to restore the nShield Connect, or a replacement, to its current state, in case of failure.

You can specify a new remote file system, and modify or delete an existing remote file system configuration. To create or modify a remote file system configuration, specify the IP address of the computer on which the file system resides.



You must have created an RFS on the client computer before you specify the IP address of the client.

For more information about the RFS and its contents, see:

- Remote file system (RFS)
- Location of Security World files.

The nShield Connect must be able to connect to TCP port 9004 of the RFS. If necessary, modify the firewall configuration to allow this connection on either the RFS itself or on a router between the RFS and the nShield Connect.

Obtain the following information about the nShield Connect before you set up an RFS for the first time:

- The IP address
- The electronic serial number (ESN)
- The hash of the K_{NETI} key (HK_{NETI}). The K_{NETI} key authenticates the nShield Connect to clients. It is generated when the nShield Connect is first initialized from factory state.

If your network is secure and you know the IP address of the nShield Connect, you can use the `anonkneti` utility to obtain the ESN and hash of the K_{NETI} key by giving the following command on the client computer. For guidance on network security, see the *nShield Security Manual*.

```
anonkneti <Unit IP>
```

In this command, *<Unit IP>* is the IP address of the nShield Connect, which could be one of the following:

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

The command returns output in the following form:

```
A285-4F5A-7500 2418ec85c86027eb2d5959fef35edc5e1b3b698f
```

In this example output, `A285-4F5A-7500` is the ESN and `2418ec85c86027eb2d5959fef35ed-c5e1b3b698f` is the hash of the K_{NETI} key.

Alternatively, you can find this information on the nShield Connect startup screen. Use the touch wheel to scroll to the appropriate information.

When you have the necessary information, set up an RFS as follows:

1. Prepare the RFS on the client computer (or another appropriate computer) by running

the following command on that computer:

```
rfs-setup <Unit IP> <EEEE-SSSS-NNNN> <keyhash>
```

In this command:

- *<Unit IP>* is the IP address of the nShield Connect.
- *<EEEE-SSSS-NNNN>* is the ESN of the nShield Connect.
- *<keyhash>* is the hash of the **K_{NETI}** key.

2. On the nShield Connect display screen, use the right-hand navigation button to select **System > System configuration > Remote file system**, and enter the IP address of the client computer on which you set up the RFS:

Remote File System

Enter IP address:

CANCEL CONTINUE

3. The next screen asks for the port number on which the RFS is listening:

Remote File System

Enter port number:
9004

CANCEL CONTINUE



Leave the port number at the default setting of 9004.

After you have defined the RFS, the nShield Connect configuration files are exported automatically. See the *User Guide* for more about configuration files.

To modify the RFS at a later date, select **System > System configuration > Remote file system**, and then select the required action.



You can allow other clients to access the remote file system and share Security World and key data that is stored in the **kmdata** directory in the same way as the HSM. Clients that access data in this way are described as *cooperating clients*. To configure client cooperation, you need to know the details of each client including IP address and ESN.

6.3.4. Configuring log file storage

You can choose to store log files on both the HSM and RFS or on the HSM only.

To configure log file storage, use the right-hand navigation button to select **System > System configuration > Log config**. Then select one of:

1. **Log** to store log files on the HSM only.
2. **Append** to store log files on both the HSM and remote file system.

We recommend selecting **Append** because if you select **Log** you can only view the log file from the Connect front panel. Moreover, the log file stored on the HSM is cleared every time it is powered down.

You may also additionally configure the logs to be sent to a remote syslog server, see [Configuring Remote Syslog](#).

6.3.5. Setting the time and date

If you do not intend to use NTP time synchronization, set the time and date as described in this section. If you configure the Connect to use NTP time synchronization, then the time and date will be maintained by NTP.

To set the time and date on the HSM as UTC:

1. Use the right-hand navigation button to select and display the **System** menu:

```
1-1
System configuration
System information
Login settings
Upgrade system
Factory state
Shutdown/Reboot
BACK                SELECT
```

2. Select **System configuration** to display the **System configuration** menu:

```
1-1-1
Network config
Hardserver config
Remote file system
Client config
Resilience config
Config file options
BACK                SELECT
```

3. Use the touch wheel to move the arrow to **Date/time setting**, and press the right-hand navigation button to select it. The **Set system date** screen is displayed:

```

Set system date
Please enter the
current UTC date as
DD/MM/YYYY:
 27/ 5/2013
CANCEL          NEXT

```

4. For each date field, use the touch wheel to set the value and move the cursor to the next field.

When you have completed all the fields, press the right-hand navigation button to confirm the date. The **Set system time** screen is displayed:

```

Set system time
Please enter the
current UTC time as
hour/mins/seconds:
 18:08:19
CANCEL          FINISH

```



Setting the time and date of the HSM as UTC does not reset the value of the Real Time Clock (RTC) on the HSM. The UTC date and time settings are used only in log messages.

6.3.6. Keyboard layout

You can connect a keyboard to the USB connector on the Connect front panel. This enables you to control the Connect using a special set of keystrokes instead of the standard front panel controls.

You can connect either a US or a UK keyboard. To configure the Connect for your keyboard type, select **System > System configuration > Keyboard layout** and then choose the keyboard type you require.

6.4. Configuring the Connect to use the client

You must inform the HSM hardserver of the location of the client computer.

You can configure the connection to use secure authentication using software-based authentication or with an nToken (or local HSM) installed in the client. When enabled the nShield Connect not only examines the client IP address, but also requires the client to identify itself using a signing key.



If an nToken is installed in a client, it can be used to both generate and protect a key that is used for the impath communication between the

Connect and the client. Thus a strongly protected key is used at both ends of the impath. A local HSM (Solo or Edge) can also be used to perform the role of the nToken.



Software-based authentication is only supported from version 12.60. Previously enrolled clients using software-based authentication will need to be re-enrolled if an earlier version of Security World software is installed.

The client configuration process varies slightly depending on whether you are enrolling the client with or without secure client authentication:

1. On the nShield Connect front panel, use the right-hand navigation button to select **System > System configuration > Client config > New client**.

The following screen is displayed:

Client configuration

Please enter your client IP address:

CANCEL NEXT

Enter the IP address of the client, and press the right-hand navigation button.

2. Use the touch wheel to confirm whether you want to save the IP or not, and press the right-hand navigation button.

Client configuration

Do you want to save the IP in the config?
(No for dynamic client IPs)

No

BACK NEXT

3. Use the touch wheel to select the connection type between the nShield Connect and the client.

Client configuration

Please choose the client permissions

Unprivileged

BACK

NEXT

The following options are available:

Option	Description
Unprivileged	Privileged connections are never allowed.
Priv. on low ports	Privileged connections are allowed only from ports numbered less than 1024. These ports are reserved for use by root on Linux.
Priv. on any ports	Privileged connections are allowed on all ports.



A privileged connection is required to administer the nShield Connect (for example, to initialize a Security World). If privileged connections are allowed, the client can issue commands (such as clearing the nShield Connect) which interfere with the normal operation of the nShield Connect. Entrust recommends that you allow only unprivileged connections unless you are performing administrative tasks.

4. When you have selected a connection option, press the right-hand navigation button.

The following screen is displayed:

```
Client configuration

Do you want secure
authentication enabled
on this client?

      Yes
BACK      NEXT
```

- a. Select **No** and press the right-hand navigation button to configure the client without secure authentication. The authentication of the client will be based on the IP address only.
 - b. Select **Yes** and press the right-hand navigation button to configure the client with secure authentication.
5. On the Connect, enter the number of the port on which the client is listening (the default is 9004), and press the right-hand navigation button. The following screen is displayed:

```
Client configuration

On what port is the
```

```

client listening?

          9004

CANCEL          NEXT

```

6. Skip this step if you have not selected secure authentication.

If an nToken is installed in the client, you will be asked to choose which authentication key to use. Select the desired option and press the right-hand navigation button:

```

>3138-147F-2D64
  Software Key

BACK          SELECT

```

- a. The ESN of the nToken installed in the client.
- b. "Software Key" for software-based authentication.

If no nToken is installed in the client, then software-based authentication is automatically selected.



Software-based authentication is only supported from version 12.60.

7. Skip this step if you have not selected secure authentication.

The next screen asks you to verify that the key hash displayed by the nShield Connect matches the client key hash:

```

Client 3138-147F-2D64
reported the key hash:
691be427bb125f387686
38a18bfd2eab75623320
Is this EXACTLY right?

CANCEL          CONFIRM

```

The client key hash is obtained by running the commands described below. Take a copy of the returned key hash and compare it to the value reported on the nShield Connect display.

With software-based authentication

Run the following command on the client:

```

enquiry -m0

```

This command returns the software key hash, tagged as **kneti hash**, as part of its output, for example:

```
Server:
  enquiry reply flags  none
  enquiry reply level  Six
  ...
  kneti hash           f8222fc007be38b78ebf442697e244dabded38a8
  ...
```

With nToken authentication

Run the following command on the client:

```
ntokenenroll -H
```

This command produces output of the form:

```
nToken module #1
nToken ESN:      3138-147F-2D64
nToken key hash: 691be427bb125f387686
                  38a18bfd2eab75623320
```

Check that the ESN also matches the one reported on the nShield Connect display.

If the client key hash matches the one reported on the nShield Connect display, press the right-hand navigation button to continue the RFS configuration. Otherwise press the left-hand navigation button to cancel the operation.

8. The Connect displays a message reporting that the client has been configured. Press the right-hand navigation button again.

To modify or delete an existing client, select **System > System configuration > Client config** and perform the appropriate procedure.

If you want to use multiple clients with the Connect, you must enable additional client licenses (see [Enabling optional features](#)). When you have additional client licenses enabled, to configure more clients, repeat the appropriate steps of the procedure described in this section for each client.

6.4.1. Remote configuration of additional clients

After you have configured the first client for your Connect and provided you have enabled auto push, you can add additional clients remotely.



Before you can use multiple clients with the Connect, you must enable

the additional clients as described in [Enabling optional features](#).

To add clients remotely:

1. Save the configuration file of the Connect `/opt/nfast/kmdata/hsm-ESN/config` as `config.new` in the same directory.
2. Edit the configuration file `config.new` and add a new entry to the `hs_clients` section to contain the details of the client to be added.

The following two entries must exist in the configuration file:

```
addr=<client_IP>
clientperm=permission_type
```

Where:

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

`0.0.0.0` or `::`, and blank result in the same behavior. You can only use them in the configuration file, you cannot enter these values in the front-panel user interface.

The default is blank.

If you set both the `<client_IP>` field (the client's IP address) and the key hash, the HSM must identify clients from both of these fields.

`permission_type` defines the type of commands the client can issue (`unpriv`, `priv` or `priv_lowport`).

- a. If the client is using an nToken, two additional entries will need to be added to the configuration file:

```
esn=nToken_ESN
keyhash=nToken_keyhash
```

Where `nToken_ESN` is the ESN of the client's nToken and `nToken_keyhash` is the hash of the key that the client's nToken should authenticate itself with.

- b. If the client is using software-based authentication, one additional entry will need to be added to the configuration file:

```
keyhash=software_keyhash
```

Where `software_keyhash` is the hash of the software generated key that the client

should authenticate itself with. The **ESN** entry must be blank or omitted for software-based authentication.



Each client entry after the first must be introduced by a line consisting of one or more hyphens.

3. Load the updated configuration file on to the Connect. To do this, run the following command:

```
cfg-pushnethsm --address=<module_IP_address> <new_config_file>
```

In this command, **<module_IP_address>** is the IP address of the Connect and **<new_config_file>** is the location of the updated configuration file (**config.new**). If **module_IP_address** is an IPv6 address, then enclose it within square brackets, for example **[fc00::1]**.

With auto push enabled, you can load the configuration file using the Connect front panel. The configuration file (**config.new**) must be in the directory **/opt/nfast/kmdata/hsm-ESN/config** on the remote file system. To do this, select **System > System configuration > Config file options > Fetch configuration**.



An SEE machine cannot be installed or configured using the fetch configuration option from the front panel. The auto push feature must be used for this. See [Remotely loading and updating SEE machines](#) for more information.

6.5. Configuring client computers to use the Connect

Each client computer must be configured to use the internal security module of your Connect. There are two methods for achieving this:

- Enrolling the client with the configuration file.
- Enrolling the client with command-line utilities.

6.5.1. Enrolling the client with the client configuration file

The client configuration files are in the directory **/opt/nfast/kmdata/config** on the client computer's file system.



For this release, you must generate a new client configuration file to take advantage of the new functionality. To generate a new client con-

figuration file, back up your existing configuration file and run the command `cfg-mkdefault`. This generates a template for the configuration file into which you can copy the settings from your old configuration file.

The `nethsm_imports` section defines the network HSMs that the client imports (See `nethsm_imports`). It can also be set up by the `nethsmenroll` utility.

- Edit the following mandatory fields: `local_module`, `remote_ip`, `remote_esn`, `remote_keyhash` and `privileged`. The default value for `remote_keyhash` (40 zeros) specifies that no authentication should occur.
- The `ESN` and `hash` of the HSM to import can be retrieved by running the command

```
anonkneti remote_ip
```

- If the client is to be enrolled with an nToken, open a command line window, and run the command: `ntokenenroll --H`. This command produces output of the form:

```
nToken module #1
nToken ESN: 3138-147F-2D64
nToken key hash: 691be427bb125f3876838a18bfd2eab75623320
```

- Enter the nToken's `ESN` in the field `ntoken_esn` in the config file.
- Each HSM entry after the first must be introduced by a line consisting of one or more hyphens, i.e. `---`.
- At the command line run the command `cfg-reread`, to reload the hardserver's configuration.
- Verify that the client can use the Connect by running `enquiry`, which reports the HSM's status.



If the client is to be enrolled with either software-based authentication or no authentication, the `ntoken_esn` field must be left **empty**.

For information about configuration file contents, see [nShield Connect and client configuration files](#).

6.5.2. Enrolling the client from the command line

The `nethsmenroll` command-line utility edits the client hardserver's configuration file to add the specified Connect. For more information about the options available to use with `nethsmenroll`, read the following section [Client configuration utilities](#), or run the command:

```
nethsmenroll --help
```

To retrieve the Connect's **ESN** and **HKNETI**, run the command

```
anonkneti <Unit IP>
```

This command produces output of the form:

```
3138-147F-2D64 691be427bb125f38768638a18bfd2eab75623320
```

If the Connect's **ESN** and **HKNETI** are not specified, **nethsmenroll** attempts to contact the HSM to determine what they are, and requests confirmation.

1. If you are enrolling the client *with* an nToken, run the command:

```
nethsmenroll --ntoken-esn <nToken ESN> [Options] --privileged <Unit IP> <Unit ESN> <Unit KNETI HASH>
```

2. If you are enrolling the client *without* an nToken, i.e. software-based authentication or no authentication, run the command:

```
nethsmenroll [Options] --privileged <Unit IP> <Unit ESN> <Unit KNETI HASH>
```

These commands produce output of the form:

```
OK configuring hardserver's nethsm imports.
```

6.5.3. Client configuration utilities

We provide the following utilities for client configuration:


Utility	Description
nethsmenroll	This utility is used to configure the client to communicate with the Connect.
config-serverstartup	This utility is used to configure the client's hardserver to enable TCP sockets.

6.5.3.1. nethsmenroll

The **nethsmenroll** command-line utility edits the client hardserver's configuration file to add the specified Connect. If the Connect's **ESN** and **HKNETI** are not specified, **nethsmenroll** attempts to contact the Connect to determine what they are, and requests confirmation.

Usage:

```
nethsmenroll [Options] --privileged <nShield Connect IP> <nShield Connect ESN> <nShield Connect KNETI HASH>
```

Options	Description
<code>-m, --module=MODULE</code>	Specifies the local HSM number to use (default is <code>0</code> for dynamic configuration by hardserver).
<code>-p, --privileged</code>	Causes the hardserver to request a privileged connection to the Connect (default <code>unprivileged</code>).
<code>-<nShield Connect IP></code>	The IP address of the nShield Connect, which could be one of the following: <ul style="list-style-type: none"> • An IPv4 address, for example <code>123.456.789.123</code>. • An IPv6 address, for example <code>fc00::1</code>. • A link-local IPv6 address, for example <code>fe80::1%eth0</code>. • A hostname.
<code>-r, --remove</code>	Deconfigures the specified Connect.
<code>-f, --force</code>	Forces reconfiguration of an Connect already known.
<code>--no-hkneti-confirmation</code>	Does not request confirmation when automatically determining the Connect's <code>ESN</code> and <code>HKNETI</code> . <div>  <p>This option is potentially insecure and should only be used on secure networks where there is no possibility of a man-in-the-middle attack. For guidance on network security, see the <i>nShield Security Manual</i>.</p> </div>
<code>-V, --verify-nethsm-details</code>	When the <code>ESN</code> and <code>HKNETI</code> have been provided on the command line, verifies that the selected HSM is alive, reachable and matches those details.
<code>-P, --port=PORT</code>	Specifies the port to use when connecting to the given Connect (default <code>9004</code>).
<code>-n, --ntoken-esn=ESN</code>	Specifies the <code>ESN</code> of the nToken to be used to authenticate this client. If the option is omitted, then software authentication will be used instead.

6.5.3.2. config-serverstartup

The `config-serverstartup` command-line utility automatically edits the `[server_startup]` section in the local hardserver configuration file in order to enable TCP ports for Java and KeySafe. Any fields for which values are not specified remain unchanged. After making any

changes you are prompted to restart the hardserver.

Run **config-serverstartup** using commands of the form:

```
config-serverstartup [OPTIONS]
```

For more information about the options available to use with **config-serverstartup**, run the command:

```
config-serverstartup --help
```

6.6. Configuring NTP in the Connect

The Connect has a standard NTP client that can be configured to support synchronization of system time on the Connect with one or more NTP servers. Network Time Protocol (NTP) is intended to synchronize all participating computers to within a few milliseconds of Coordinated Universal Time (UTC). System time on the Connect is independent of the Real Time Clock (RTC) in the HSM and is used for log messages and front panel display.



Entrust recommends that the NTP Server(s) are trusted servers within your local network, not internet time servers.


After configuring NTP the Connect has to be re-started for the configuration to take effect. When starting up after being configured, the NTP client can make a step change to the system time to bring it into line with that of the NTP server(s). At all other times, the NTP client will only slew (gradually change) the system time. When using NTP there should be no need to set the system time by setting time and date from the front panel of the Connect.

Before configuring NTP you must ensure the following:

- **auto push** is enabled for the client computer you wish to use for configuring NTP, see [Configuring auto push](#).
- The client computer enabled for **auto push** is configured for Privileged connections, see [Configuring the Connect to use the client](#), so that the Connect can be rebooted from the client computer.

6.6.1. Using the NTP configuration tool

NTP is configured using the **cfg-pushntp** utility on a client computer.

Utility	Description
<code>cfg-pushntp</code>	<p>This tool enables or disables NTP time synchronization on the specified Connect. When enabling NTP synchronization, the IP addresses of up to 3 NTP servers may be specified.</p> <div>  <p>The new NTP settings will take effect the next time the target Connect is restarted.</p> </div>

Usage:

```
cfg-pushntp -a ADDR [-p PORT -k -m MODULE] -1 ADDR [-2 ADDR -3 ADDR] enable
```

```
cfg-pushntp -a ADDR [-p PORT -k -m MODULE] disable
```

Options:

Field	Description
<code>enable disable</code>	Enable or disable NTP service on the Connect.
<code>-a, --address=ADDR</code>	IP address of Connect to configure NTP on.
<code>-p, --port=PORT</code>	Set port to use to connect to the Connect (default=9004).
<code>-k, --use-kneti</code>	Use KNETI to authenticate ourselves.
<code>-m, --module=MODULE</code>	Set the HSM to use for KNETI authentication. The default is HSM 1. This option can only be used with the <code>--use-kneti</code> option
<code>-1, --ntp1=ADDR</code>	IP address of NTP server.
<code>-2, --ntp2=ADDR</code>	IP address of NTP server.
<code>-3, --ntp2=ADDR</code>	IP address of NTP server.

For example, running the command:

```
cfg-pushntp --address=192.30.100.150 --ntp1=192.23.24.256 enable
```

Returns:

The requested NTP configuration changes have been uploaded and will take effect when the target Connect is restarted.

6.6.2. Restarting the Connect

After configuring NTP, restart the Connect, for example see [Using nethsmadmin to reboot an Connect](#). Once the Connect has rebooted and the syslog output is available, check that there are no NTP failures reported in the syslog output.

6.7. Configuring Remote Syslog

The Connect can be configured to send logs directly to a remote syslog server, listening on a User Datagram Protocol (UDP) port, by editing the `remote_sys_log` section of the config file.

This behavior can be configured in addition to storing the log files on the RFS (i.e. you can configure the logs to be sent to a remote syslog server regardless of whether the Connect logs are configured to be stored on the HSM or the RFS). For further information see [Configuring log file storage](#).

There is no additional formatting of log messages (the logs sent are the same log messages that will appear on the unit or the RFS). It is the responsibility of the remote syslog server / SIEM application to format, sort and aggregate the incoming log messages.

To configure an nShield Connect to send logs to a remote syslog server:

1. In the `remote_sys_log` section of the config file for the remote module, add the following settings:



If your configuration file predates the functionality to send logs to a remote syslog server, you will need to manually create a new config file section called `remote_sys_log`.

```
remote_sys_log_ip=REMOTE_SYSLOG_SERVER_IP
remote_sys_log_port=REMOTE_SYSLOG_SERVER_PORT
```

2. Run the following command to push the new config file to the module:

```
cfg-pushnethsm
```



If you are using an older version of the Security World software with an Connect image that supports remote syslog, you will see this error message: **unrecognized section name: 'remote_sys_log'**. Use the following command to push the updated configuration file:

```
cfgpushnethsh --force
```



If you are using a version of the Security World software that supports remote syslog with an Connect image that does not support remote syslog, the configuration file will be pushed to the Connect but will be rejected by the Connect. You will see that the upgraded configuration file on the RFS is unchanged.

To turn off sending logs to a remote syslog server, remove the entries from the `remote_sys_log` section of the configuration file and push the updated configuration file.

6.8. Setting up client cooperation

If you do not need to allow multiple clients access to your remote file system (RFS), you only need to follow the instructions provided in [Configuring the Remote File System \(RFS\)](#) to initialize your RFS. If you need to allow other clients to access your RFS (that is, able to access the RFS to share key data), complete the following steps:

1. Configure the RFS to accept access by cooperating clients:

- For every authenticated client (with write access and K_{NETI} authorization) that needs to be a client of this remote file system, run the command:

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

In this command:

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

+

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



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

1. On each client that is to be a cooperating client, you must run the `rfs-sync` command-

line utility with appropriate options:

- for clients using a software K_{NET} key to authenticate themselves to the RFS, run the command with the default options:

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

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

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

In this command:

- **<RFS_IP_ADDRESS>** is the IP address of the RFS.
- **<MODULE>** is the local module to use for authentication.

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

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

+ For more information about the **rfs-sync** utility, see [rfs-sync](#).

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

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

+

```
remote_ip=<client_IP_address>
remote_esn=keyhash=0000000000000000000000000000000000000000000000000000000000000000
native_path=/opt/nfast/kmdata/local
volume=kmdata-local
allow_read=yes
allow_write=yes
allow_list=yes
is_directory=yes
is_text=no
```

and

```
remote_ip=<client_IP_address>
remote_esn=keyhash=0000000000000000000000000000000000000000000000000000000000000000
native_path=/opt/nfast/kmdata/local/sync-store/
volume=kmdata-backup
allow_read=yes
allow_write=yes
allow_list=yes
is_directory=yes
is_text=no
```

6.8.1. Useful utilities

6.8.1.1. anonkneti

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

The IP address could be one of the following:

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

6.8.1.2. rfs-sync

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

6.8.1.2.1. Usage

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

6.8.1.2.2. Options

-U, --update

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



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

`-c, --commit`

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

`-s, --show`

These options display the current synchronisation configuration.

`--setup`

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

`-a, --authenticate`

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

`-m, --module=module`

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

`-p, --port=port`

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

`ip_address`

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

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

`--remove`

This option removes the synchronisation configuration.

A client can use `rfs-sync --show` to display the current configuration, or `rfs-sync --remove` to revert to a standalone configuration. Reverting to a standalone configuration leaves the

current contents of the Key Management Data directory in place.

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



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

6.8.2. Setting environmental variables

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

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



Ensure that all variables are exported as well as set.

6.8.3. Logging and debugging

The Connect and applications that use it generate log files. You can view the logs using the unit front panel. Application log messages are handled on the client.

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

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



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

given default values.

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

Some components of the Security World Software generate separate debugging information which you can manage differently. Debugging information for applications is handled on the client. If you are setting up the unit to develop software that uses it, you should configure debugging before commencing software development.

6.8.4. Configuring Java support for KeySafe

To use KeySafe, follow the instructions in [Using KeySafe](#).

6.9. Routing

If you have configured only one network interface, you do not need to configure a static route for the unit, although you can do so if you wish. If you have configured a second network interface, you can choose to configure a static route.

If the unit is to connect to a remote host or network that is unreachable through the default gateway, you must set up extra static routes in the system routing table.

To set up the Ethernet interfaces (IPv4 and IPv6), see the *Connect Installation Guide*.

After you have defined static routes, you should test them as described in [Testing routes](#).



If you define, edit, or delete a route, you must reboot the unit before the route can be used and the routing table is updated.

6.9.1. Testing routes

When you have set up or edited a route, you should test the route.

6.9.1.1. Testing a route between the unit and the client

When you have installed the unit in its final location, you should test the connection between the unit and the client. You can do this from the client, as described in this section, or by using the **Ping remote host** option on the unit. To do this from the unit, select **System > System configuration > Network config > Ping remote host**.

You can also use the method described in this section to test the route between the client and a remote computer.

To test the route from the client to the unit, issue a **ping** command from the client for the IP address that you specified for the unit. (The format of the command and results may vary according to the platform that you are using.)

```
ping <xxx.xxx.xxx.xxx>
```

If the **ping** operation is successful, a message similar to the following is displayed:

```
Pinging xxx.xxx.xxx.xxx with 32 bytes of data
Reply from xxx.xxx.xxx.xxx: bytes=32 time=10ms TTL=125
Reply from xxx.xxx.xxx.xxx: bytes=32 time=10ms TTL=125
Reply from xxx.xxx.xxx.xxx:10ms TTL=125
Reply from xxx.xxx.xxx.xxx:10ms TTL=125
```

6.9.1.2. Testing a route between the unit and a remote host

When you have defined or edited a route from the unit to a remote computer, you should test it. To do this you can issue a **ping** command from the unit to the IP address of the host.

You can also use this method to test the connection between the unit and the client.

To test a route from the unit to a host:

1. Select **System > System configuration > Network config > Ping remote host**. The following screen appears:

```
Ping remote host
Select IP address:
  0. 0. 0. 0
RESET FINISH
```

2. Enter the IP address of the remote host.
3. Press **FINISH** to issue the ping. The following message appears:

```
Please wait, running ping
```

4. After a short wait, a display similar to the following should appear, showing that the unit has managed to communicate with the host:

```
Ping xxx.xxx.xxx.xxx:
#0: rtt=0.0503 ms
#1: rtt=0.0503 ms
#2: rtt=0.0503 ms
```

```
#3: rtt=0.0503 ms
4 of 4 packets back.
min avg max SD
0.29 0.36 0.50 0.09 ms
```

If not all of the information is visible onscreen, use the touch wheel to scroll up and down the page.

5. Press the left-hand navigation button to return to the **Network config** menu.

6.9.2. Tracing network routes

You can trace network routes from the unit and from clients.

6.9.2.1. Tracing the route from the unit

You can trace the route taken from the unit to a remote computer. You can also use this method to trace the route from the unit to the client.

1. Select **System > System configuration > Network config > Trace route to host**. The following screen appears:

```
Trace route
Select IP address:
0.0.0.0
CANCEL FINISH
```

2. Press the right-hand navigation button to issue the **tracert**. The following message appears:

```
Please wait, running traceroute.
```

3. After a short wait, a display similar to the following should appear, showing the IP addresses encountered along the route:

```
1: xxx.xx.xxx.x
0.40 ms
2: *
3: xx.xxx.xx.xxx
2.1 ms
4: xxx.xx.xxx.xxx
2.4 ms
BACK
```

If not all of the information is visible onscreen, use the touch wheel to scroll up and down the page.

4. Press the left-hand navigation button to return to the **Network config** menu.

6.9.2.2. Tracing the route from the client

You can trace the route from the client to the unit or (if the client is connected to the public Internet) to a remote computer.

To trace the route from the client to the unit, issue a **tracert** command from the client for the IP address of the unit. (The format of the command, and results, may vary depending upon the platform that you are using.)

```
tracert <xxx.xxx.xxx.xxx>
```

After a short wait, a display similar to the following should appear, showing the IP addresses encountered along the route:

```
Tracing route to modulename (xxx.xxx.xxx.xxx) over a maximum of 30 hops:
1 xxx.xxx.xxx.xxx (xxx.xxx.xxx.xxx) 1.457 ms 0.513 ms 0.311 ms
2 xxx.xxx.xxx.xxx (xxx.xxx.xxx.xxx) 0.773 ms 0.523 ms 1.615 ms
```

6.9.2.3. Displaying the routing table

You can view details of all the IP addresses for which the internal security module has a route stored. The routing table includes entries for static routes (which are stored permanently) and local hosts to which the module has set up temporary routing entries.

To view the routing table:

1. Select **System > System configuration > Network config > Show routing table**. A display similar to the following appears:

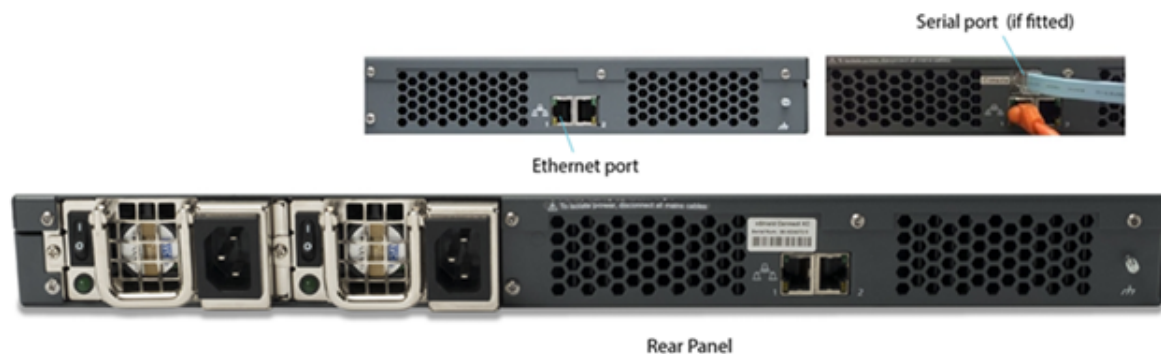
```
Dest Gateway Flg
1 xxx.xxx.xxx.xxx
  xxx.xxx.xxx.xxx UG
2 xxx.xx.xx.x
  xxx.x.x.xxx UG
BACK
```

If not all of the information is visible on the unit display screen, use the touch wheel to scroll up and down the page.

2. Press the left-hand navigation button to return to the **Network config** menu.

6.10. Configuring an Connect using the Serial Console

On supported Connect hardware variants (see *Model numbers* in the [nShield Security World: nShield Connect v12.81 User Guide \(Linux\)](#)) there is an RJ45 serial port connector at the rear of the Connect marked **Console**. The serial port provides access to a serial console command line interface that enables remote configuration of the unit whilst also facilitating status monitoring. Tasks which would typically require a physical presence in front of the HSM, including setting IP addresses, can be done remotely using the serial console.



This functionality can help provide separation of duty between the data center technician installing the Connect and the administrator configuring and using the HSM. The only required local activity is to connect the Connect to power and to serial and Ethernet ports. Everything that can be configured using the front panel can then be configured remotely either over the serial interface, by using the `nethsmadmin` utility (see [nethsmadmin](#)) or by pushing an updated configuration file to the nShield Connect (see [Configuring the Connect with configuration files](#)).

The Serial Console supports IPv4 and IPv6 addresses.

6.10.1. Serial port configuration

The RJ45 connector can be directly connected to your client machine or connected to a serial port aggregator for remote access.

To access the serial console command line interface, first determine the device name of the serial connection once it is connected to your client machine. Then configure the serial port connection in your serial port communication software with the following parameters:

- Line Speed (baud): 9600
- Data bits: 8
- Parity: None
- Stop bits: 1.

Once the connection is established, press **Return** until a login prompt is displayed. The login prompt should look like:

```
nethsm login:
```

6.10.1.1. Troubleshooting

Error	Action
Nothing on the screen	<p>Press Enter a few times.</p> <p>Make sure that the RJ45 connector is properly connected and that remote configuration is enabled on the Connect, see Enabling and disabling the serial console.</p>
Miscellaneous characters displayed on the screen	<p>Make sure the serial port connection is configured correctly, see Serial port configuration.</p>

6.10.2. Creating a serial console session

The username for accessing the serial console is **cli** and the default password is **admin**.

On first login you will be prompted to change the password for the **cli** user. The minimum length of the new password is 5 characters. For guidance on selecting a password, see the *Connect Security Manual*.

Once you are successfully logged in to a serial console session you will see the welcome message:

```
Welcome to the Connect Serial Console. Type help or ? to list commands.
```

```
(cli)
```

The serial console session will automatically logout after 180 seconds of inactivity. To manually end a session, use the **logout** command.

6.10.3. Enabling and disabling the serial console

The serial console interface can be enabled and disabled using the Connect front panel.

- To enable the serial console interface, navigate to **System > System configuration > Remote config options > Serial Console** and set to **On**.
- To disable the serial console interface, set **Serial Console** to **Off**.

The serial interface is enabled by default and will turn back on with the default password if the unit is reset to its factory state. This means if you do not want the serial console enabled you will need to disable it after each factory state.

If you do not see the menu option **System > System configuration > Remote config options > Serial Console** on the front panel then this means that your Connect does not support the serial console feature (the hardware does not support serial access).

The availability of the serial console feature can also be checked remotely from an enrolled client by running the enquiry utility.

Feature availability	Enquiry output
Serial console available	... level six flags SerialConsoleAvailable ...
Serial console not available	... level six flags none ...

6.10.4. Serial console commands

The serial console command line interface provides the following commands:

Command	Description
<code>date</code>	Get/set the HSM system time

Command	Description
<code>enquiry</code>	Prints enquiry data from the HSM
<code>esn</code>	Show the Electronic Serial Number (ESN) of the HSM
<code>factorystate</code>	Reset unit to its original (factory) state  This will reset the IP and serial console settings
<code>gateway</code>	Get/set the default IPv4 gateway address
<code>gateway6</code>	Get/set the default IPv6 gateway address
<code>help</code> or <code>?</code>	List available commands with <code>help</code> or detailed help with <code>help cmd</code>
<code>kneti</code>	Show the (hash of) Kneti (used for enrolling the HSM with clients)
<code>logout</code>	Log out of the Connect Serial Console
<code>netcfg</code>	Get/set the IPv4 network interface configuration
<code>netcfg6</code>	Get/set the IPv6 network interface configuration
<code>netlink</code>	Get/set network interface link, Get network interface MAC address
<code>bondcfg</code>	Get/set HSM bond interface configuration
<code>bondlink</code>	Get/set bond interface link
<code>passwd</code>	Set the serial console password
<code>ping</code>	Ping a remote host
<code>push</code>	Get/set the config push setting
<code>reboot</code>	Reboots the HSM
<code>rfsaddr</code>	Get/set the RFS IP address and port
<code>route</code>	Get/set IPv4 network routes
<code>route6</code>	Get/set IPv6 network routes
<code>routing</code>	Show IPv4 routing table
<code>routing6</code>	Show IPv6 routing table
<code>stattree</code>	Run the <code>stattree</code> command
<code>sworldcheck</code>	Check for any Security World data on the HSM
<code>tamperlog</code>	Show the Connect tamper log
<code>uptime</code>	Show how long the Connect has been running (since last boot)
<code>version</code>	Show Connect Serial Console version information

For additional help on a command, run `help` command to see additional guidance on command usage, syntax and parameter documentation.

6.11. Enabling optional features

nShield Connect supports a range of optional features. Optional features must be enabled with a certificate that you order from Entrust. You can order the features when you purchase a unit, or you can obtain the certificate at a later date and use the front panel of the unit to enable the features.

For more information about:

- Ordering optional features, see [Ordering additional features](#)
- Feature-enabling procedures, see [Enabling features](#).

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

Most features are *static*; that is, they are enabled by means of a switch in the **EEPROM** of the unit. A dynamic feature must be enabled again if the unit is reinitialized.

Some optional features are *static*; that is, they are enabled by means of a switch in the **EEPROM** of the unit. A dynamic feature must be enabled again if the unit is reinitialized.



If you are enabling the Remote Operator feature, you must enable it on the unit that is to be used as the unattended unit. For information about Remote Operator, see [Remote Operator](#).

6.11.1. Available optional features

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

6.11.1.1. Elliptic Curve

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

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

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

6.11.1.2. Elliptic Curve activation

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

Contact Sales if you require an EC activation.

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

6.11.1.3. Elliptic Curve support on the nShield product line

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

HSM module type	Elliptic Curve support		Elliptic Curve offload acceleration ³	
	Named curves ²	Custom curves ^{1, 5}	Named curves ²	Custom curves ^{1, 5}
nShield Edge (Windows only)	Yes	Yes	No	No
nShield Solo 500 and 6000 nShield 500, 1500 and 6000	Yes	Yes	No	No
nShield Solo 500+, 6000+ nShield 6000+	Yes	Yes	Yes, Prime curves and twisted Brainpool curves are accelerated ⁴ .	Yes
nShield Solo XC nShield Solo XC	Yes	Yes	Yes, Prime curves and both twisted and non-twisted Brainpool curves are accelerated ⁴ .	Yes

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

²Both Prime and Binary named curves are supported. Refer to [Named Curves](#), below, which

lists the most commonly supported elliptic curves.

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

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

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

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

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

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

6.11.1.5. Named Curves

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

Supported named curves			
ANSIB163v1	BrainpoolIP160r1	NISTP192	SECP160r1
ANSIB191v1	BrainpoolIP160t1	NISTP224	SECP256k1
	BrainpoolIP192r1	NISTP256	
	BrainpoolIP192t1	NISTP384	
	BrainpoolIP224r1	NISTP521	
	BrainpoolIP224t1	NISTB163	
	BrainpoolIP256r1	NISTB233	
	BrainpoolIP256t1	NISTB283	
	BrainpoolIP320r1	NISTB409	
	BrainpoolIP320t1	NISTB571	
	BrainpoolIP384r1	NISTK163	
	BrainpoolIP384t1	NISTK233	

Supported named curves			
	BrainpoolP512r1	NISTK283	
	BrainpoolP512t1	NISTK409	
		NISTK571	

6.11.1.6. Custom curves

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

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

6.11.1.7. Further information on using elliptic curves

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

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



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

6.11.1.8. Secure Execution Engine (SEE)

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

SEE Activation (EU+10)	This SEE feature is provided with the CodeSafe developer product to enable you to develop and run SEE applications. The CodeSafe developer product is only available to customers in the Community General Export Area (CGEA, also known as EU+10). Contact Entrust to find out whether your country is currently within the CGEA.
------------------------	--

SEE Activation (Restricted)	This SEE feature is provided with specific products that include an SEE application. This feature enables you to run your specific SEE application and is available to customers in any part of the world.
-----------------------------	--

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

6.11.1.9. Remote Operator support

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

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

For more information about using Remote Operator, see [Remote Operator](#).

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

6.11.1.10. ISO smart card Support (ISS)

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

6.11.1.11. Korean algorithms

This feature enables the following mechanisms:

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

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

- SEED, which is a block cipher.

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

6.11.1.12. Fast RNG for ECDSA

Utilise a faster alternative for Random Number Generation (RNG) for Elliptic Curve Digital Signature Algorithm (ECDSA). This feature is applicable for Solo XC / Connect XC modules only.

The faster performance, comparable with V12.40 performance, is achieved by the RNG part of ECDSA being done on the NXP C291 Crypto Coprocessor.

This implementation of ECDSA uses an RNG that is not within scope for the Connect certifications and for this reason it will not be used when the HSM is in a fips-140-2-level-3 or common-criteria-cmts Security World (regardless of the feature bit setting).

6.11.1.13. Client licenses

You can purchase additional client licenses that allow you to run multiple clients for the unit. Three clients are always enabled on each unit.

6.11.2. Ordering additional features

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

- If possible, make a note of the unit serial number. This can be found on the base of the unit.
- Make a note of the Electronic Serial Number of the unit. You can find this from the front panel menu, by going to **HSM > HSM information > Display details**.

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

If you prefer, you can include this information in an e-mail to Sales.

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

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

The Feature Enabling Certificate contains the information that you need to enable the fea-

tures you have ordered.

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

6.11.3. Enabling features

Feature enabling differs for static and dynamic features.

- You can enable static features from the front panel of the unit or from the client.
- Entrust recommends that you enable dynamic features from the client. If the dynamic feature applies directly to the Connect, for example client licenses, you can use a **nethsadmin** option to apply them. See [Remotely enabling dynamic feature certificates including nShield Connect client licenses](#)



When enabling static feature(s) from the front panel, either using a card or a file, the module is cleared without warning. This will cause the HSM to drop or restart any SEE machine, and lose all the application keys that were loaded. In some cases, applications may need to be restarted.

6.11.3.1. Viewing enabled features

To see which (if any) features have already been enabled on the Connect, from the main menu select **HSM > HSM feature enable > View current state**.

To print this list to a file on the unit, select **HSM > HSM feature enable > Write state to file**. The resulting file is transferred when the unit configuration is pushed to the remote file system. You can find it in `/opt/nfast/kmdata/hsm-ESN/features/fet.txt`.

6.11.3.2. Enabling features with a smart card

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

1. Insert the Feature Enabling smart card into the unit's slot.
2. From the front panel, select **HSM > HSM feature enable > Read FEM from card**.

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

6.11.3.3. Enabling features without a smart card

You can also provide the Feature Enabling Certificate information supplied by Entrust from a file.

To enable a feature without a smart card:

1. Put the file that contains the feature enabling certificate in `/opt/nfast/kmdata/hsm-ESN/features` on the remote file system. In this path, `ESN` is the ESN of the module.

You can give the file any name that you wish. You must enter the file name on the unit's front panel, so you may prefer to use a short name.

2. From the front panel, select **HSM > HSM feature enable > Read from a file**.
3. Specify the name of the file that contains the certificate.

If the feature is enabled successfully, a message confirms this.

6.11.4. Using multiple modules

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

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

6.11.4.1. Identifying modules

Modules are identified in two ways:

- By serial number
- By `ModuleID`.

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

6.11.4.2. Electronic Serial Number (ESN)

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

6.11.4.2.1. ModuleID

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

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

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

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

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

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

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

6.11.4.3. Adding a module

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

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



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

1. Install the module hardware. Refer to the *Installation Guide* for information on installing nShield hardware.

2. Run the script `/opt/nfast/sbin/install`.
3. Add the module to the Security World. Refer to [Adding or restoring an HSM to the Security World](#).

6.11.4.4. Module fail-over

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

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

6.12. Stopping and restarting the hardserver

If necessary, you can stop the hardserver on the client, and where applicable the Remote Administration Service, by running the following command:

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

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

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

6.13. Resetting and testing the Connect

6.13.1. Default configuration

To reset the unit to its default configuration, select **System > System configuration > Default config** and confirm that you want to set the default configuration.

This removes the configuration of the module but does not change its K_{NET} .

6.13.2. Factory state

To reset the unit to its original (factory) state, select **Factory state** from the main menu and confirm that you want to return the unit to its factory state.

This gives a new K_{NETI} to the unit, which means that you must update the keyhash field of the unit's entry in the `nethsm_imports` section of the configuration file of all the clients that use it.

For more information about:

- The contents of the configuration files, see [Module and client configuration files](#)
- Configuring a new remote file system for the unit, see [Configuring the Remote File System \(RFS\)](#).

6.13.3. Testing the installation

To test the installation and configuration, follow these steps:

1. Log in on the client computer as a regular user, and open a command window.
2. Run the command:

```
opt/nfast/bin/enquiry
```

A successful `enquiry` command returns an output of the following form:

```
Server:
enquiry reply flags  none
enquiry reply level Six
serial number       #####-####-####
mode                 operational
version              #-#-#
speed index          #####
rec. queue           #####.####
---
version serial      #
remote port (PV4)   #####

Module ##:
enquiry reply flags  none
enquiry reply level Six
serial number       #####-####-####
mode                 operational
version              #-#-#
speed index          #####
rec. queue           ##.####
---
rec. LongJobs queue ##
SEE machine type     PowerPCELF
supported KML types  DSAp1024s160 DSAp3072s256
hardware status      OK
```

If the mode is **operational**, the unit is installed correctly.

If the `enquiry` command says that the unit is not found:

- a. Restart the client computer.
- b. Re-run the `enquiry` command.

7. Security World Remote Administration

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

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

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

Remote Administration enables:

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

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

7.1. Remote Administration components

Remote Administration consists of a number of components:

7.1.1. Remote Administration software

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

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

The Remote Administration Service should be installed on a client machine of the HSMs in your Security World that you can make accessible to Remote Administration Clients.

+

See [Remote Administration Service](#) for more information.

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

7.1.2. Security World programs and utilities

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

The `nethsmadmin` tool provides many of the Remote Administration capabilities for an Connect without accessing the front panel.

For more information, see [nethsmadmin](#).

7.1.3. nShield Remote Administration smart cards

You must use nShield Remote Administration Cards with Remote Administration. These are

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

The nShield Remote Administration smart cards provide:

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

The nShield Remote Administration smart cards are FIPS 140-2 Level 3 certified devices, supporting execution of a custom Java Applet developed by Entrust. The smart cards used with previous versions of Security World software and nShield HSMs are still useable but, as previously, only in an HSM's local slot. Remote Administration smart cards can be used both remotely and in an HSM's local slot.

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



Existing Administrator smart cards can be migrated to new Remote Administration smart cards using the **racs** (replace administrator card set) utility. Similarly existing OCS can be migrated using the **rocs** (replace operator card set) utility, provided the Security World has recovery enabled and the keys protected by that OCS are recovery enabled.



7.2. Authorized Card List

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

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

For more information on the Authorized Card List, see [Authorized Card List](#).



When administrative operations involving Remote Administration smart cards are initiated from an Connect Front Panel, the Connect fetches the Authorized Card List from the RFS.



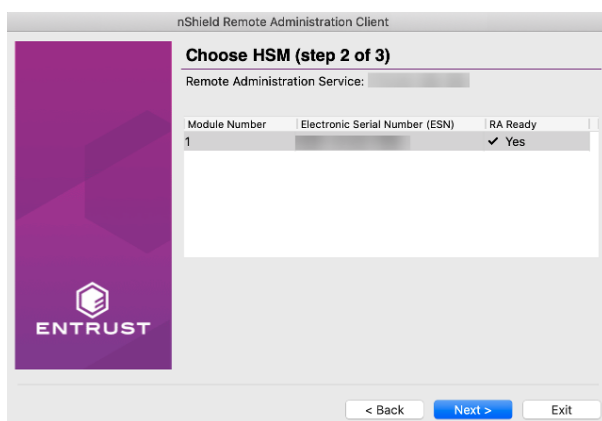
It is necessary to keep the Authorized Card List in sync by copying the

file between the RFS and clients manually.

7.3. Remote Administration Client

The Remote Administration Client (RAC) is a utility that enables you to select an HSM located elsewhere from a list provided by the Remote Administration Service (RAS), and associate an nShield Trusted Verification Device attached to your computer with the HSM.

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



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

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

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

Windows 8.1 + only

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

7.4. Remote Administration Service

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

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

The RAS participates as a crypto client of the HSMs. As such, the server used to host this software component must be a licensed client of the Connects being remotely administered. If your Remote File System (RFS) is already a licensed client, the RAS can be collocated on the RFS server without needing to purchase an additional client license.

7.5. nShield Trusted Verification Device

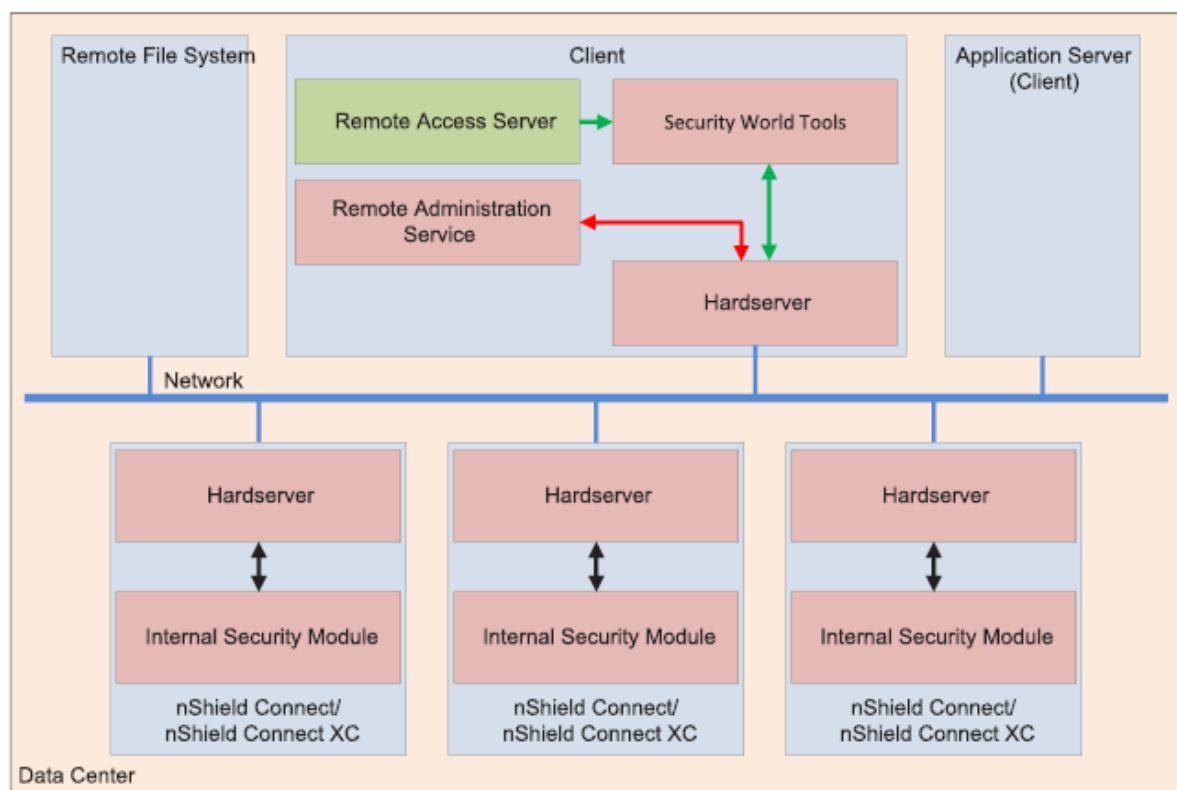
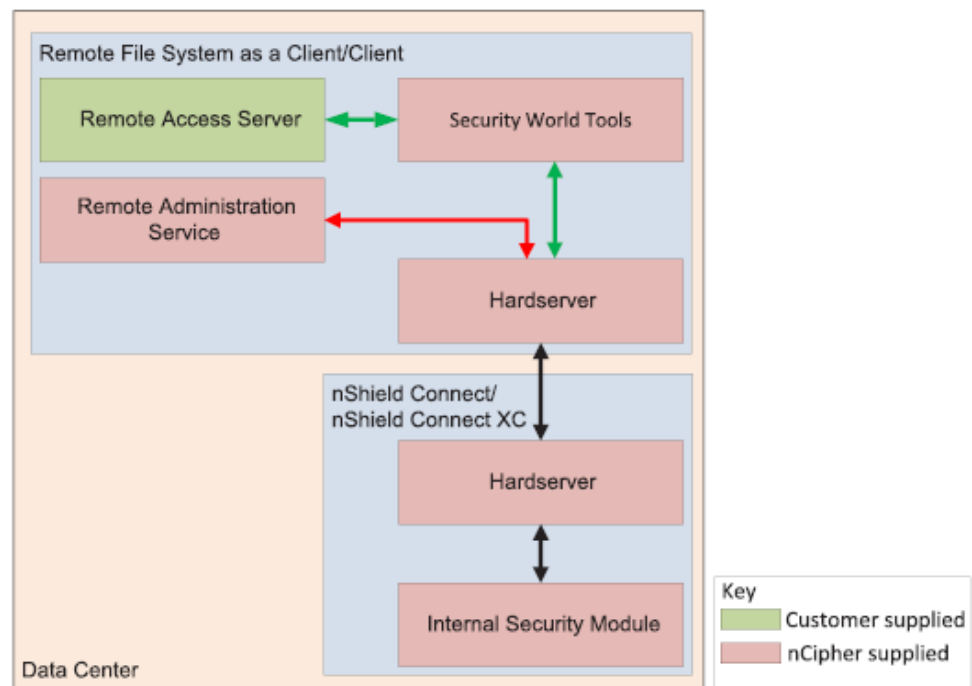
Entrust supply and recommend the use of the nShield Trusted Verification Device (TVD). This is an intelligent smart card reader that blocks any malware on the client machine from spoofing the HSM identity passed to the nShield Remote Administration smart card. The TVD allows the card holder to securely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

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



7.6. Software installation

7.6.1. The Remote Administration Service with the Connect or Connect XC



The Remote Access Server can be on a different client to the one where

the Remote Administration Service is installed.

To use Remote Administration with an Connect or Connect XC, the Remote Administration Service must be installed on a client, which may also be the RFS.

The Remote Administration Service does not require a dedicated server.

A privileged connection is required to carry out privileged operations, such as, for example, changing the mode of the Connect or Connect XC.

7.6.2. Remote Administration Service bundle

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

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

7.6.3. Remote Administration Client

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

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

7.6.4. TVD

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

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

7.7. System configuration

7.7.1. Remote Administration Service port

The port used by Remote Administration Clients to access the Remote Administration Service can be changed by setting the `port` field in the `remote_administration_service_s-`

lot_server_startup section of the hardserver configuration file, see [\[remote_administration_service_startup\]](#).

For more information on the Remote Administration port, see [remote_administration_s-lot_server_startup](#).

7.7.2. Stopping and restarting the Remote Administration Service

The Remote Administration Service can be stopped and started using a command window or the Window Control Panel, see [Stopping and restarting the hardserver](#).

7.7.3. Firewall settings

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

7.8. Configuring auto push

The auto push feature allows future HSM configuration to be performed remotely (that is, without access to the front panel of the HSM).



Enabling the auto push feature allows any client computer to change the HSM configuration file and gives the client the power to make configuration changes that are otherwise only available through the HSM secure user interface.



The auto push feature must be enabled if you want to use NTP time synchronisation on the HSM, see [Configuring NTP in the nShield Connect](#).

To enable auto push, on the Connect display, use the right-hand navigation button to select **System > System configuration > Config file options > Setup auto push > Auto push mode**, set **ON** or **OFF**, then select **CONFIRM**. A confirmation message will be displayed.

Once auto push has been enabled, you must specify the IP address of the client from which to push the configuration from. On the Connect display, use the right-hand navigation button to select **System > System configuration > Config file options > Setup auto push > Push address**. Enter the IP address and select **CONFIRM**. A message will be displayed confirming your chosen IP address. Select **CONTINUE**.

Once auto push is enabled, you can complete the configuration steps by editing the configuration files, rather than by using the front panel of the Connect. See [Configuring the Con-](#)

[nect with configuration files](#) for more about configuration files.



If you do not want to enable auto push, you can fetch the updated configuration file manually from the HSM, select **System > System configuration > Config file options > Fetch configuration**. An SEE machine can not be installed or configured using the fetch configuration option from the front panel. The auto push feature must be used for this.

7.9. Configuring Dynamic slots

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



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

1. Do one of the following:
 - a. Use the **dynamic_slots** section in the module configuration file to define the number of Dynamic Slots for an HSM and push the updated configuration file to the Connect.
See [nShield Connect and client configuration files](#) for more about module configuration files, [dynamic_slots](#) for more about the **dynamic_slots** section and [Configuring the Connect with configuration files](#) for more about editing the module configuration file.
 - or:
 - b. Use the front panel controls to navigate to **Security World mgmt > Set up dynamic slots > Dynamic Slots** and follow the instructions on the screen.
2. Clear the HSM for the changes to take effect.

For example, run the **nopclearfail** command:

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

You can check that the HSM has Dynamic Slots by:

- Running the command:

```
slotinfo -m 1
```

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

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

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



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

or:

- Using the front panel controls to navigate to **Security World mgmt > Display World**.

7.10. Privileged client

Some Connect administrative tasks can only be carried out from clients that have a privileged connection. There are two aspects to configuring a client with a privileged connection:

- Configuring the Connect to accept privileged connections from a particular client, see [Configuring the Connect to use the client](#)
- Enrolling the client as privileged, see [Configuring the Connect to use the client](#) and *Configuring client computers to use the Connect in the Connect Installation Guide*.

7.11. Using nethsmadmin to reboot an Connect

The Connect can be rebooted using the **nethsmadmin** command-line utility. Run the command:

```
nethsmadmin --module=<MODULE> --reboot
```

In this command:

--module=<MODULE>

Specifies the HSM to use, by its **ModuleID** (default = 1).

7.11.1. Enabling and disabling remote reboot

You can enable or disable the ability to remotely reboot an Connect with the `nethsmadmin` utility using:

- The module configuration file
- The front panel of the Connect.

7.11.2. Enabling and disabling remote reboot using the module configuration file

See [Configuring the Connect with configuration files](#) for more about editing the module configuration file.

Do the following:

- To enable remote reboot, locate the `server_settings` section of the module configuration file, and set the `enable_remote_reboot` field to `yes`.
- To disable remote reboot, set the `enable_remote_reboot` field to `no`. See [server_settings](#) for more about the `enable_remote_reboot` field.

7.11.3. Enabling and disabling remote reboot using the front panel of the Connect

Do the following:

- To enable remote reboot, navigate to **System > System configuration > Remote config options > Remote Reboot** and set to `on`.
- To disable remote reboot, set **Remote Reboot** to `off`.

Once you have enabled remote reboot, you can reboot an Connect from a computer using the `nethsmadmin` command, without accessing the unit itself.

7.11.4. Enabling and disabling remote mode change

You can enable or disable the ability to make remote mode changes with the `nopclearfail` utility using:

- The module configuration file
- The front panel of the Connect

7.11.4.1. Enabling and disabling remote mode changes using the module configuration file

See [Configuring the Connect with configuration files](#) for more about editing the module configuration file.

Do the following:

- To enable mode change using `nopclearfail`, locate the `server_settings` section of the module configuration file, and set the `enable_remote_mode` field to `yes`.
- To disable mode change using `nopclearfail1`, set the `Enable_remote_mode` field to `no`. See [server_settings](#) for more about the `enable_remote_mode` field.

7.11.4.2. Enabling and disabling remote mode changes using the front panel of the Connect

Do the following:

- To enable remote mode change, navigate to **System > System configuration > Remote config options > Remote mode changes** and set to `on`.
- To disable remote mode change, set **Remote mode changes** to `off`.

Once you have enabled remote mode change, you can change the mode of an Connect from a computer using the `nopclearfail` command, without accessing the unit itself.

7.11.5. Enabling and disabling remote upgrade

You can enable or disable the ability to remotely upgrade an Connect with the `nethsmadmin` utility using:

- The module configuration file
- The front panel of the Connect

7.11.5.1. Enabling and disabling remote upgrade using the module configuration file

See [Configuring the Connect with configuration files](#) for more about editing the module configuration file.

Do the following:

- To enable remote upgrade, locate the `server_settings` section of the module configuration file, and set the `enable_remote_upgrade` field to `yes`.
- To disable remote upgrade, set the `enable_remote_upgrade` field to `no`. See `server_settings` for more about the `enable_remote_upgrade` field.

7.11.5.2. Enabling and disabling remote upgrade using the front panel of the Connect

Do the following:

- To enable remote upgrade, navigate to **System > System configuration > Remote configuration options > Remote Upgrade** and set to **on**.
- To disable remote upgrade, set **Remote Upgrade** to **off**.

Once you have enabled remote upgrade, you can upgrade an Connect from a computer using the `nethsmadmin` command, without accessing the unit itself.

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

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

Do the following:

- Use The `dynamic_slot_timeouts` section in the module configuration file to define the round trip (HSM to smartcard and back) time limit (the default is 10 seconds), and the card removal detection timeout (the default is 30 seconds).
- Push the updated configuration file to the Connect. See [dynamic_slot_timeouts](#) and [Configuring the Connect with configuration files](#) for more information.

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

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

1. Do one of the following:
 - a. Use the `slot_mapping` section in the module configuration file to define a Dynamic Slot to exchange with slot 0 for an HSM and push the updated configuration file to the Connect.
See [nShield Connect and client configuration files](#) for more about module configuration file, [slot_mapping](#) for more about the `slot_mapping` section and [About user privileges](#) for more about editing the module configuration file.

Or:

- b. Use the front panel controls to navigate to **Security World mgmt > Set up dynamic slots > Slot mapping** and follow the instructions on the screen. You can check the mapping by:

- Running the command:

```
slotinfo -m 1
```

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

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

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

or:

- Using the front panel controls to navigate to **Security World mgmt > Display World**.

7.14. Authorized Card List

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

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

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

The Authorized Card List is a text file `/opt/nfast/kmdata/config/cardlist` on the RFS and each client computer. The file is read from the RFS by associated Connects as and when required for front panel operations. The list applies to all Connects associated with the RFS, regardless of the Security World to which a Connect may belong, including when creating a Security World from the front panel. For client initiated card operations the Authorized

Card List file on that client computer is used. The RFS and client copies of the Authorized Card List have to be kept in step manually.

7.14.1. Adding cards to the Authorized Card List

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



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

7.15. Using Remote Administration

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

- Enable and disable remote mode of an Connect using `nopclearfail -0/-I`, see [Changing the mode](#)
- Reboot an Connect using the `nethsmadmin` utility, `nethsmadmin --module=[MODULE] --reboot`, see [Using nethsmadmin to reboot an Connect](#)
- Upgrade Connect firmware using the `nethsmadmin` utility, `nethsmadmin --module=MODULE—upgrade-image=image_name`, see [Upgrading the nShield Connection image file and firmware from a privileged client](#)
- Synchronise an Connect's Security World data using the `nethsmadmin` utility, `nethsmadmin --module=[ModuleID] --update-world`, see [Creating a Security World using new-world](#)
- Check if Security World files have been copied to an Connect using the `nethsmadmin` utility, `nethsmadmin --module=[ModuleID] --check-world`, see [Creating a Security World using new-world](#)

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

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

remote card presentation.

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

7.15.2. Configuring the Connect with configuration files

You normally configure the Connect using the front panel controls. However, you can choose to export a configuration file, edit it, and then re-import it. You may prefer this approach, for example, if you are importing a number of clients.

There are two ways to configure the Connect by importing edited configuration files.

If you need to configure the Connect regularly, but do not require physical access to it every time you configure it, follow these steps:

1. Ensure that the Connect is configured to accept a configuration from a remote computer, see [Configuring auto push](#).
2. Create a copy of the configuration file from the remote file system in the `/opt/nfast/kmdata/hsm-ESN/config/` directory on the remote computer.
3. Change the name of the copied configuration file to `config.new`.
4. Edit the `config.new` file so that it contains the required configuration. For information about the contents of the Connect configuration file, see [nShield Connect and client configuration files](#).
5. Run the `cfg-pushnethsm` utility on the updated configuration file, specifying the `config.new` file and the IP address of the Connect to load the new configuration. To do this, use a command similar to the following:

```
cfg-pushnethsm --address= <module_IP_address> <config_file>
```

In this command, `<module_IP_address>` is the Connect on which to load the configuration and `<config_file>` is the path to, and name of the updated configuration file.

If `module_IP_address` is an IPv6 address, then enclose it within square brackets, for example `[fc00::1]`.

6. Check that the configuration file on the RFS has been updated with the required changes.



The Connect checks that it is able to update the configuration file on the RFS before applying changes to its own configuration. If you want the Connect to apply the configuration changes even if it can

not update the configuration file on the RFS, for example, because the RFS is going to be moved to a new IP address, use the `--no-rfs-check` command line option:

```
cfg-pushnethsm --address= <module_IP_address> <config_file> --no-rfs-check
```

If `module_IP_address` is an IPv6 address, then enclose it within square brackets, for example `[fc00::1]`.

You will have to copy the configuration file to the RFS manually.

To permit Connect configuration remotely from the RFS computer without enabling the auto push option:

1. Save the file `/opt/nfast/kmdata/hsm-ESN/config` as `config.new` in the same directory and edit it to contain the configuration you require. For more information about the contents of the configuration file, see HSM and client configuration files.
2. On the Connect, from the main menu select **System > System configuration > Config file options > Fetch configuration**.



An SEE machine cannot be installed or configured using the fetch configuration option from the front panel. The auto push feature must be used for this. See [Remotely loading and updating SEE machines](#) for more information.

Whichever method you use, the updated configuration file becomes the new Connect configuration. It is automatically copied back to the file `/opt/nfast/kmdata/hsm-ESN/config/` on the remote file system.

7.15.3. Remote Administration Configuration file sections

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

7.15.3.1. [server_settings]

```
# Is remote mode changing enabled on this system? (default=yes)
# enable_remote_mode=ENUM
#
# Is remote reboot enabled on this system? (default=yes)
# enable_remote_reboot=ENUM
#
# Is remote upgrade enabled on this system? (default=yes)
```

```
# enable_remote_upgrade=ENUM
```

7.15.3.2. [dynamic_slot_timeouts]

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



The `dynamic_slot_timeout` section is in the Module configuration file for Connect HSMs .

7.15.3.3. [dynamic_slots]

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



The `dynamic_slots` section is in the Module configuration file for Connect HSMs .

7.15.3.4. [slot_mapping]

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



The `slot_mapping` section is in the Module configuration file for Connect

HSMs.



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

7.15.3.5. [remote_administration_service_startup]

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

7.15.3.6. [ui_lockout]

```
# Start of the ui_lockout section
# UI lockout settings
# Each entry has the following fields:
#
# Set to "locked" to enable UI lockout without requiring a logical token.
Set
# to "locked_lt" to enable UI lockout with a logical token (requires a valid
# ltui_hash to be set) or "unlocked" for no UI lockout (default=unlocked).
# lockout_mode=ENUM
#
# The hash of the logical token (LTUI) required to authorise access to the
# unit menu structure when the lockout_mode is set to locked_lt; if the
# lockout_mode is locked_lt and a valid hash is provided then the lockout will
# be enabled.
Default is all-zero (disabled).
# ltui_hash=HASH
#
# Set to "no" to disable the front panel power switch in Operational mode.
# (default=yes, power switch causes shutdown)
# panel_poweroff=ENUM
```



The ui_lockout section is relevant for Connect only. It does not apply to nShield Solo, nShield Solo XC or nShield Edge HSMs.

To update an existing hardserver configuration file, edit and insert the sections above. Alteratively factory resetting an Connect will generate a new configuration file including the new Remote Administration relevant sections listed above. See the appropriate User Guide for more information on editing and loading configuration files.

8. Creating and managing a Security World

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

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

- The Installation Guide for more about installing the module and software.
- [Client Software and module configuration](#)

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



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

For more information about the type of user that is required for different operations, see [About user privileges](#).



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

8.1. Creating a Security World

You can use the following to create Security Worlds:

- The unit front panel controls
See [Creating a Security World using the Connect front panel](#).
- The `new-world` command line utility
See [Creating a Security World using new-world](#).

8.1.1. The creation process

When you create a Security World:

- The HSM is erased

- A new HSM key for this Security World is generated
- A new ACS to protect this HSM key is created
- The Security World information is stored within the file system of the Connect operating system and on the RFS
 - The information is encrypted using the secrets stored on the ACS
- The HSM and Security World are configured for Audit Logging if selected



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



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



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

8.1.2. Security World Files

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

For more information about the remote file system, see:

- [Remote file system \(RFS\)](#)
- [Configuring the remote file system \(RFS\)](#)

Other Connect modules can also use a Security World created on an Connect using client cooperation. For more information, see [Setting up client cooperation](#).

8.1.2.1. Location of Security World files

If a Security World operation is done on the module, the files are created or updated on the module and automatically copied to the directory specified by the environment variable

NFAST_KMLOCAL on the remote file system. By default, this is **/opt/nfast/kmdata/local**.

If the Security World operation is done on a client machine (for example, Operator Card Set and key operations), files are created or updated in the or the directory specified by the **NFAST_KMLOCAL** environment variable, on the client machine only.

The **/opt/nfast/kmdata/local** directory on the remote file system is updated automatically when an operation is done on the module.

If you want to make cards or keys which are normally created from the client available from the module's front panel, we recommend that you use client co-operation to automate the copying of files to the module. For information about configuring client co-operation, see [Setting up client cooperation](#).

If you do not use client cooperation, you must manually copy the appropriate card and key files from the client or host on which the card set or key was created to the **/opt/nfast/kmdata/local**'s remote file system. These files must then be updated on the module by selecting **Security World mgmt > RFS operations > Update World files** from the main menu.

To be able to create Operator Cards or keys, the user on the client must have write permission for this directory. All other valid users must have read permission.



By default, the Key Management Data directory, and sub-directories, inherit permissions from the user that creates them. Installation of the Security World Software must be performed by a user with Administrator rights that allow read and write operations, and the starting and stop ping of applications.

8.1.2.2. Files and operations

Security World operations create or modify files in the **/opt/nfast/kmdata/local** directory as follows:

Operation	creates/modifies	the file(s) ...
Create a Security World	creates	world (for each module in the Security World) module_ESN
Load a Security World	creates or modifies	(for each module in the Security World) module_ESN
Replace an ACS	modifies	world

Operation	creates/modifies	the file(s) ...
Create an OCS	creates	card_HASH cards_HASH_NUMBER
Create a softcard	creates	softcard_HASH
Generate a key	creates	key_APPNAME__IDENT
Recover a key	modifies	key_APPNAME (for each key that has been recovered)

In this table:

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

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

8.1.2.3. Required files

The following files must be present and up to date in the %NFAST_KMDATA%\local directory, or the directory specified by the **NFAST_KMLOCAL** environment variable, for a client to use a Security World:

- **world**
- A **module_ESN** file for each module that this host uses
- A **cards_<IDENT>** file for each card set that is to be loaded from this host
- A **card_<IDENT>_NUMBER** file for each card in each card set that is to be loaded from this host
- A **key_<APPNAME>_<IDENT>** file for each key that is to be loaded from this host.

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

8.1.3. Security World options

Decide what kind of Security World you need before you create it. Depending on the kind

of Security World you need, you can choose different options at the time of creation. For convenience, Security World options can be divided into the following groups:

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

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

8.1.3.1. Security World basic options

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

8.1.3.1.1. Cipher suite

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

8.1.3.1.2. ACS quorum

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



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



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

In many cases, it is desirable to make K greater than half the value of N (for example, if N is 7, to make K 4 or more). Such a policy makes it harder for a potential attacker to obtain

enough cards to access the Security World. Choose values of K and N that are appropriate to your situation.

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

8.1.3.1.3. FIPS 140-2 Level 3 compliance

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



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

If you enable compliance with FIPS 140-2 Level 3 roles and services, authorization is required for the following actions:

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

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

8.1.3.1.4. UseStrongPrimes Security World setting

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

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

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

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

8.1.3.1.5. Remote Operator

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

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

8.1.3.2. OCS and softcard replacement

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



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

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



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

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

8.1.3.3. Pass phrase replacement

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

However, you can choose to enable pass phrase replacement at the time you create a Secu

rity World. This option makes it possible to replace the pass phrase of a card or softcard even if you do not know the existing pass phrase. Performing such an operation requires authorization from the Security World's ACS.

For details about performing pass phrase replacement operations, see [Changing unknown or lost pass phrase](#).

8.1.3.4. Nonvolatile memory (NVRAM) options

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



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

8.1.3.5. Security World SEE options

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

8.1.3.5.1. SEE debugging

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

8.1.3.5.2. Real-time clock (RTC) options

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

8.1.3.6. Security World replacement options

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

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

8.1.4. Creating a Security World using the Connect front panel



When initiated from the Connect front panel, while a Security World is being created the Connect disconnects itself from the network to ensure that the operation is not interrupted. This means that the Remote Administration feature cannot be used to present cards from a remote location when creating a Security World from the front panel.

8.1.4.1. Before you start

Before you start to create a Security World:

- The directory `/opt/nfast/kmdata/local` on the remote file system must exist and be empty.
- Before configuring the Security World, you should know:
 - The security policy for the HSM
 - The number and quorum of Administrator Cards and Operator Cards to be used.

To help you decide on the Security World you require, see [Security World options](#).

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

To create a Security World from the Connect Front Panel:

1. From the main menu, select **Security World mgmt > Module initialization > New Security World**.
2. Specify the Security World mode:
 - a. **FIPS 140-2 Level 3** creates a Security World compliant with FIPS 140-2 requirements for roles and services at Level 3.

- b. **Common Criteria CMTS** creates a Security World supporting Common Criteria Protection Profile EN 419 221-5.
 - c. **Unrestricted** creates a Security World which doesn't impose any particular conformance. With appropriate environmental constraints, an unrestricted Security World can be compliant with FIPS 140-2 Level 2.
3. Select the Cipher suite for the Security World. Currently only one option is available for the Security World key, AES (SP800-131AR1).
4. Enter the default quorum for the ACS. This consists of:
 - a. The maximum number of cards from the ACS required by default for an operation. This number must be less than or equal to the total number of cards in the set.
 - b. The total number of cards to be used in the ACS. This must be a value in the range 1 – 64 except for the Common Criteria CMTS Security World mode, for which the range is 2 – 64.



We recommend that you do not create an ACS for which the required number of cards is equal to the total number of cards because you cannot replace such an ACS if even a single card is lost or damaged.

5. If you answer the question **Specify all quorums?** by selecting:
 - a. **no** - all operations and features (with the exception of pass phrase recovery) will be enabled and require the maximum number of cards
 - b. **yes** - you can specify which operations and features you want to enable (including pass phrase recovery) and what the required number of cards for each of these will be.
6. If you chose to disable individual features or require a lower number of cards required for an operation, specify these parameters now. You can select a different number of Administrator Cards (K) to be required for each operation. You can also disable recovery and replacement operations and choose to use KNSO to authorize SEE (Secure Execution Engine) operations. The options for which you can specify a separate value of K are as follows:

Operation	Action allowed on HSM
Module reprogramming	Initializing an HSM into a Security World. You must specify a value of K for this operation.
Pass phrase replacement	Replacement of pass phrases from backup files when recovering an OCS. You can disable this operation, see Pass phrase replacement . This operation is disabled in Common Criteria CMTS mode and cannot be enabled.

Operation	Action allowed on HSM
OCS/softcard replacement	Recovery of keys from backup files when replacing an OCS. You can disable this operation if you are using the Connect, see OCS and softcard replacement .
NVRAM access	Reading from and writing to the NVRAM. You can choose to authorize this operation with K_{NSO} , see Nonvolatile memory (NVRAM) options .
RTC access	Updating the real time clock. You can choose to authorize this operation with K_{NSO} , see Real-time clock (RTC) options .
SEE debugging	Viewing full SEE debug information. You can specify a value of K for this operation, all it for all users or authorize it with K_{NSO} , see SEE debugging . This operation is disabled in Common Criteria CMTS mode.
FTO	Use of an Foreign Token Open (FTO) Delegate Key (ISO Smart Card Support). You can specify a value of K for this operation or authorize it with K_{NSO} . This operation is disabled in Common Criteria CMTS mode.

7. Specify if audit logging should be enabled.



In Common Criteria CMTS mode, audit logging is automatically enabled and cannot be disabled.

8. Specify whether the HSM is a valid target for remote shares (that is, whether it can import slots), see [Remote Operator](#). This option is disabled for Common Criteria CMTS mode.
9. For Common Criteria CMTS mode only, choose whether to specify the maximum number of times an Assigned key can be used since it was authorized. A use limit compatible with the specified maximum will be imposed at key creation time and can be verified for Assigned keys. If you choose to specify a maximum key usage limit:
 - a. Enter the key usages allowed, up to a maximum of 9999.
10. For Common Criteria CMTS mode only, choose whether to specify a maximum timeout for Assigned keys since key authorization. A time limit compatible with the specified maximum will be imposed when the key is created, and can be verified for Assigned keys. If you choose to specify a key timeout:
 - a. Select the units from **Seconds**, **Minutes**, **Hours**, or **Days**.
 - b. Enter a value up to a maximum of 9999 in your selected unit.
11. Format a card for the ACS as follows:
 - a. Insert a card for the ACS and confirm that you want to use it.

- b. If the card is not blank, choose whether to overwrite it or to use a different card.
 - c. Choose whether to specify a pass phrase for the card. If you choose to specify a pass phrase:
 - i. Enter the pass phrase.
 - ii. Enter the pass phrase again to confirm it. If the two pass phrases do not match, you must enter the correct pass phrase twice.
 - d. When prompted, remove the card.
12. Repeat the previous step to format additional cards for the ACS, setting their pass phrases as described, until the ACS is complete. Each prompt screen shows how many cards are required and how many have been used.
13. At completion, a message confirms that the Security World has been created.

8.1.5. Creating a Security World using new-world

8.1.5.1. Before you start

Before you start to create a Security World:

- The HSM must be in pre-initialization mode. See [Checking and changing the mode on an nShield Connect](#) for more about changing the mode.
- You must be logged in to the computer that is running the RFS. The RFS should be a privileged client that has the client tools installed. For more information, see [server_settings](#).
- If you have installed the Security World Software in a directory other than `/opt/nfast/`, you must have created a symbolic link from `/opt/nfast/` to the directory in which the software is actually installed.
- Before configuring the Security World, you should know:
 - The security policy for the HSM
 - The number and quorum of Administrator Cards and Operator Cards to be used

To help you decide on the Security World you require, see [Security World options](#).

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

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

8.1.5.2. Using nethsmadmin to copy a Security World to an Connect and check the current version

If a Security World is created using new-world, the **nethsmadmin** command-line utility enables you to copy the resultant files to a Connect. Run the command:

```
nethsmadmin --module=<MODULE> --update-world
```

nethsmadmin can also be used to check if the Security World files have been copied to the Connect. Run the command:

```
nethsmadmin --module=<MODULE> --check-world
```

In these commands:

--module=<MODULE>

Specifies the HSM to use, by its **ModuleID** (default = 1).

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


8.1.5.3. Running the new-world command-line utility




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



The example below will create a Security World supporting FIPS140-2 Level 3 with a ACS quorum of 3/5 and with audit logging enabled.



```
new-world --mode=fips-140-2-level-3 --acs-quorum=3/5 --audit-logging
```

In this command:

Option	Description
<code>--initialize</code>	<p>This option tells <code>new-world</code> to create a new Security World, replacing any existing <code>/opt/nfast/kmdata/local/</code> directory.</p> <div>  <p>Replacing an existing Security World in this way does not delete the Security World's host data and recovery and replacement data, but renames the existing <code>/opt/nfast/kmdata/local/</code> directory in which these reside as <code>/opt/nfast/kmdata/localN</code> (where <i>N</i> is an integer assigned depending on how many Security Worlds have been previously saved during overwrites).</p> </div>
<code>--factory</code>	This option tells <code>new-world</code> to erase an HSM, restoring it to factory state.
<code>--no-remoteshare-cert</code>	This option tells <code>new-world</code> not to make the HSM a target for remote shares.
<code>--no-strict-rsa-keygen</code>	If you have not specified a mode parameter you can use the <code>-no-strict-rsa-keygen</code> flag to disable the <code>UseStrongPrimes</code> setting. Otherwise it will be enabled by default. See UseStrongPrimes Security World setting .
<code>--mode=MODE</code>	<p><code>FIPS-140-2-level-3</code> creates a Security World compliant with FIPS 140-2 Level 3.</p> <p><code>common-criteria-cmts</code> creates a Security World supporting <i>Common Criteria PP 419 221-5</i>.</p> <p>Omitting this option will create a default Security World compliant with FIPS 140-2 Level 2.</p>

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

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

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



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

8.1.5.4. new-world command-line utility features

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

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



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



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



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

The following feature names are available:

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

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

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

- `nvram`
- `rtc`
- `fto`

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

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



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



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



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

For example, the following features:

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

Create a Security World for which:

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

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

8.1.5.5. new-world command-line utility output

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

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

If the HSM is not in the pre-initialization mode, `new-world` advises you that you must put the HSM in this mode and waits until you have changed the HSM mode before continuing. See [Checking and changing the mode on an nShield Connect](#) for more about changing the mode.



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

8.1.6. After you have created a Security World

Store the ACS in a safe place.



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

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



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

The Security World data is stored on the HSM and on the RFS. For more information, see [Security World Files](#).

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

8.2. Displaying information about your Security World

To display information about the status of your Security World:

- Select **Security World mgmt** > **Display World info** from the main menu
- Run the `nfkminfo` command-line utility. See [Displaying information about a Security World with nfkminfo](#).
- Run the `kmfile-dump` command-line utility. See [Displaying information about a Security World with kmfile-dump](#).
- Run the `nethsmadmin` command-line utility. See [Using nethsmadmin to copy a Security World to an Connect and check the current version](#).

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

8.2.1. Displaying information about a Security World with nfkminfo

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

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

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

Optionally, the command can also include the following:

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

To output a detailed list of Security World information, run `nfkminfo` with the `-w|--world-info` option (with or without the other options). For a description of the fields in this list, and more information about using `nfkminfo`, see [nfkminfo: information utility](#).

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

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

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

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

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

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

8.3. Adding or restoring an HSM to the Security World

When you have created your Security World, you can add additional HSMs to it. You can restore HSMs that have previously been removed from the same Security World in the same way.

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

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



The additional HSMs can be any nShield HSMs.

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

- Have installed the additional HSM hardware, as described in the *Installation Guide*.
- Have a copy of the Security World data on the HSM's remote file system in the Key Management Data directory.
- Possess a sufficient number of cards from the ACS and the appropriate pass phrases.

Adding or restoring an HSM to a Security World:

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

After adding an HSM to a Security World, you cannot access any keys that were protected by a previous Security World that contained that HSM.



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

8.3.1. Adding an HSM to a Security World using the Connect front panel

To add an HSM to a Security World:

1. If the HSM already belongs to a Security World, erase it from the Security World to which it belongs, as described in [Erasing a module from a Security World](#).
2. From the main menu, select **Security World mgmt > Module initialization > Load Security World**.
3. Specify whether the HSM can use the Remote Operator feature import slots. For more information, see [Remote Operator](#).
4. At the prompt, insert an Administrator Card, and enter its pass phrase if required.
5. Continue to insert Administrator Cards when prompted until you have inserted the number required to authorize HSM reprogramming.

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

1. Open a command window and type the command:

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

In this command:

- **-l|--program**

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

- **-S|--no-remoteshare-cert**

These options tell **new-world** not to make the HSM a target for remote shares.

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

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

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

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

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

If the HSM is not in the pre-initialization state, `new-world` displays an error and exits. See [Checking and changing the mode on an nShield Connect](#) for more about changing the mode.

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

2. After `new-world` has reprogrammed the HSM, restart the HSM in the operational state. See [Checking and changing the mode on an nShield Connect](#) for more about changing the mode.
3. Store the ACS in a safe place.



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

8.4. Security World migration

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

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

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



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



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

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

8.4.1. Pre-requisites for migrating keys

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

- Two HSMs. These can be any of the hardware variants Solo+, Solo-XC, Connect+, Connect-XC and do not need to be of the same type.
- A quorum of ACS cards for the source World.
- A quorum of ACS cards for the destination World.
- Sufficient blank cards to create new OCS cards for any keys that are OCS protected.
- Remote mode switching must be enabled on both HSMs used for the migration. For instructions, see [Enabling and disabling remote mode change](#).

8.4.2. Restrictions on migrating keys

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

- The source module must be running firmware version 12.50 or later.
- The destination module must be running firmware version 12.50 or later.
- Only recoverable keys can be migrated. If your source keys are non-recoverable, you cannot use the migration utility to migrate keys.
- It is not possible to alter a key's protection or change the quorum of any created OCS during the migration.
- Replacement cards should be of the same or newer generation than the cards that they replace.
- The source and destination modules must both have KLF2 warrants. If one or both of the modules are Solo XC and have a KLF warrant you should request an upgrade to a KLF2 warrant before starting migration, see the *Warrant Management* section in the *Solo User Guide*.
- The operator running the migrate-world utility must have the access rights to create a privileged connection to the hardserver.
- The migration tool must have exclusive use of the modules during migration. Do not use them for any other purpose during migration and if either module is a Connect do

not enter anything via the front panel during migration.



If the destination World is **fips-140-2-level-3**, then some keys that were usable in the source World may not be usable in the destination World due to those algorithms or key lengths being restricted. The migration tool might not be able to successfully migrate these keys so they should be removed from the source World before attempting the migration. Any keys of this type that do migrate successfully will be restricted at the point of use.



If the destination World is **fips-140-2-level-3** or **common-criteria-cmts** the migration tool will automatically remove ExportAsPlain from the ACL of any migrated key during the migration process.



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

8.4.3. About the migration utility

You can run the migration utility in the following modes:

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

8.4.3.1. Usage and options

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

Option	Enables you to...
-k <KEYS> --keys-at-once=<KEYS>	Migrate no more than this number of keys per ACS loading. This is useful to prevent ACS time-outs if you have a large number of keys to migrate. (0=unlimited, default=0). It is recommended to limit the number of keys to be migrated at any one time to no more than 100.
-h, --help	Obtain information about the options you can use with the utility.

Option	Enables you to...
<code>-c <CARDSETS></code> <code>--cardsets-at-once=<CARDSETS></code>	Migrate keys protected by this number of card sets or softcards per ACS loading. This is useful to prevent ACS time-outs if you have a large number of different card sets or softcards to migrate. (0=unlimited, default=0).
<code>--version</code>	View the version number of the utility.
<code>--src-warrant=<src-warrantfile></code>	Specify the location of the warrant file of the source module.
<code>--src-module=<MODULE></code>	Specify which module ID to use as the source module.
<code>--source=<SOURCE></code>	Specify the path to the folder that contains the source World data.
<code>--plan</code>	View the list of steps that will be carried out.
<code>--perform</code>	Migrate keys interactively.
<code>--dst-warrant=<dst-warrantfile></code>	Specify the location of the warrant file of the destination module.
<code>--dst-module=<ModuleId></code>	Specify which module ID to use as the destination module.
<code>--debug</code>	Outputs debug messages and stack traces in case of errors. It is recommended to use this only for testing as it will slow down operation and make card timeouts more likely to occur. A large volume of output is produced for each key that is migrated, so it is recommended to migrate a single key at a time when using this option.
<code>--key-logging</code>	This option will enable key usage logging on all migrated keys. If the destination World does not support audit logging the keys will still be migrated but LogKeyUsage logging will not be set in the ACL of the migrated keys.



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

8.4.4. Migrating keys

8.4.4.1. Preparing for migration

Before you begin:

- Install the latest version of the Security World Software from the installation media. See the *Installation Guide* for more information.
- Ensure that the warrant files for the source and destination modules are stored in their

default locations. For Solo+, or SoloXC modules, the default location is **NFAST_KM-DATA/warrants/**. For Connect+, or ConnectXC modules, the default location is **NFAST_KMDATA/hsm-<ESN>/warrants/**. If the warrant files are not at the default location, the `--src-warrant` and `--dst-warrant` parameters need to be specified in the `migrate-world` command.

- Copy the source World data to a location defined by the `--source=<SOURCE>` parameter of the migration tool.
- If the destination World does not exist already, create a new destination World. For instructions, see [Creating a Security World](#)



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

8.4.5. Migrating keys process



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



If the destination World supports audit logging you can choose whether the migrated keys will have key usage logging enabled or not by use of the `--key-logging` command line switch. If you only wish key usage logging to be enabled on a subset of the keys then you must separate the source keys into two groups and run the `migrate-world` command separately for each group.

To migrate keys to the destination World:

1. Run the migration utility in the perform mode with the required options. For information about the usage and options you can use, see [About the migration utility](#).
2. Ensure that the data for the destination World is in the standard location for World data, derived from one of the following:
 - Either the environment variable **NFAST_KMLOCAL** or **NFAST_KMDATA**.
 - The default directory: **/opt/nfast/kmdata/local/**.
3. If the module is not configured to use the destination World, the utility prompts you to program the module and supply the ACS of the destination World.
4. The utility guides you through specific steps, prompting you to supply the required

card sets and pass phrases.

5. At the end of the migration both the source and destination modules are cleared. If you wish to use the modules then you must reload them with an appropriate Security World.



The utility will attempt to automatically change the module mode when needed. Should the automatic change of mode fail for any reason, then the utility will prompt you to change the module state to either initialization or operational at various points during the procedure. See [Checking and changing the mode on an nShield Connect](#) for more about changing the mode.

8.4.6. Verifying the integrity of the migrated keys

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

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

Example:

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



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



Note that if the destination World is `fips-140-2-level-3` then some keys that were usable in the source World may not be usable in the destination World due to those algorithms or key lengths being restricted. The migration tool will successfully migrate the keys but they will be restricted at the point of use.

8.4.7. Troubleshooting



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

Error	Explanation	Action
There are no keys requiring migration.	Any migrate-able keys found in the source World already exist in the destination World. The migration utility returns this error if: <ul style="list-style-type: none"> • The keys have already been migrated • All keys are non-recoverable and therefore cannot be migrated. 	None.
Source module must be specified. Destination module must be specified. Source and Destination modules must be different. Module is not usable.	This utility requires you to specify both a source and destination module which must be different modules and both must be usable.	Specify the correct modules.
Source World has indistinguishable cardsets or softcards. Destination World has indistinguishable keys.	There are irregularities in one of the Worlds, but these irregularities do not affect the migration process.	None.
Destination World has indistinguishable cardsets or softcards. Source World has indistinguishable keys. Cannot determine protection of keys.	There are problems with one of the Worlds.	Contact Support.
Source World not recoverable.	The source World is not recoverable and the keys therefore cannot be migrated.	If the source World is not recoverable, you cannot use the migration utility to migrate keys. Contact Support.

Error	Explanation	Action
Missing security World at PATH. Source world must be specified.	The path for the source World is wrong. There is no World data at the location that was specified when running the migration utility.	Supply the correct path to the source World. If you have supplied the correct path to the directory that contains the source World data, the migration utility has not found a destination World.
Source World is the same as the destination World.	An incorrect path was supplied for the source World data when running the utility. The destination World data does not exist in the default location defined by the environment variable <code>NFAST_KMLOCAL</code> or <code>NFAST_KM-DATA</code> .	Run the utility with the correct path to the source World data. Move the source World data to a different location and then copy the destination World data to the default location. If the default location is defined by an environment variable, configure the variable to point to the location of the destination World, which then becomes the new default location.
Cannot find <NAME> utility, needed by this utility. <NAME> utility is too old, need at least version <VERSION NUMBER>.	The software installation is partially completed. The path (in the environment variable for the operating system) might be pointing to an old version of the software.	Reinstall the software. Ensure that the path points to the latest version of the software.
nFast error: TimeLimitExceeded; in response to SetKM...	The ACS time-out limit has expired.	Restart the key migration process; see Security World migration .
Destination world does not support audit logging.	You have specified the <code>--key-logging</code> option but the destination world does not support audit logging.	None. The keys will be migrated but LogKeyUsage will not be set in the ACL of migrated keys.
Failed to load warrant file <FILE>.	There is a problem reading the warrant file.	Check that your warrant files are in the correct location and have not been edited in any way.

8.5. Migrating KMDATA

To move KMDATA from the default location of `C:\ProgramData\nCipher:`

1. Open a command prompt window as an administrator.
2. Use `Xcopy` with the following parameters to copy the default folder to a new location:


```
Xcopy C:\ProgramData\nCipher <Destination> /e /v /o /i
```

3. Enter the new location for the following environment variables:
 - a. In the Windows Control Panel, navigate to **Control Panel > System and Security > System > Advanced system settings**.
 - b. In the **Advanced** tab, select **Environment Variables**.
 - c. Update the following system variables:
 - **NFAST_CERTDIR:** <path\to\new\folder>\Feature Certificates
 - **NFAST_KMDATA:** <path\to\new\folder>\Key Management Data
 - **NFAST_LOGDIR:** <path\to\new\folder>\Log Files
 - d. If your Security World client is on or above v12.70.4, add the following environment variable in the same section:
 - **NFAST_KNETIDIR:** <path\to\new\folder>\hardserver.d
4. In the Services tool, restart the nFast Server process.
5. After the service restarts, run the following command to check the migration was successful:

```
anonkneti -m 127.0.0.1
```

6. After confirming that the migration was successful, delete **C:\ProgramData\nCipher**.

8.6. Erasing a module from a Security World

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

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



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

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

8.6.1. Erasing a module from the unit front panel

To erase a module from a Security World, from the main menu, select **Security World mgmt** > **Module initialization** > **Erase Security World**.

When you erase a Security World in this way, the Security World files remain on the remote file system. Delete these files if you wish to remove Security World completely. For more information, see [Security World Files](#).

8.6.2. Erasing a module with new-world

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

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

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

In this command:

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

8.6.2.1. Output

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

8.6.3. Erasing a module with KeySafe

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

1. Start KeySafe. (For an introduction to KeySafe and information on starting the software, see [Using KeySafe](#).)
2. Click the **World** menu button, or select **World** from the **Manage** menu. KeySafe takes you to the **World Operations** panel.
3. Click the **Erase Module** button. KeySafe takes you to the **Erase Module** panel.

4. Select the module that you want to erase by clicking its listing on the **Security world status** tree, then click the **Commit** command button.
5. KeySafe erases all secrets from the module, returning it to its factory state.



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

8.6.4. Erasing a module with initunit

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

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

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

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



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

8.6.4.1. Output

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

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

Otherwise, **initunit** returns an error message.

8.7. Replacing an existing Security World

When you erase a Security World from the module's front panel, all long-term key material is deleted from the module's memory and all Security World data is removed from the mod-

ule's internal file system.

This operation does not remove any files from the remote file system or client machines. You should remove the files manually from the `/opt/nfast/kmdata/local` directory on the remote file system and any client computers to which the Security World was copied.



Any Operator Cards created in a previous Security World cannot be used in the new Security World. If you are replacing a Security World, you must erase all the Operator Cards created in the previous Security World before you create the new Security World. See [Erasing cards and softcards](#).

8.8. Deleting a Security World

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

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



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



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

To delete an existing Security World:

1. Remove all the HSMs from the Security World.
2. Delete the files in the Key Management Data directory.



There may be copies of the Security World data archive saved on your backup media. If you have not reused or destroyed the old

ACS, an attacker in possession of these cards could access your old keys using this backup media.



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

8.8.1. Deleting the Security World using the Connect front panel

When you erase a Security World using the unit front panel, all long-term key material is deleted from the HSM's memory and all Security World data is removed from the HSM's internal file system.

- You will not be able to access any of the keys that you have previously used
- Before you remove an old Security World, you must reformat any smart cards that were used previously as Operator Cards within this Security World.



If you do not reformat the smart cards used as Operator Cards before you reinitialize your HSM, you must throw them away because they can not be used, erased, or reformatted without the old Security World key.

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

To erase a Security World using the front panel of the unit, from the main menu select **Security World mgmt > Module initialization > Erase Security World**.

This operation does not remove any files from the RFS or client machines. You should remove the files manually from the `/opt/nfast/kmdata/local` directory on the RFS and any client computers to which the Security World was copied.

9. Managing card sets and softcards

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

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

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

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

A Security World offers three levels of key protection:

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

For more information about creating a Security World, see [Creating and managing a Security World](#).

For more information about key management, see [Working with keys](#).

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



To perform the tasks described in this chapter, we recommend using the unit front panel or a client on the same computer that contains the RFS. To perform these tasks on a different client, you must transfer the card data to the RFS.



If you are sharing the Security World across several client computers, you must ensure that the changes are propagated to all your computers. One way to achieve this is to use client cooperation. For more information, see [Setting up client cooperation](#).

If you want to use the Remote Operator feature to configure smart cards for use with a remote unit, see [Remote Operator](#).

9.1. Creating Operator Card Sets (OCSs)

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

You can create OCSs that have:

- Names for individual cards, as well as a name for the whole card set
- Specific *K/N* policies
- Optional pass phrases for any card within a given set
- Formal FIPS 140-2 Level 3 compliance.



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

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

Creating (and managing) OCSs can be done with the unit front panel, as described in [Creating an Operator Card Set using the nShield Connect front panel](#). However, you can also use the following tools to create an OCS:

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

9.1.1. Persistent Operator Card Sets

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

use the keys after you have removed the last card, you must make that OCS persistent.

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

For more information about persistent OCSs, see [Using persistent Operator Card Sets](#).

An OCS to be used to authorize login on a unit must be persistent and not loadable remotely. It is recommended that such an OCS is not used to protect sensitive keys.

9.1.2. Time-outs

OCSs can be created with a time-out, so that they can only be used for limited time after the OCS is loaded. An OCS is loaded by most applications at start up or when the user supplies the final required pass phrase. After an OCS has timed out, it is not loadable by another application unless it is removed and reinserted. Time-outs operate independently of OCS persistence.

9.1.3. FIPS 140-2 Level 3-compliant Security Worlds

When you attempt to create an OCS for a Security World that complies with FIPS 140-2 Level 3, you are prompted to insert an Administrator Card or Operator Card from an existing set. You may need to specify to the application the slot you are going to use to insert the card. You need to insert the card only once in a session.

9.1.4. Creating an Operator Card Set using the nShield Connect front panel

To create an OCS, follow these steps:

1. From the main menu, select **Security World mgmt > Cardset operations > Create OCS**.

You are prompted to enter the name of the OCS.

2. Enter a name and press right-hand navigation button.
3. Enter the quorum for the OCS, using the touch wheel to move from one field to the other. The quorum consists of:
 - The maximum number of cards from the OCS required by default for an operation. This number must be less than or equal to the total number of cards in the set.

- The total number of cards to be used in the OCS. This must be a value in the range 1 – 64.
4. Press the right-hand navigation button to move to the next screen.
 5. If you wish to specify a time out for the card set, enter the time out in seconds.
 6. Choose whether to create a persistent card set. You can select:
 - **Not persistent** (which is the default)
 - **Persistent**
 - **Remoteable/Persistent**
 7. Choose whether to name individual cards and enable pass phrase replacement by answering **Yes** or **No** to each question and then pressing the right-hand navigation button.
 8. Insert a smart card to be formatted for the OCS.

If the card is not blank, choose whether to overwrite it or to use a different card. (If the card is an Operator Card from another Security World, you cannot overwrite it and are prompted to enter a different card.)

9. If you have chosen to name individual cards, you are prompted to enter the name for the card.
10. You are asked whether you wish to specify a pass phrase for the card. If you choose **Yes**, you are prompted to enter the pass phrase twice.

While the Operator Card is being created, the screen displays the message **Processing**.

If there are further cards from this OCS to be processed, the screen changes to **Waiting**. Remove the card, and repeat steps 8 through 10 for each of the remaining cards.

When all the cards in the set have been processed, you are told that the card set has been created successfully.



9.1.5. Creating an Operator Card Set using the command line

To create an OCS from the command line:

1. Run the command:

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

This command uses the following options:

Option	Description
<code>-m <MODULE>, --module=<MODULE></code>	This option specifies the number of the hardware security module to be used to create the token. If you only have one hardware security module, <code><MODULE></code> is 1.
<code>-Q --ocs-quorum=<K>/<N></code>	<p>In this option, <code><K></code> is the minimum required number of cards. If you do not specify the value <code><K></code>, the default is 1.</p> <div>  <p>Some applications do not have mechanisms for requesting that cards be inserted. Therefore any OCSs that you create for use with these applications must have <code><K>=1</code>.</p> </div> <p><code><N></code> is the total number of cards. If you do not specify the value <code><N></code>, the default is 1.</p>
<code>-N --name=<NAME></code>	This option specifies a name for the card set. The card set must be named with this option before individual cards can be named using the <code>-M/--name-cards=<NAME></code> options.
<code>-M --name-cards</code>	Specifying this option allows you to name individual cards within the card set. You can only use this option after the card set has been named by using the <code>--name='NAME' option. 'createocs</code> prompts for the names of the cards as they are created. Not all applications can display individual card names.
<code>-p --persist</code>	This option creates a persistent card set.
<code>-P --no-persist</code>	This option creates a non-persistent card set.
<code>-R --no-pp-recovery</code>	This option specifies that pass phrase replacement for this OCS is disabled. Setting this option overrides the default setting, which is that the card pass phrases are replaceable. You can specify the enablement of pass phrase replacement explicitly by setting the <code>--pp-recovery</code> option.
<code>-q --remotely-readable</code>	<p>This option allows this card set to be read remotely. For information on configuring Remote OCSs, see Remote Operator.</p> <div>  <p>Not required for Remote Administration.</p> </div>
<code>-T --timeout=<TIME></code>	This option sets the time-out for the card set. Use the suffix <code>s</code> to specify seconds, <code>m</code> for minutes, <code>h</code> for hours, and <code>d</code> for days. If the time-out is set to 0, the OCS never times out. Otherwise, the hardware security module automatically unloads the OCS when the amount of time specified by <code>TIME</code> has passed since the OCS was loaded.
<code>-e --erase</code>	Specifying this option erases a card (instead of creating a card set). You can specify this option twice in the form <code>-ee</code> to repeatedly erase cards.



With Security World Software v11.72 and later, pass phrases are limited to a maximum length of 254 characters, when using `createocs`.

See [Maximum pass phrase length](#).

If you have created a FIPS 140-2 Level 3 compliant Security World, you must provide authorization to create new Operator Cards; **createocs** prompts you to insert a card that contains this authorization. Insert any card from the Administrator Card Set or any Operator Card from the current Security World.

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

2. Insert the smart card to use.

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

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

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

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

When you insert a valid card, **createocs** prompts you to type a pass phrase.



The nShield PKCS #11 library requires Operator Cards with pass phrases.



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

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

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

If you entered a pass phrase, **createocs** prompts you to confirm it.

4. Type the pass phrase again and press **Enter**.

If the pass phrases do not match, **createocs** prompts you to input and confirm the pass phrase again.

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

9.1.6. Creating an Operator Card Set with KeySafe

KeySafe enables you to create OCSs with:

- Their own names
- *K/N* policies
- Optional pass phrases for any card within the OCS
- Formal FIPS 140-2 Level 3 compliance.

To create an OCS with KeySafe:

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

The **List Operator Card Sets** panel is displayed.

3. Select an HSM within the Security World from the Security World status pane.
4. Click the **Create new card set** button to open the **Create Operator Card Set** panel.

You can specify the following options:

- a. A name for the card set.
 - b. Whether pass phrase recovery will be enabled for the OCS. (Only available if the Security World has pass phrase recovery enabled.)
 - c. Whether the card set can be used remotely. (Only available if the Security World has remote sharing available.) For more information, see [Remote Operator](#).
 - d. Whether this OCS will be persistent.
 - e. Whether this OCS will have a time-out (a period after which the card set must be inserted again).
 - f. The value for the time-out, in seconds.
 - g. The total number of Operator Cards (*N*) that you want this OCS to have. This must be a value in the range 1 – 64.
 - h. The number of Operator Cards needed to re-create a key (*K*). *K* must be less than or equal to *N*.
5. When you have entered all the details, click **Commit**. KeySafe takes you to a new **Create Operator Card Set** panel.



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

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

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



If you are creating the card set in a FIPS 140-2 Level 3 Security World, insert an Administrator Card or an existing Operator Card when prompted.

6. Insert a blank, unformatted card into the reader.

A message is displayed, confirming that the card is blank. Click **OK** to open the **Set Card Protection pass phrase** panel.



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

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

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



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

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



You can change a pass phrase at any time. If you do not set a pass

phrase now, you can use the KeySafe **Change Pass Phrase** option (on the **Examine/Change Card** panel) to add one later. Likewise, if you later decide that you do not need a pass phrase on a card, you can use this option to remove it.

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

9.2. Creating softcards

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

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

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



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



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



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



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

You can create softcards from either:

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

9.2.1. Creating a softcard with ppmk

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

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

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

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

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

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

2. When prompted, type a pass phrase for the new softcard, and press **Enter**.

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

3. When prompted, type the pass phrase again to confirm it, and press **Enter**.

If the pass phrases do not match, **ppmk** prompts you to input and confirm the pass phrase again.

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

9.2.2. Creating softcards with KeySafe

To create a softcard with KeySafe:

1. Start KeySafe. (For an introduction to KeySafe and information on starting the software, see [Using KeySafe](#).)

2. Click the **Softcards** menu button, or select **Softcards** from the **Manage** menu. KeySafe takes you to the **List Softcards** panel.
3. Click **Create New Softcard** to open the **Create Softcard** panel.
4. Choose parameters for the softcard:
 - a. Enter a name for the softcard. You must provide a valid name for each softcard.
 - b. Choose whether you want pass phrase replacement to be enabled for the softcard.



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

5. Click **Commit**.



If you are creating the softcard in a FIPS 140-2 Level 3 Security World, insert an Administrator Card or an existing Operator Card when prompted.

The **Set Softcard Protection Pass Phrase** pane is displayed.

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

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

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

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

8. Click the **OK** button.

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

9.3. Erasing cards and softcards

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



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

You can erase Operator Cards using the unit front panel, KeySafe or the `createocs` utility. You can also use these methods to erase Administrator Cards other than those in the current Security World's ACS (for example, you could use these methods to erase the remaining Administrator Cards from an incomplete set that has been replaced or Administrator Cards from another Security World).



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

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



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

9.3.1. FIPS 140-2 Level 3-compliant Security Worlds

When you attempt to erase cards for a Security World that complies with FIPS 140-2 Level 3, you are prompted to insert an Administrator Card or Operator Card from an existing set. You may need to specify to the application the slot you are going to use to insert the card. You need to insert the card only once in a session. You can therefore use one of the cards that you are about to erase.

9.3.2. Erasing card sets using the Connect front panel

To erase a card set using the front panel, follow this procedure:

1. From the main menu select: **Security World mgmt > Card operations > Erase card**
2. Insert the card set that you want to erase. The card is read.
3. You are asked to confirm that you want to erase this card from the card set.
4. To confirm, press the right-hand navigation button.
5. You are asked once again if you want to erase this card.

6. To confirm, press the right-hand navigation button.

9.3.3. Erasing cards with KeySafe

To erase a card using KeySafe use the following procedure:

1. Start KeySafe. (For an introduction to KeySafe and information on starting the software, see [Using KeySafe](#).)
2. Click the **Card Sets** menu button. KeySafe takes you to the Card Operations panel.
3. Click the **Examine/Change Card** navigation button. KeySafe takes you to the **Examine/Change Card** panel.
4. Insert the card that you want to erase into the reader.
5. Click the **Erase Card** button. You do not need to supply the pass phrase (if there is one) to erase an Operator Card.
6. KeySafe asks you to confirm that you want to erase this card. If you are sure that you want to erase it, click the **Yes** button.



Erasing a card does not erase the keys protected by that card. The keys are still listed on the keys panel but are unusable.

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

7. After erasing a card, KeySafe displays a dialog to confirm that the card has been erased. Click **OK** to continue using KeySafe.



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

9.3.4. Erasing cards using the command line

To erase a card from the command line, run the command:

```
createocs -m|--module=<MODULE> -e|--erase
```

This command uses the following options:

Option	Description
<code>-m --module=<MODULE></code>	These options specify the module number of the module. If you only have one module, <i>MODULE</i> is 1.
<code>-e --erase</code>	These options specify that you want to erase a card (rather than create an OCS).



If you have more than one card reader and there is more than one card available, `createocs` prompts you to confirm which card you wish to erase. Use **[Ctrl][X]** to switch between cards.

If you have created a FIPS 140-2 Level 3 compliant Security World, you must provide authorization in order to erase or create Operator Cards. You can obtain this authorization from any card in the ACS or from any Operator Card in the current Security World, including cards that are to be erased. After you insert a card containing this authorization, `createocs` prompts you to insert the card to be erased.

As an alternative, you can reformat using `slotinfo --format`.

9.3.5. Erasing softcards

Erasing a softcard deletes all information about the softcard from the host. You can erase softcards using KeySafe or with the `ppmk` command-line utility.

9.3.5.1. Erasing softcards with KeySafe

To erase softcards with KeySafe:

1. Start KeySafe.
2. Click the **Softcards** menu button. KeySafe takes you to the Softcard Operations panel.
3. Select the softcard you want to erase from the list.
4. Click the **Discard Softcard** button.
5. KeySafe asks you to confirm that you want to erase this card. Click **Yes** to confirm.
6. After erasing a softcard, KeySafe displays a dialog box to confirm that the card has been erased. Click **OK** to continue using KeySafe.

9.3.5.2. Erasing softcards with ppmk

To erase a softcard with `ppmk`, open a command window, and give the command:

```
ppmk --delete <NAME>|<IDENT>
```

In this command, you can identify the softcard to be erased either by its name (**NAME**) or by its logical token hash as listed by **nfkminfo** (**<IDENT>**).

If you are working within a FIPS 140-2 Level 3 compliant Security World, you must provide authorization to erase softcards; **ppmk** prompts you to insert a card that contains this authorization. Insert any card from the ACS or any Operator Card from the current Security World.

If you insert an Administrator Card from another Security World or an Operator Card that you have just created, **ppmk** displays an error message and prompts you to insert a card with valid authorization. When **ppmk** has obtained the authorization from a valid card or if no authorization is required, it completes the process of erasing the softcard.

9.4. Viewing cards and softcards

It is often necessary to obtain information from card sets, usually because for security reasons they are left without any identifying markings.

To view details of all the Operator Cards in a Security World or details of an individual Operator Card, you can use the front panel, KeySafe or the **nfkminfo** command-line utility. To check which pass phrase is associated with a card, you can use the front panel or the **cardpp** command-line utility.

To list all softcards in a Security World or to show details of an individual softcard, you can use the **ppmk** or **nfkminfo** command-line utilities. To check which pass phrase is associated with a softcard, you can use the **ppmk** command-line utility.

9.4.1. Viewing card sets using the Connect front panel

You can use the unit front panel to view details of all the Operator Cards in a Security World or to view details of an individual Operator Card.

To view a list of all the card sets in the Security World, from the front panel select **Security World mgmt > Cardset operations > List cardsets**.

To view details of a single card using the unit front panel:

1. Insert the card into the unit.
2. From the main menu, select **Security World mgmt > Card operations > Card details**.

3. The type of the card (Administrator or Operator) is displayed with the number of the card in the card set.

9.4.2. Viewing card sets with KeySafe

You can use KeySafe to view details of all the Operator Cards in a Security World, details of individual OCSs or details of an individual Operator Card.

9.4.2.1. Examining a Card

In order to view information about individual cards with KeySafe, follow these steps:

1. Start KeySafe. (For an introduction to KeySafe and information on starting the software, see [Using KeySafe](#).)
2. Click the **Card Sets** menu button, or select the **Card sets** menu item from the **Manage** menu. KeySafe takes you to the **List Operator Card Sets** panel.
3. Click **Examine/Change Card** to open the **Examine/Change Card** panel.
4. Insert a card into the appropriate smart card slot. KeySafe displays information about the smart card currently in the slot. If there is no smart card in the slot, KeySafe displays a message **Card slot empty - please insert the card that you want to examine**.

From the **Examine/Change Card** panel, you can also:

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

9.4.2.2. List an Operator Card Set

In order to view information about whole OCSs with KeySafe, follow these steps:

1. Start KeySafe. (For an introduction to KeySafe and information on starting the software, see [Using KeySafe](#).)
2. Click the **Card Sets** menu button, or select the **Card sets** menu item from the **Manage** menu. KeySafe takes you to the **List Operator Card Sets** panel, which displays information about all OCSs in the current Security World.

From the **List Operator Card Sets** panel, you can also:

- Examine / change a card (see [Examining a Card](#))

- Create a new card set (see [Creating an Operator Card Set with KeySafe](#))
- Replace an Operator Card Set (see [Replacing OCSs with KeySafe](#))
- Discard a card set (see [Erasing cards with KeySafe](#)).

9.4.3. Viewing card sets using the command line

You can use the `nfkminfo` command-line utility to view details of either all the Operator Cards in a Security World or of an individual Operator Card.

To list the OCSs in the current Security World from the command line, open a command window, and give the command:

```
nfkminfo --cardset-list
```

In this command, `--cardset-list` specifies that you want to list the operator card sets in the current Security World.

`nfkminfo` displays output information similar to the following:

```
Cardset summary - 1 cardsets:           (in timeout, P=persistent, N=not)
Operator logical token hash             k/n timeout name
hash                                   1/1 none-N name
```

To list information for a specific card, use the command:

```
nfkminfo <TOKENHASH>
```

In this command, `<TOKENHASH>` is the `Operator logical token hash` of the card (as listed when the command `nfkminfo --cardset-list` is run).

This command displays output information similar to the following:

```
name           "name"
k-out-of-n     1/1
flags          NotPersistent
timeout        none
card names     ""
hkltu          794ada39038fa8c4e9ea46a24136bbb2b8b337f2
```



Not all software can give names to individual cards.

9.4.4. Viewing softcards

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

9.4.4.1. Viewing softcards with KeySafe

To view a softcard with KeySafe, follow these steps:

1. Start KeySafe.
2. Click the **Softcards** menu button. KeySafe takes you to the **Softcard Operations** panel.
3. Click the **List Softcards** navigation button. KeySafe takes you to the **List Softcards** panel, which displays information about all softcards in the current Security World.

From the **List Softcards** panel, you can also choose to remove a softcard from the Security World. For more information about this procedure, see [Erasing cards and soft-cards](#).

9.4.4.2. Viewing softcards with nfkminfo

To list the softcards in the current Security World using the **nfkminfo** command-line utility, give the command:

```
nfkminfo --softcard-list
```

In this command **--softcard-list** specifies that you want to list the softcards in the current Security World.

To show information for a specific softcard using the **nfkminfo** command-line utility, give the command:

```
nfkminfo --softcard-list <IDENT>
```

In this command **<IDENT>** is the softcard's logical token hash (as given by running the command **nfkminfo --softcard-list**). This command displays output information similar to the following:

```
SoftCard
name      "mysoftcard"
hkltu     7fb95888ea2850d4e3ffcc8f0c22100937344308
Keys protected by softcard 7fb95888ea2850d4e3ffcc8f0c22100937344308:
AppName simple      Ident mykey
AppName simple      Ident myotherkey
```

9.4.4.3. Viewing softcards with ppmk

To list the softcards in the current Security World using the **ppmk** command-line utility, use the command:

```
ppmk --list
```

In this command **--list** specifies that you want to list the softcards in the current Security World.

In order to view the details of a particular softcard using the **ppmk** command-line utility, give the command:

```
ppmk --info <NAME>|<IDENT>
```

In this command, you can identify the softcard whose details you want to view either by its name (**<NAME>**) or by its logical token hash (as given by running the command **nfkminfo --softcard-list**).

9.4.5. Verifying the pass phrase of a card or softcard

9.4.5.1. Verifying the pass phrase of a card using the Connect front panel

To verify the pass phrase associated with a card using the unit front panel:

1. Insert the card into the unit.
2. From the main menu, select **Security World mgmt > Card operations > Check PIN**.

The type of the card (Administrator or Operator) is displayed with the number of the card in the card set.

3. If this is the card that you want to check, press the right-hand navigation to confirm.
4. Enter the pass phrase.

If the pass phrase that you entered is correct, a confirmation message is shown. Otherwise, an error is reported.

9.4.5.2. Verifying the pass phrase of a card with cardpp

To verify the pass phrase associated with a card using the **cardpp** command-line utility, use the command:


```
cardpp --check [-m|--module=<MODULE>]
```

This command uses the following options:

Option	Description
<code>--check</code>	This option tells <code>cardpp</code> to check the pass phrase.
<code>--module=<MODULE></code>	This option specifies the number of the module to use. If you only have one module, <code><MODULE></code> is 1. If you do not specify a module number, <code>cardpp</code> uses all modules by default.

The `cardpp` utility polls all available slots; if there is no card inserted, it prompts you to insert one. If the card belongs to this Security World, `cardpp` either tells you if no pass phrase is set or prompts you to enter the pass phrase and checks to see if it is correct.

9.4.5.3. Verifying the pass phrase of a softcard with ppmk

In order to verify the pass phrase of a particular softcard, open a command window, and give the command:

```
ppmk --check <NAME>|<IDENT>
```

In this command, you can identify the softcard whose pass phrase you want to verify either by its name (`<NAME>`) or by its logical token hash (as given by running the command `nfk-minfo --softcard-list`).

`ppmk` prompts you to enter the pass phrase and then tells you whether the pass phrase you entered is correct for the specified softcard.

9.5. Changing card and softcard pass phrase

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

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

that does currently have one.

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

9.5.1. Changing known pass phrase

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

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



Prior to Security World Software v11.72, we set no absolute limit on the length of a pass phrase. However, some applications may not accept a pass phrase longer than 255 characters. Likewise, the Security World does not impose restrictions on which characters you can use, although some applications may not accept certain characters. Entrust recommends that your password only contains 7-bit ASCII characters:

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

See [Maximum pass phrase length](#) for more about pass phrase length when using Security World Software v11.72.

9.5.1.1. Changing known pass phrase from the unit front panel

To change the pass phrase of a card using the unit front panel:

1. Insert the card.
2. From the main menu, select **Security World mgmt > Card operations > Change PIN**.
3. Select the card whose pass phrase you want to change.
4. Enter the old pass phrase, and then enter it again to confirm it.
5. Enter the new pass phrase. If you do not want this card to have a pass phrase, select **NO** at the prompt.

9.5.1.2. Changing known pass phrase with KeySafe

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

1. Start KeySafe. (For an introduction to KeySafe and information on starting the soft-

ware, see [Using KeySafe.](#))

2. Click **Card sets**, or select **Card sets** from the **Manage** menu. The **List Operator Card Sets** panel is displayed.
3. Click **Examine / change card** to open the **Examine / Change Card** panel.
4. Click **Change pass phrase**. The Set Card Protection pass phrase panel is displayed.
5. Enter the old pass phrase, and click the **OK** button.
6. A screen is displayed asking **Do you want to set a pass phrase?**. Select **Yes**.
7. Enter your new pass phrase, and enter it again in the second box as confirmation of the change.
8. Click **OK**.

9.5.1.2.1. Changing a known softcard pass phrase with KeySafe

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

1. Start KeySafe. (For an introduction to KeySafe and information on starting the software, see [Using KeySafe.](#))
2. Click the **Softcards** menu button, or select **Softcards** from the **Manage** menu. KeySafe takes you to the **List Softcards** panel.
3. Select the softcard for which you want to change the pass phrase, and click the **Change Pass phrase** button. KeySafe takes you to the **Change/Recover Softcard Pass phrase** panel.



If a softcard is listed as PIN Recovery Enabled = No, then you will be unable to change the pass phrase.

4. Select the softcard whose pass phrase you want to change, and click the **Change Pass phrase** button. KeySafe takes you to the **Get Softcard Protection pass phrase** panel.
5. Enter the old pass phrase, and click the **OK** button.

KeySafe either displays an error dialog (if the pass phrase is not correct) or takes you to the **Set Softcard Protection** pass phrase panel.

6. Enter your new pass phrase, and enter it again in the second field to confirm the pass phrase is correct.
7. Click the **OK** button.

After changing a pass phrase, KeySafe displays a dialog to confirm that the pass phrase has been successfully changes.

8. Click the **OK** button to continue using KeySafe.

9.5.1.3. Changing known pass phrase with cardpp

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



With Security World Software v11.72 and later, pass phrases are limited to a maximum length of 254 characters, when using **cardpp**. See [Maximum pass phrase length](#).

To change a known card's pass phrase with the **cardpp** command-line utility, take the following steps:

1. Run the **cardpp** utility using the command:

```
cardpp --change [-m|--module=<MODULE>]
```

If you only have one HSM, **<MODULE>** is 1. If you do not specify an HSM number, **cardpp** uses all HSMs by default.

2. If prompted, insert the card whose pass phrase you want to change. (If there is a card already in the slot, you are not prompted.)
3. If prompted, enter the existing pass phrase for the card (If the card has no current pass phrase you are not prompted.) If you enter the pass phrase correctly, **cardpp** prompts you to enter the new pass phrase.
4. Enter a new pass phrase, and then enter it again to confirm it.

After you have confirmed the new pass phrase, **cardpp** changes the card's pass phrase.

9.5.1.4. Changing known softcard pass phrase with ppmk



With Security World Software v11.72 and later, pass phrases are limited to a maximum length of 254 characters, when using **ppmk**. See [Maximum pass phrase length](#) for more information.

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

1. Give the following command:

```
ppmk --change <NAME>|<IDENT>
```

In this command, you can identify the softcard whose pass phrase you want to change

either by its name (<NAME>) or by its logical token hash as listed by `nfkminfo` (<IDENT>).

`ppmk` prompts you to enter the old pass phrase.

2. Type the old pass phrase, and press **Enter**. If you enter the old pass phrase correctly, `ppmk` prompts you to enter the new pass phrase.
3. Type the old pass phrase, and press **Enter**. Type the new pass phrase again, and press **Enter** to confirm it.

After you have confirmed the new pass phrase, `ppmk` then changes the softcard's pass phrase.

9.5.2. Changing unknown or lost pass phrase

9.5.2.1. Changing unknown card pass phrase with `cardpp`

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

To change an unknown card pass phrase with the `cardpp` command-line utility:

- Run a command of the form:

```
cardpp --recover [--module=<MODULE>]
```

In this command, <MODULE> specifies the number of the hardware security module to use. If you only have one hardware security module, <MODULE> is 1. If you do not specify a number, `cardpp` uses all hardware security modules by default.

- As prompted, insert the appropriate number of cards from the ACS required to authorize pass phrase replacement.
- When prompted, insert the Operator Card whose pass phrase you want to replace. To replace its pass phrase:
 - a. When prompted, type the new pass phrase, and then press **Enter**.
 - b. When prompted, type the new pass phrase again to confirm it, and then press **Enter**.

`cardpp` sets the new pass phrase, and then prompts you for another Operator Card.

- Repeat the process in the previous step to change the pass phrase on further cards, or

press **Q** to quit.



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

9.5.2.2. Replacing unknown pass phrase with ppmk

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

To change an unknown softcard pass phrase with the **ppmk** command-line utility:

1. Run a command of the form:

```
preload --admin=p ppmk --recover <NAME>|<IDENT>
```

In this command, you can identify the softcard by its **<NAME>** or by its **<IDENT>** (its logical token hash as shown in output from the **nfkminfo** command-line utility).

2. As prompted, insert the appropriate number of cards from the ACS required to authorize pass phrase replacement.
3. When prompted, type the new pass phrase, and then press **Enter**.
4. When prompted, type the new pass phrase again to confirm it, and then press **Enter**.

If the pass phrase does not match, **ppmk** prompts you to input and confirm the pass phrase again.

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



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

9.6. Replacing Operator Card Sets



Replacing an OCS requires authorization from the ACS of the Security World to which it belongs. You cannot replace an OCS unless you have the required number of cards from the appropriate ACS.

If you have lost a card from a card set, or you want to migrate from standard nShield cards to nShield Remote Administration Cards, you should use one of the following:

- The **rocs** utility
- The KeySafe **Replace Operator Card Set** option.

Accessed from the **Card Operations** panel.



You cannot mix standard nShield cards with nShield Remote Administration Cards. in the same set.

We recommend that after you have replaced an OCS, you then erase the remaining cards in the old card set and remove the old card set from the Security World. For more information, see [Erasing cards and softcards](#).

Deleting the information about an OCS from the client does not remove the data for keys protected by that card set. On the KeySafe **Key Operations** panel), such keys are listed as being protected by **Deleted Card Set**.

To prevent you from losing access to your keys if the smart card you are using as the Operator Card is lost or damaged, Entrust supplies several utilities that can recover the keys protected by the lost Operator Card to another OCS

Replacing one OCS with another OCS also transfers the keys protected by the first OCS to the protection of the new OCS.

When you replace an OCS or softcard and recover its keys to a different OCS or softcard, the key material is not changed by the process. The process deletes the original Security World data (that is, the encrypted version of the key or keys and the smart card or softcard data file) and replaces this data with host data protected by the new OCS or softcard.

To replace an OCS or softcard, you must:

- Have enabled OCS and softcard replacement when you created the Security World



If you did not enable OCS and softcard replacement, you cannot recover keys from lost or damaged smart cards or softcards.

- Have created the original OCS using the front panel of the unit, **createocs**, **createocs-simple**, KeySafe, or the nShield PKCS #11 library version 1.6 or later



If you initialized the token using **ckinittoken** from the nShield PKCS #11 library version 1.5 or earlier, you must contact Support to arrange for them to convert the token to the new format while you still possess a valid card.

- Have a sufficient number of cards from the ACS to authorize recovery and replacement



All recovery and replacement operations require authorization from the ACS. If any of the smart cards in the ACS are lost or damaged, immediately replace the entire ACS.

- Have initialized a second OCS using the front panel of the unit, **createocs**, **createocs-simple**, KeySafe, or the nShield PKCS #11 library version 1.6 or later.



The new OCS need not have the same *K/N* policy as the old set.

If you are sharing the Security World across several client computers, you must ensure that the changes to the host data are propagated to all your computers. One way to achieve this is to use client cooperation. For more information, see [Setting up client cooperation](#).

9.6.1. Replacing OCSs from the unit front panel

To replace an OCS from the unit front panel, follow these steps:

1. From the main menu, select **Security World mgmt > Admin operations > Recover keys**.
2. Select **all** to recover all keys in the Security World, or select the application for which you want to recover the keys.
3. If you selected an application, select the keys that you want to recover.
4. Insert the required number of Administrator Cards to recover keys, and enter their pass phrases if required.
5. Insert the required number of Operator Cards, and enter their pass phrases if required.

When you have inserted the required number of cards, details of the recovered key are displayed.

6. Check the key details are correct and then scroll down and select **Recover key**.

You can also select **More info** to see more information about the keys.

A message is displayed when the keys are recovered.

9.6.2. Replacing OCSs with KeySafe

In order to replace an OCS, you must have another OCS onto which to copy the first set's data. If you do not already have an existing second OCS, you must create a new one. For more information, see [Creating Operator Card Sets \(OCSs\)](#).

When you have a second OCS ready, follow these steps in order to replace the first OCS:

1. Start KeySafe. (For an introduction to KeySafe and information on starting the software, see [Using KeySafe.](#))
2. Click the **Card Sets** menu button, or select **Card Sets** from the **Manage** menu. KeySafe takes you to the **List Operator Card Sets** panel.
3. Click **Replace card set**. KeySafe takes you to the **Replace card set** panel.

This panel lists existing OCSs in tabular form. For each card set it displays:

Attribute	Description
Name	The name of the card set.
Required (K)	The number of cards needed to re-create a key.
Total (N)	The total number of cards in the set.
Persistent	Indicates whether or not the card set is persistent.
Timeout	The timeout value of the card, in seconds
Recoverable Key Count	The number of private keys protected by this card set that are recoverable.
Nonrecoverable Key Count	The number of private keys protected by this card set that are not recoverable.

You can click and drag with your mouse in order to resize the column widths and to rearrange the column order of this table. Clicking a column heading sorts the rows in ascending order based on that column heading.

4. Select an OCS that you want to replace, and click **Replace card set**.



If an OCS does not have any recoverable keys, it cannot be replaced.

5. KeySafe takes you to the **Load Administrator Card Set** panel, where it prompts you to insert cards from the ACS in order to authorize the action. Each time you insert an Administrator Card into the smart card of the hardware security module slot, you must click the **OK** button to load the card.



Only insert your ACS into a module that is connected to a trusted server.

6. When you have loaded enough cards from the ACS to authorize the procedure, KeySafe takes you to the **Load Operator Card Set** panel, where it prompts you to insert the OCS that is to protect the recoverable keys (this is the OCS onto which you are copying data from the OCS you are replacing). Each time you insert a card from the new OCS into the smart card slot of the hardware security module, you must click the

OK button.

When you have loaded enough cards from the new OCS, KeySafe creates new working versions of the recoverable keys that are protected by this card set.

KeySafe deletes the original host data for all recovered keys and replaces this data with host data that is protected by the new OCS. If there are no nonrecoverable keys protected by the card set, KeySafe also removes the old card set from the Security World. However, if the OCS has nonrecoverable keys, the host data for the original card set and for the nonrecoverable keys is not deleted. These keys can only be accessed with the original OCS. If you want to delete these files, use the **Remove OCS** option.

7. When the process is complete, KeySafe displays a dialog indicating that the OCS has been successfully replaced. Click the **OK** button. KeySafe returns you the Replace Operator Card Set panel, where you may replace another OCS or choose a different operation.

9.6.3. Replacing OCSs or softcards with rocs

You can use the **rocs** command-line utility interactively, or you can supply all the parameters using the command line.

9.6.3.1. Using rocs interactively

To use the **rocs** command-line utility interactively, run it without any parameters:

```
rocs
```

rocs displays the following prompt:

```
'rocs' key recovery tool
Useful commands: 'help', 'help intro', 'quit'.
rocs >
```

In order to use **rocs** to replace an OCS or recover keys to a softcard, take the following steps:

1. You must select a hardware security module to use by using the **module** command, which is described in the section [module <number>](#).
2. List the OCSs and softcards in the current Security World by using the **list cardsets** command, which is described in the section [list cardsets](#).
3. Select the OCS or softcard to which you want to transfer the keys by using the **target**

command, which is described in the section [target <cardset-spec>](#).



Keys protected by an OCS can only be recovered to another OCS, and not to a softcard. Likewise, softcard-protected keys can only be recovered to another softcard, and not to an OCS.

4. List the keys in the current Security World using the **list keys** command, which is described in the section [list keys](#).
5. Select the keys that are to be recovered (from a different OCS or softcard than the one you selected for key transfer) by using the **mark** command, which is described in the section [mark <key-spec>](#).
6. If you have selected any keys by mistake, deselect them by using the **unmark** command, which is described in the section [unmark <key-spec>](#).
7. After you have selected the keys that are to be recovered, transfer these keys by using the **recover** command, which is described in the section [recover](#).

rocs prompts you to insert a card from the ACS.

8. Insert a card from the ACS.

rocs prompts you for the pass phrase for this card. This action is repeated until you have loaded the required number of cards from the ACS.

If you do not have the required number of cards from the ACS, press **Q** and then **Enter**. The **rocs** utility returns you to the **rocs >** prompt without processing any keys.



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

9. If you are recovering keys to an OCS:
 - a. **rocs** prompts you to insert a card from the first OCS that you have selected as the target. OCSs are processed in ascending numerical order as listed by the **list cardsets** command.
 - b. Insert a card from this OCS.
 - c. **rocs** prompts you for the pass phrase for this card. This action is repeated until you have loaded the required number of cards from the OCS.

If you are recovering keys to a softcard, **rocs** prompts you for the pass phrase for the softcard that you have selected as the target.

If you decide that you do not want to transfer the keys to the selected card set or softcard, press **Q** and then **Enter** (to quit. **rocs** returns you to the **rocs >** prompt and does

not process any further OCSs or softcards.

When you have loaded the target softcard or the required number of cards from the target OCS, **rocs** transfers the selected keys to the target OCS or softcard.

If you have selected other target OCSs or softcards, **rocs** prompts for a card from the next OCS.

10. Repeat step 9 for each selected target.
11. If you have transferred the correct keys, write the key blobs to disk by using the **save** function (described in the section [save <key-spec>](#)). If you have transferred a key by mistake, you can restore it to its original protection by using the **revert** command (described in the section [revert <key-spec>](#)).

At the **rocs** prompt, you can use the following commands:

- **help <topic>**
- **help intro**
- **list cardsets**
- **list keys**
- **mark <key-spec>**
- **module <number>**
- **quit**
- **recover**
- **rescan**
- **revert <key-spec>**
- **save <key-spec>**
- **status**
- **target <cardset-spec>**
- **unmark <key-spec>**



You can specify a command by typing enough characters to identify the command uniquely. For example, for the **status** command, you can type **st** and then press **Enter**.

9.6.3.1.1. help

With no arguments specified, **help** shows a list of available commands with brief usage messages and a list of other help topics. With an argument, **help** shows detailed help information about a given topic.

`help intro` displays a brief step-by-step guide to using `rocs`.

9.6.3.1.2. list cardsets

This command lists the OCSs and softcards in the current Security World.

For example:

```
No.
Name           Keys (recov) Sharing
1 test         6 (6)       3 of 5; 20 minute timeout
2 test2        3 (2)       2 of 3
3 test3        1 (1)       1 of 1; persistent
```

In this output:

Output	Description
No.	The card set or softcard number, which you can use to identify this card set in <code>rocs</code> commands.
Name	The OCS or softcard name.
Keys	The number of keys protected by this OCS or softcard.
(recov)	The number of keys protected by this OCS or softcard.
Sharing	The <i>K</i> of <i>N</i> parameters for this OCS.
persistent	The OCS is persistent and does not have a time-out set.
### minute timeout	The OCS is persistent and has a time-out set.

9.6.3.1.3. list keys

This command lists the keys in the current Security World, as in the following example:

```
No.
Name           App           Protected by
1 rsa-test     hwcrhk       module
2 Id: uc63e0ca3cb032d71c1c pkcs11 test2
R 3 Server-Cert pkcs11 test --> test2
4 Id: uc63e0ca3cb032d71c1c pkcs11 test --> test3
5 Server-Cert  pkcs11 module (test ---> fred2)
```

In this output:

Output	Description
No.	The key number, which you can use in <code>mark</code> and <code>unmark</code> commands.

Output	Description
Name	The key name.
App	The application with which the key is associated.
Protected by	This indicates the protection method (see table below).

In this output, the protection methods include:

Method	Description
module	Key protected by the Security World.
name	Key protected by the named OCS or softcard.
name-->name2	Key protected by the OCS or softcard <i>name1</i> marked for recovery to OCS or softcard <i>name2</i> .
module (name)	PKCS #11 public object. These are protected by the Security World but associated with a specific OCS or softcard.
module (name-->name2)	PKCS #11 public object marked for recovery.

9.6.3.1.4. mark <key-spec>

This command marks the listed keys that are to be recovered to the target OCS or softcard. You can mark one or more keys by number, *ident*, OCS or softcard, or hash. For more information, see [Specifying keys](#).

To mark more than one key at a time, ensure that each *key-spec* is separated from the other by spaces, as in the following example:

```
mark key-spec1 key-spec2 key-spec3
```

If you have not selected a target OCS or softcard, or if **rocs** cannot parse the *key-spec*, then **rocs** displays an error message.

You can mark and remark the keys to be recovered to various target OCSs or softcards. Remarking a key displaces the first target in favor of the second target.



Keys protected by an OCS can only be recovered to another OCS, and not to a softcard. Likewise, softcard-protected keys can only be recovered to another softcard, and not to an OCS.

9.6.3.1.5. module <number>

This command selects the hardware security module to be used. The module number must correspond to a hardware security module in the current Security World. If the hardware security module does not exist, is not in the Security World, or is otherwise unusable, then **rocs** displays an error message and does not change to the selected module.

9.6.3.1.6. quit

This command allows you to leave **rocs**. If you attempt to **quit** when you have recovered keys but have not saved them, **rocs** displays a warning.

9.6.3.1.7. recover

This command transfers the marked keys to their target OCSs or softcards. This operation is not permanent until you save these keys by using the **save** command.

9.6.3.1.8. rescan

This command updates the card set and key information.

9.6.3.1.9. revert <key-spec>

This command returns keys that have been recovered, but not saved, to being protected by the original protection method. If the selected keys have not been recovered, **rocs** displays an error message.

9.6.3.1.10. save <key-spec>

This command writes the new key blobs to disk. If you specify a key-spec, only those keys are saved. Otherwise, all recovered keys are saved.

9.6.3.1.11. status

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

9.6.3.1.12. target <cardset-spec>

This command selects a given OCS or softcard as the target. You can specify the card set or softcard name, the number returned by **list cardsets**, or the hash.

9.6.3.1.13. unmark <key-spec>

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

9.6.3.2. Using rocs from the command line

You can select all the options for **rocs** using the command line by running a command of the form:

```
rocs -m|--module=<MODULE> [-t|--target=<CARDSET-SPEC>] [-k|--keys=<KEYS-SPEC>] [-c|--cardset=<CARDSET-SPEC>] [-i|--interactive]
```

In this command:

Option	Description
-m, --module=<MODULE>	These options specify the number of the hardware security module to use.
-t, --target=<CARDSET-SPEC>	These options specify the OCS or softcard to be used to protect the keys. For more information, see Specifying card sets .
-k, --keys=<KEYS-SPEC>	These options select the keys to be recovered. For more information, see Specifying keys .
-c, --cardset=<CARDSET-SPEC>	These options select all keys that are protected by the given OCS or softcard. For more information, see Specifying card sets .
-i, --interactive	These options force rocs to start interactively even if you have already selected keys.

You must specify the target before you specify keys.

You can use multiple **--keys=<KEYS-SPEC>** and **--cardset=<CARDSET-SPEC>** options, if necessary.

You can specify multiple targets on one command line by including separate **--keys=<KEYS-SPEC>** or **--cardset=<CARDSET-SPEC>** options for each target. If a key is defined by **--keys=<KEYS-SPEC>** or **--cardset=<CARDSET-SPEC>** options for more than one target, it is transferred to the last target for which it is defined.

If you have selected a hardware security module, a target OCS or softcard, and keys to recover but have not specified the **--interactive** option, **rocs** automatically recovers the keys. **rocs** prompts you for the ACS and OCS or softcard. For more information, see [Using rocs interactively](#).



If you use **rocs** from the command line, all keys are recovered and saved automatically. You cannot revert the keys unless you still have cards

from the original OCS.

If you do not specify the target and keys to recover, or if you specify the `--interactive` option, `rocs` starts in interactive mode with the selections you have made. You can then use further `rocs` commands to modify your selection before using the `recover` and `save` commands to transfer the keys.

9.6.3.3. Specifying card sets

The value of `<CARDSET-SPEC>` identifies one or more OCSs or softcards. It may have any of the following forms:

Value	Description
<code>[number] cardset-number</code>	A value of this form selects the OCS or softcard with the given number from the list produced by the <code>list cardsets</code> command.
<code>[name] cardset-name</code>	A value of this form selects card sets or softcards by their names (the card set or softcard name may be a wildcard pattern in order to select all matching OCSs or softcards).
<code>hash cardset-hash</code>	A value of this form selects the OCS or softcard with the given hash.

In order to specify multiple OCSs or softcards, include several `CARDSET-SPEC`'s using the command line.



Keys protected by an OCS can only be recovered to another OCS, and not to a softcard. Likewise, softcard-protected keys can only be recovered to another softcard, and not to an OCS.

9.6.3.4. Specifying keys

The `--keys=<KEYS-SPEC>` option identifies one or more keys. It may have any of the following forms:

Value	Description
<code>mark key-number</code>	<p>A value of this form selects the key with the given number from the list produced by the <code>list keys</code> command. Examples of usage are:</p> <pre>rocs -t <target_OCS> -k <key_number></pre> <p>and</p> <pre>rocs -t <target_OCS> -k "mark 56"</pre>
<code>appname_.keyident</code>	<p>A value of this form selects keys by their internal application name and <code>ident</code>. You must supply at least one of <code>appname</code> or <code>keyident</code>, but you can use wildcard patterns for either or both in order to select all matching keys. An example of usage is:</p> <pre>rocs -t <target_OCS> --keys="simple:simplekey"</pre>
<code>hash keyhash</code>	<p>A value of this form selects the key with the given key hash. An example of usage is:</p> <pre>rocs -t <target_OCS> --keys="hash e364[...]"</pre>
<code>--cardset cardset-spec</code>	A value of this form selects all keys protected by a given card set.

9.7. Replacing the Administrator Card Set

Replacing the ACS requires a quorum of cards from the current ACS (K/N) to perform the following sequence of tasks:

1. loading the secret information that is to be used to protect the archived copy of the Security World key.
2. creating a new secret that is to be shared between a new set of cards
3. creating a new archive that is to be protected by this secret.

If you discover that one of the cards in the current ACS has been damaged or lost, or you want to migrate from standard nShield cards to nShield Remote Administration Cards, you should use one of the following to create a new set:

- The `racs` utility
- The front panel of the Connect
- The KeySafe **Replace Administrator Card Set** option

Accessed from the **Card Operations** panel



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



You cannot mix nShield cards with nShield Remote Administration Cards in the same set.



Replacing the ACS modifies the **world** file. In order to use the new ACS on other machines in the Security World, you must copy the updated **world** file to all the machines in the Security World after replacing the ACS. Failure to do so could result in loss of administrative access to the Security World.



We recommend that you erase your old Administrator Cards as soon as you have created the new ACS. An attacker with the old ACS and a copy of the old host data could still re-create all your keys. With a copy of a current backup, they could even access keys that were created after you replaced the ACS.



Before you start to replace an ACS, you must ensure that you have enough blank cards to create a complete new ACS. If you start the procedure without enough cards, you will have to cancel the procedure part way through.

9.7.1. Replacing an Administrator Card Set using the Connect front panel

To replace an ACS:

1. From the main menu, select **Security World mgmt > Admin operations > Replace ACS**.
2. Insert one of the remaining cards from the card set that you want to replace and press the right-hand navigation button.

Continue to insert cards until you have inserted the number of cards required to authorize the process.
3. When prompted, insert a card for the replacement card set and press the right-hand navigation button.
4. If required, specify a pass phrase for the card.
5. Insert cards until the card set is complete. A message confirms that the card set has been created.

6. At this point the modified **world** file has been pushed to the RFS, so make a backup of the modified **world** file on the RFS, preferably in a way that distinguishes it from previous backups.
7. Copy the **world** file to any other Connects in the same Security World, either using the **Security World mgmt > RFS operations > Update World files** option on the Connect concerned or using the **nethsmadmin** utility, see [Using nethsmadmin to copy a Security World to a Connect and check the current version](#).
8. If client cooperation is not enabled, copy the modified **world** file onto any client machines where it is needed.
9. Check that the new Administrator Cards are usable and that their pass phrases have been set as intended, see [Verifying the pass phrase of a card or softcard](#)
10. Erase the Administrator Cards from the old card set. For more information, see [Erasing cards and softcards](#).

9.7.2. Replacing an ACS with KeySafe

When you have enough cards to create a complete new ACS ready and a quorum of the ACS you want to replace, follow these steps:

1. Start KeySafe. (For an introduction to KeySafe and information on starting the software, see [Using KeySafe](#)).
2. Click the **Card sets** menu button, or select **Card sets** from the **Manage** menu. KeySafe takes you to the **List Operator Card Sets** panel.
3. Click the **Replace ACS** navigation button, and KeySafe takes you to the **Replace Administrator Card Set** panel.
4. If you are sure that you want to replace the ACS, click the **Replace ACS** command button
5. KeySafe takes you to the **Load Administrator Card Set** panel, where it prompts you to insert cards from the ACS in order to authorize the action. Each time you insert an Administrator Card into the module's smart card slot, you must click the **OK** button to load the card.



Only insert cards from your ACS into a module that is connected to a trusted server.

6. When you have loaded enough Administrator Cards to authorize the action, KeySafe takes you to the Create Administrator Card Set panel, where it prompts you to insert the cards that are to form the ACS. These must be blank cards or cards that KeySafe can erase. KeySafe will not let you use cards from the existing ACS. If you do not have

enough cards to form a complete new ACS, cancel the operation now.



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

7. When you insert a blank card, KeySafe takes you to the **Set Card Protection Pass Phrase** panel.
8. If you want to set a pass phrase for this Administrator Card:
 - a. Select the **Yes** option.
 - b. Enter the same pass phrase in both text fields.
 - c. Click the **OK** button.

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

9. If you do not want to set a pass phrase for this Administrator Card:
 - a. Select the **No** option.
 - b. Click the **OK** button.
10. After you have created all the Administrator Cards, KeySafe displays a message confirming that the ACS has been successfully replaced.
11. Click the **OK** button, and KeySafe returns you to its introduction panel.

When you have finished replacing the ACS:

1. If you ran KeySafe on a client machine, ensure that there is a copy of the modified **world** file on the RFS.
2. Make a backup of the **world** file, preferably in a way that distinguishes it from previous backups.
3. Copy the **world** file to any other Connects in the same Security World, for example using the `nethsmadmin` utility, see [Using nethsmadmin to copy a Security World to an Connect and check the current version](#).
4. If client cooperation is not enabled, copy the modified **world** file onto any other client machines where it is needed.
5. Check that the new Administrator Cards are usable and that their pass phrases have been set as intended, see [Verifying the pass phrase of a card or softcard](#).

6. Erase the old Administrator Cards; for more information, see [Erasing cards and soft-cards](#).

9.7.3. Replacing an Administrator Card Set with `racs`

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

1. Ensure the hardware security module is in operational mode.
2. Run a command of the form:

```
racs [-m|--module=<MODULE>]
```

In this command, the `-m|--module=<MODULE>` option specifies the `ModuleID (<MODULE>)` of the module to use.

3. When prompted, insert the appropriate quorum of Administrator Cards to authorize the replacement.
4. When prompted that `racs` is writing the new ACS, insert blank cards as necessary on which to write the replacement Administrator Cards.
5. If you ran `racs` on a client machine, ensure that there is a copy of the modified **world** file on the RFS.
6. Make a backup of the **world** file, preferably in a way that distinguishes it from previous backups.
7. Copy the **world** file to any other Connects in the same Security World, for example using the `nethsmadmin` utility, see [Using nethsmadmin to copy a Security World to an Connect and check the current version](#).
8. If client cooperation is not enabled, copy the modified **world** file onto any other client machines where it is needed.
9. Check that the new Administrator Cards are usable and that their pass phrases have been set as intended, see [Verifying the pass phrase of a card or softcard](#).
10. Erase the old Administrator Cards; for more information, see [Erasing cards and soft-cards](#).

10. Application interfaces

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

- nCipherKM JCA/JCE CSP
- PKCS #11 applications
- nShield native and Custom applications
- CodeSafe applications

You can use KeySafe or the **generatekey** utility to generate or import keys for use with your applications (see [Working with keys](#)). By default, KeySafe uses the same mechanisms and supports the same applications as the **generatekey** utility.



You must add the user of any application that uses an nShield HSM to the group **nfast** before the application runs.



If you create keys on a client that is not on the same computer as the RFS, you must copy the key data to the RFS before the nShield HSM can use these keys.

10.1. nCipherKM JCA/JCE CSP

The nCipherKM JCA/JCE CSP (Cryptographic Service Provider) allows Java applications and services to access the secure cryptographic operations and key management provided by Entrust hardware. This provider is used with the standard JCE (Java Cryptographic Extension) programming interface.

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

- the nShield Java package which includes the nShield Java jars and KeySafe.

For more information about the bundles and components supplied on your Security World Software installation media, see the User Guide.

The following versions of Java have been tested to work with, and are supported by, your nShield Security World Software:

- Java 7 (or Java 1.7x)
- Java 8 (or Java 1.8x).
- Java 11

We recommend that you ensure Java is installed before you install the Security World Soft-

ware. The Java executable must be on your system path.

If you can do so, please use the latest Java version currently supported by Entrust that is compatible with your requirements. Java versions before those shown are no longer supported. If you are maintaining older Java versions for legacy reasons, and need compatibility with current nShield software, please contact Entrust nShield Technical Support, <https://nshieldsupport.entrust.com>.

To install Java you may need installation packages specific to your operating system, which may depend on other pre-installed packages to be able to work.

Suggested links from which you may download Java software as appropriate for your operating system:

- <http://www.oracle.com/technetwork/java/index.html>
- <http://www.oracle.com/technetwork/java/all-142825.html>



Detailed documentation for the JCE interface can be found on the Oracle Technology web page <https://docs.oracle.com/en/java/javase/11/security/java-cryptography-architecture-jca-reference-guide.html>.



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

10.1.1. Installing the nCipherKM JCA/JCE CSP

To install the nCipherKM JCA/JCE CSP:

1. In the hardserver configuration file, ensure that:
 - **priv_port** (the port on which the hardserver listens for local privileged TCP connections) is set to 9001
 - **nonpriv_port** (the port on which the hardserver listens for local nonprivileged TCP connections) is set to 9000.

If you need to change either or both of these port settings, you restart the hardserver before continuing the nCipherKM JCA/JCE CSP installation process. For more information, see [Stopping and restarting the hardserver](#).

2. For Java 7 and 8 only. Copy the **nCipherKM.jar** file to the extensions folder of your local Java Virtual Machine installation from the following directory:

- **/opt/nfast/java/classes**

The location of the extensions folder depends on the type of your local Java Vir-

tual Machine (JVM) installation:

JVM type	Extensions folder
Java Developer Kit (JDK)	<code>\$JAVA_HOME/jre/lib/ext</code>
Java Runtime Environment (JRE)	<code>\$JAVA_HOME/lib/ext</code>

In these paths, `$JAVA_HOME` is the home directory of the Java installation (commonly specified in the `JAVA_HOME` environment variable). If you are using Java 11 you do not need to copy the jar file.

3. Add `$JAVA_HOME/bin` to your PATH system variable
4. For Java 7 and 8 only. Install the unlimited strength JCE jurisdiction policy files that are appropriate to your version of Java. JDK 9 and later ship with, and use by default, the unlimited policy files.

The Java Virtual Machine imposes limits on the cryptographic strength that may be used by default with JCE providers. Replace the default policy configuration files with the unlimited strength policy files.

To install the unlimited strength JCE jurisdiction policy files:

- a. If necessary, download the archive containing the Java Cryptography Extension (JCE) Unlimited Strength Jurisdiction Policy Files from the Web site of your Java Virtual Machine vendor.



The Java Cryptography Extension (JCE) Unlimited Strength Jurisdiction Policy Files are covered and controlled by U.S. Export Control laws and may be subject to the export or import laws in other countries. We recommend that you take legal advice before downloading these files from your Java Virtual Machine vendor.

- b. Extract the files `local_policy.jar` and `US_export_policy.jar` from Java Virtual Machine vendor's Java Cryptography Extension (JCE) Unlimited Strength Jurisdiction Policy File archive.
- c. Copy the extracted files `local_policy.jar` and `US_export_policy.jar` into the security directory for your local Java Virtual Machine (JVM) installation:

JVM type	Extensions folder
Java Developer Kit (JDK)	<code>\$JAVA_HOME/jre/lib/security</code>
Java Runtime Environment (JRE)	<code>\$JAVA_HOME/lib/security</code>

In these paths, `$JAVA_HOME` is the home directory of the Java installation (commonly specified in the `JAVA_HOME` environment variable).



Copying the files `local_policy.jar` and `US_export_policy.jar` into the appropriate folder must overwrite any existing files with the same names.

5. Add the nCipherKM provider to the Java security configuration file `java.security` (located in the security directory for your local Java Virtual Machine (JVM) installation).

The `java.security` file contains a list of providers in preference order that is used by the Java Virtual Machine to decide from which provider to request a mechanism instance. Ensure that the nCipherKM provider is registered in the first position in this list, as shown in the following example:

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

For Java 11 you do not need to specify the fully qualified class name for the provider. Instead you can just use the provider name.

```
security.provider.1=nCipherKM
```

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



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

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



If you add the nCipherKM provider as `security.provider.1`, ensure that the subsequent providers are re-numbered correctly. Ensure you do not list multiple providers with the same number (for example, ensure your list of providers does not include two instances of `security.provider.1`, both `com.ncipher.provider.km.nCipherKM` and another provider).

6. Save your updates to the file `java.security`.

When you have installed the nCipherKM JCA/JCE CSP, you must have created a Security World before you can test or use it. For more information about creating a Security World, see [Creating a Security World](#).



If you have a Java Enterprise Edition Application Server running, you must restart it before the installed nCipherKM provider is loaded into the Application Server virtual machine and ready for use.

10.1.1.1. Testing the nCipherKM JCA/JCE CSP installation

After installation, you can test that the nCipherKM JCA/JCE CSP is functioning correctly by running the command.

For Java 7 and Java 8:

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

For Java 11:

```
java --module-path /opt/nfast/java/classes com.ncipher.provider.InstallationTest
```



For this command to work, you must have added `$JAVA_HOME` to your `PATH` system variable).

If the nCipherKM JCA/JCE CSP is functioning correctly, output from this command has the following form:

```
Installed providers:
1: nCipherKM
2: SUN
3: SunRsaSign
4: SunJSSE
5: SunJCE
6: SunJGSS
7: SunSASL
Unlimited strength jurisdiction files are installed.
The nCipher provider is correctly installed.
nCipher JCE services:
Alg.Alias.Cipher.1.2.840.113549.1.1.1
Alg.Alias.Cipher.1.2.840.113549.3.4
Alg.Alias.Cipher.AES
Alg.Alias.Cipher.DES3
```

If the nCipherKM provider is installed but is not registered at the top of the providers list in

the `java.security` file, the `InstallationTest` command produces output that includes the message:

```
The nCipher provider is installed, but is not registered at the top of the providers list in the java.security file.  
See the user guide for more information about the recommended system configuration.
```

In such a case, edit the `java.security` file (located in the security directory for your local JVM installation) so that the nCipherKM provider is registered in the first position in that file's list of providers. For more information about the `java.security` file, see [Installing the nCipherKM JCA/JCE CSP](#).

If the nCipherKM provider is not installed at all, or you have not created a Security World, or if you have not configured ports correctly in the hardserver configuration file, the `InstallationTest` command produces output that includes the message:

```
The nCipher provider is not correctly installed.
```

In such case:

- Check that you have configured ports correctly, as described in [Installing the nCipherKM JCA/JCE CSP](#). For more information about hardserver configuration file settings, see [server_startup](#).
- Check that you have created a Security World. If you have not created a Security World, create a Security World. For more information, see [Creating a Security World](#).
- If you have already created a Security World, repeat the nCipherKM JCA/JCE CSP installation process as described in [Installing the nCipherKM JCA/JCE CSP](#).

After making any changes to the nCipherKM JCA/JCE CSP installation, run the `InstallationTest` command again and check the output.

Whether or not the nCipherKM provider is correctly installed, if the unlimited strength jurisdiction files are not installed or (not correctly installed), the `InstallationTest` command produces output that includes the message:

```
Unlimited strength jurisdiction files are NOT installed.
```

This message means that, because the Java Virtual Machine imposes limits on the cryptographic strength that you can use by default with JCE providers, you must replace the default policy configuration files with the unlimited strength policy files. For information about how to install the unlimited strength jurisdiction files, see [Installing the nCipherKM JCA/JCE CSP](#).

10.1.2. Named Modules in Java 11

The nCipherKM Provider has been implemented as a named module. This means that, for Java 11, if you have added the provider to your `java.security` file, then you can run your application with the `nCipherKM.jar` on the module-path and the Java ServiceLoader class will automatically find it, e.g.

```
java --module-path /opt/nfast/java/classes com.ncipher.provider.InstallationTest
```

Alternatively, you can specify the location of the nCipherKM jar on the classpath:

```
java --class-path /opt/nfast/java/classes/nCipherKM.jar com.ncipher.provider.InstallationTest
```

10.1.3. keytool

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

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

- Sun Microsystems `keytool` utility:

For Java 11, use the following command:

```
java --module-path /opt/nfast/java/classes sun.security.tools.keytool.Main -genkey -storetype nCipher.world -keyalg RSA -sigalg SHA1withRSA -storepass <KeyStore_passphrase> -keystore <KeyStore_path>
```

For Java 8, use the following command:

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

For Java 7, use the following command:

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

- IBM `keytool` utility:

```
java com.ibm.crypto.tools.KeyTool -genkey -storetype nCipher.world -keyalg RSA -sigalg SHA1withRSA
```

```
-storepass <KeyStore_passphrase> -keystore <KeyStore_path>
```

In these example commands, **<KeyStore_passphrase>** is the passphrase for the OCS that protects the KeyStore and **<KeyStore_path>** is the path to that KeyStore.

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

- Sun Microsystems **keytool** utility:

For Java 11, use the following command:

```
java --module-path /opt/nfast/java/classes -Dprotect=module -DignorePassphrase=true
sun.security.tools.keytool.Main -genkey -storetype nCipher.sworld -keyalg RSA -sigalg SHA1withRSA -keystore
<KeyStore_path>
```

For Java 8, use the following command:

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

For Java 7, use the following command:

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

- IBM **keytool** utility:

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

In these example commands, **<KeyStore_path>** is the path to the KeyStore.

By default, the **keytool** utilities use the **MD5withRSA** signature algorithm to sign certificates used with a KeyStore. This signature mechanism is unavailable on modules with firmware version 2.33.60 or later.

10.1.4. Using keys

Only the nCipherKM provider can use keys stored in an nShield KeyStore because the underlying key material is held separately in the Security World.

You can always store nShield keys in an nShield KeyStore. You can also store keys generated by a third-party provider into an nShield KeyStore if both of the following conditions apply:

- the key type is known to the nCipherKM provider
- the Security World is *not* compliant with FIPS 140-2 Level 3.

When you generate an nShield key (or create it from imported key material), that key is associated with an ACL (Access Control List). This ACL prevents the key from being used for operations for which it is unsuited and enforces requirements that certain tokens be presented; for example, the ACL can specify that signing key cannot be used for encryption.

10.1.5. System properties

You can use system properties to control the provider. You set system properties when starting the Java Virtual Machine using a command such as:

```
java -D<property>=<value> <MyJavaApplication>
```

In this example command, **<property>** represents any system property, **<value>** represents the value set for that property, and **<MyJavaApplication>** is the name of the Java application you are starting. You can set multiple system properties in a single command, for example:

```
java -Dprotect=module -Dignorepass phrase=true <MyJavaApplication>
```

The available system properties and their functions as controlled by setting different values for a property are described in the following table:

Property	Function for different values
JCECSP_DEBUG	This property is a bit mask for which different values specify different debugging functions; the default value is 0. For details about the effects of setting different values for this property, see JCECSP_DEBUG property values .
JCECSP_DEBUGFILE	<p>This property specifies a path to the file to which logging output is to be written. Set this property if the JCECSP_DEBUG property is set to a value other than the default of 0. For details about the effects of setting different values for this property, see JCECSP_DEBUG property values.</p> <p>In a production environment, we recommend that you disable debug logging to prevent sensitive information being made available to an attacker.</p>
protect	This property specifies the type of protection to be used for key generation and nCipherKM KeyStore instances. You can set the value of this property to one of module , <SOFTCARD_NAME:SOFTCARD_IDENT> , or cardset . OCS protection (cardset) uses the card from the first slot of the first usable hardware security module. To find the logical token hash <IDENT> of a softcard, run the command nfkminfo --softcard-list .

Property	Function for different values
<code>module</code>	This property lets you override the default HSM and select a specific HSM to use for HSM and OCS protection. Set the value of this property as the ESN of the HSM you want to use.
<code>slot</code>	This property lets you override the default slot for OCS-protection and select a specific slot to use. Set this the value of this property as the number of the slot you want to use.
<code>ignorepass phrase</code>	If the value of this property is set to <code>true</code> , the nCipherKM provider ignores the pass phrase provided in its KeyStore implementation. This feature is included to allow the Oracle or IBM <code>keytool</code> utilities to be used with module-protected keys. The <code>keytool</code> utilities require a pass phrase be provided; setting this property allows a dummy pass phrase to be used.
<code>seeintegname</code>	Setting the value of this property to the name of an SEE integrity key causes the provider to generate SEE application keys. These keys may only be used by an SEE application signed with the named key.
<code>com.ncipher.provider.announcemode</code>	The default value for this property is <code>auto</code> , which uses firmware auto-detection to disable algorithms in the provider that cannot be supported across all installed HSMs. Setting the value of this property to <code>on</code> forces the provider to advertise all mechanisms at start-up. Setting the value of this property to <code>off</code> forces the provider to advertise no mechanisms at start-up.
<code>com.ncipher.provider.enable</code>	For the value of this property, you supply a comma-separated list of mechanism names that are to be forced on, regardless of the announce mode selected.
<code>com.ncipher.provider.disable</code>	For the value of this property, you supply a comma-separated list of mechanism names that are to be forced off, regardless of the announce mode selected. Any mechanism supplied in the value for the <code>com.ncipher.provider.disable</code> property overrides the same mechanism if it is supplied in the value for the <code>com.ncipher.provider.enable</code> property.

10.1.5.1. JCECSP_DEBUG property values

The `JCECSP_DEBUG` system property is a bit mask for which you can set different values to control the debugging functions. The following table describes the effects of different values that you can set for this property:

JCECSP_DEBUG value	Function
<code>0</code>	If this property has no bits set, no debugging information is reported. This is the default setting.
<code>1</code>	If this property has the bit 1 set, minimal debugging information (for example, version information and critical errors) is reported.

JCECSP_DEBUG value	Function
2	If this property has the bit 2 set, comprehensive debugging information is reported.
4	If this property has the bit 3 set, debugging information relating to creation and destruction of memory and HSM resources is reported.
8	If this property has the bit 4 set, <code>debugFunc</code> and <code>debugFuncEnd</code> generate debugging information for functions that call them.
16	If this property has the bit 5 set, <code>debugFunc</code> and <code>debugFuncEnd</code> display the values for all the arguments that are passed in to them.
32	If this property has the bit 6 set, context information is reported with each debugging message (for example, the <code>ThreadID</code> and the current time).
64	If this property has the bit 7 set, the time elapsed during each logged function is calculated, and information on the number of times a function is called and by which function it was called is reported.
128	If this property has the bit 8 set, debugging information for NFJAVA is reported in the debugging file.
256	If this property has the bit 9 set, the call stack is printed for every debug message.

To set multiple logging functions, add up the `JCECSP_DEBUG` values for the debugging functions you want to set, and specify the total as the value for `JCECSP_DEBUG`. For example, if you want to set the debugging to use both function tracing (bit 4) and function tracing with parameters (bit 5), add the `JCECSP_DEBUG` values shown in the table for these debugging functions ($8 + 16 = 24$) and specify this total (24) as the value to use for `JCECSP_DEBUG`.

10.1.6. Compatibility

The nCipherKM JCA/JCE CSP supports both module-protected keys and OCS-protected keys. The CSP currently supports 1/N OCSs and a single protection type for each nCipherKM JCE KeyStore.

You can use the nCipherKM JCA/JCE CSP with Security Worlds that comply with FIPS 140-2 at either Level 2 or Level 3.



In a Security World that complies with FIPS 140-2 Level 3, it is not possible to import keys generated by other JCE providers.

The nCipherKM JCA/JCE CSP supports load-sharing for keys that are stored in the nCipherKM KeyStore. This feature allows a server to spread the load of cryptographic operations across multiple connected HSMs, providing greater scalability.



We recommend that you use load-sharing unless you have existing code that is designed to run with multiple HSMs. To share keys with load-sharing, you must create a 1/N OCS with at least as many cards as you have HSMs. All the cards in the OCS must have the same pass phrase.



The nCipherKM JCA/JCE CSP does not support HSM Pool mode. If you want to use HSM Pool mode with a Java application that only uses module protected keys, one option may be to use the Sun PKCS #11 provider to access the nShield PKCS #11 library instead of using nCipherKM JCA/JCE CSP.

Keys generated or imported by the nCipherKM JCA/JCE CSP are not recorded into the Security World until:

1. The key is added to an nCipherKM KeyStore (by using a call to `setKeyEntry()` or `setCertificateEntry()`).
2. That nCipherKM KeyStore is then stored (by using a call to `store()`).

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

10.2. nShield PKCS #11 library

To use the nShield PKCS #11 library, you must tell the application the name and location of the library. The exact method for doing this depends on the application.

Instructions for using the nShield PKCS #11 library with specific applications are available from Entrust nShield Technical Support, <https://nshieldsupport.entrust.com>.

Depending on the application, you may need to set the path and library name `/opt/nfast/toolkits/pkcs11/libcknfast.so` in a dialog or configuration file.

The nShield PKCS #11 library has security options which you must configure before you use the PKCS #11 library. For more information, see [PKCS #11 library with Security Assurance Mechanism](#).

From version 1.7, the nShield PKCS #11 library can be used with FIPS 140-2 Level 3 compliant Security Worlds. This version of the library also introduces load-sharing mode. This feature provides support for multiple hardware security modules that are connected to a single server, spreading the load of cryptographic operations between the HSMs in order to provide scalability in terms of performance.

To share OCS protected keys with load-sharing mode, you must create a 1/*N* OCS that contains at least as many cards as you have HSMs. All the cards on the OCS must have the same passphrase.

With module firmware version 2.65.2 or later, if your application only uses module protected keys, you can use HSM Pool mode as an alternative to using load-sharing mode. HSM Pool mode supports returning or adding a hardware security module to the pool without restarting the system.



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

10.2.1. Choosing functions

Some PKCS #11 applications enable you to choose which functions you want to perform on the PKCS #11 token and which functions you want to perform in your application.

The following paragraphs in this section describe the functions that an nShield HSM can provide.

10.2.1.1. Generating random numbers and keys

The nShield HSM includes a hardware random number generator. A hardware random number generator provides greater security than the pseudo-random number generators provided by host computers. Therefore, always use the nShield HSM to generate random numbers and keys.

10.2.1.2. Digital signatures

The nShield PKCS #11 library can use the nShield HSM to sign and verify messages using the following algorithms:

- DSA
- RSA
- DES3_MAC
- AES
- ECDSA (if the appropriate feature is enabled)

An nShield hardware security module is specifically optimized for public key algorithms, and therefore it will provide significant acceleration for DSA, RSA and ECDSA signature generation and verification. You should always choose to perform asymmetric signature genera-

tion and verification with an nShield HSM.

10.2.1.3. Asymmetric encryption

The nShield PKCS #11 library can use an nShield HSM to perform asymmetric encryption and decryption with the RSA algorithm.

The nShield HSM is specifically optimized for asymmetric algorithms, so you should always choose to perform asymmetric operations with the nShield HSM.

10.2.1.4. Symmetric encryption

The nShield PKCS #11 library can use the nShield HSM to perform symmetric encryption with the following algorithms:

- DES
- Triple DES
- AES

Because of limitations on throughput, these operations can be slower on the nShield HSM than on the host computer. However, although the nShield HSM may be slower than the host under a light load, you may find that under a heavy load the advantage gained from off-loading the symmetric cryptography (which frees the host CPU for other tasks) means that you achieve better overall performance.

10.2.1.5. Message digest

The nShield PKCS #11 library can perform message digest operations with MD5, SHA-1, SHA-224, SHA-256, SHA-384, and SHA-512 algorithms. However, for reasons of throughput, the library performs these operations on the host computer.

10.2.1.6. Mechanisms

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

10.2.1.7. Key wrapping

The nShield PKCS #11 library can use an nShield HSM to wrap (encrypt) a private or secret key, or to unwrap (decrypt) a wrapped key.

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

10.2.2. PKCS #11 library with Security Assurance Mechanism

It is possible for an application to use the PKCS #11 API in ways that do not necessarily provide the expected security benefits, or which might introduce additional weaknesses. For example, the PKCS #11 standard requires the nShield library to be able to generate keys that are extractable from the HSM in plaintext. An application could use this ability in error, when a secure key would be more appropriate.

The PKCS #11 library with the Security Assurance Mechanism (SAM), **libcknfast**, can help users to identify potential weaknesses, and help developers create secure PKCS #11 applications more easily.

The SAM in the PKCS #11 library is intended to detect operations that reveal questionable behavior by the application. If these occur, the application fails with an explanation of the cause of failure.

After a review of your security policy and the way the application uses the PKCS #11 library with the SAM, if there are questionable operations that are considered to be acceptable and pose no security risk, the PKCS #11 library can be configured to permit some, or all, of them by means of the **CKNFAST_OVERRIDE_SECURITY_ASSURANCES** environment variable (described in **CKNFAST_OVERRIDE_SECURITY_ASSURANCES**).



To ensure the security of your keys, you must review any messages returned by the PKCS #11 library before changing the settings of the **CKNFAST_OVERRIDE_SECURITY_ASSURANCES** environment variable.

The **CKNFAST_OVERRIDE_SECURITY_ASSURANCES** environment variable uses a semicolon separated list of parameters, with associated values, to explicitly allow operations that could compromise the security of cryptographic keys if the operations are not well understood.

If no parameters, or the **none** parameter, are supplied to the **CKNFAST_OVERRIDE_SECURITY_ASSURANCES**, the PKCS #11 library fails to perform the operation in question, and issues a warning, when the following operations are detected:

- Creating short-term session keys as long-term objects
- Creating keys that can be exported as plain text
- Importing keys from external sources
- Creating or importing wrapping keys

- Creating or importing unwrapping keys
- Creating keys with weak algorithms (such as DES)
- Creating keys with short key lengths.

For more information about parameters and diagnostic warnings, see [CKNFAST_OVERRIDE_SECURITY_ASSURANCES](#).

10.2.2.1. Key security

Questionable operations largely relate to the concept of a key being *secure*. A private or secret key is considered insecure if there is some reason for believing that its value may be available outside the HSM. Public keys are never considered insecure; by definition they are intended to be public.

An explicitly insecure PKCS #11 key is one where [CKA_SENSITIVE](#) is set to false. If an application uses a key that is insecure but [CKA_SENSITIVE](#) is not set to false, it is possible that the application is using an inadequate concept of key security, and that the library disallows use of that key by default. Use of insecure keys should, by default, be restricted to short-term session keys, and applications should explicitly recognize the insecurity.

10.2.3. Using the nShield PKCS #11 library

After you have loaded the nShield PKCS #11 library, it is added to your application's list of cryptographic HSMs or PKCS #11 slots.

Whether or not the library uses load-sharing mode depends on the value of the [CKNFAST_LOADSHARING](#) environment variable, described in [CKNFAST_LOADSHARING](#). Whether or not the library uses HSM Pool mode depends on the value of the [CKNFAST_HSM_POOL](#) environment variable, described in [CKNFAST_HSM_POOL](#).

10.2.3.1. nShield PKCS #11 library with load-sharing mode

If load-sharing mode is enabled, the nShield PKCS #11 library creates a virtual slot for each OCS in the Security World (returning the name of the card set) unless you have set [CKNFAST_CARDSET_HASH](#) (as described in [CKNFAST_CARDSET_HASH](#)).

An additional virtual slot may be returned (with the label of [accelerator](#)), depending on the value given to the variable [CKNFAST_NO_ACCELERATOR_SLOTS](#) (described in [CKNFAST_NO_ACCELERATOR_SLOTS](#)). Accelerator slots can:

- Be used to support session objects

- Be used to create module-protected keys
- Not be used to create private objects.

When you insert a smart card from an OCS in the current Security World, the nShield PKCS #11 library treats this card as a PKCS #11 token that is present in the virtual slot for that OCS.

After the PKCS #11 token is present, you can open a session to that token. Until you log in, a session can only access public objects that belong to that PKCS #11 token.

The PKCS #11 token is present until you remove the last card belonging to the OCS. When you remove the token, the nShield PKCS #11 library closes any open sessions.

Logging in gives access to the private objects that are protected by the PKCS #11 token. Logging in requires the passphrase for the OCS. The exact mechanism for supplying the passphrase depends on the application that you are running.

The PKCS #11 token is shared across all the HSMs that have a smart card from the OCS in the reader at the point that you log in. After you have logged in, inserting additional cards from this OCS has no effect.

If you remove a smart card that belongs to a logged-in token, the nShield PKCS #11 library closes any open sessions and marks the token as being not present (unless the OCS is persistent). Removing a card from a persistent OCS has no effect, and the PKCS #11 token remains present until you log out.

10.2.3.2. nShield PKCS #11 library with HSM Pool mode

If HSM Pool mode is enabled, the nShield PKCS #11 library exposes a single pool of HSMs and a single virtual slot for a fixed token with the label **accelerator**. This accelerator slot can be used to create module protected keys and to support session objects. HSM Pool mode does not support token protected keys, any pre-existing OCS or softcard protected keys are hidden from PKCS #11. In FIPS 140-2 Level 3 Security Worlds, keys cannot be created in HSM Pool mode, however keys created outside HSM Pool mode, for example using **generatekey** or a non-Pool mode PKCS #11 application, can be used in HSM Pool mode.

10.2.3.3. nShield PKCS #11 library without load-sharing

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



The entry called **accelerator** cannot be used to create private objects. It can be used to create module-protected keys.

Use the second of the two entries (which has the same name as the Operator Card that is currently in a smart card reader) to protect your keys or token objects.

PKCS #11 does not allow two tokens to be present in the same slot. Therefore, when you insert a smart card into a reader, the nShield PKCS #11 library logs out any previously logged-in token from the slot and closes any open sessions.

10.2.3.4. nShield PKCS #11 library with the preload utility

You can use the **preload** command-line utility to preload *K/N* OCSs before actually using PKCS #11 applications. The **preload** utility loads the logical token and then passes it to the PKCS #11 utilities.

You must provide any required passphrase for the tokens when using **preload** to load the card set. However, because the application is not aware that the card set has been pre-loaded, the application operates normally when handling the login activity (including prompting for a passphrase), but the PKCS #11 library will not actually check the supplied passphrase. **preload** must be also used with the **cksotool** utility to perform operations that require the PKCS #11 Security Officer role.

Normally, **preload** uses environment variables to pass information to the program using the preloaded objects, including the PKCS #11 library. Therefore, if the application you are using is one that clears its environment before the PKCS #11 library is loaded, you must set the appropriate values in the **cknfastrc** file (see [nShield PKCS #11 library environment variables](#)). The current environment variables remain usable. The default setting for the **CKN-FAST_LOADSHARING** environment variable changes from specifying load-sharing as disabled to specifying load-sharing as enabled. Moreover, in load-sharing mode, the loaded card set is used to set the environment variable **CKNFAST_CARDSET_HASH** so that only the loaded card set is visible as a slot.

The **NFAST_NFKM_TOKENSFILE** environment variable must also be set in the **cknfastrc** file to the location of the preload file (see [Environment variables](#)).

A logical token preloaded by **preload** for use with the nShield PKCS #11 library is the only such token available to the application for the complete invocation of the library. You can use more than one HSM with the same card set.

If the loaded card set is non-persistent, then a card must be left in each HSM on which the set has been loaded during the start-up sequence. After a non-persistent card has been removed, the token is not present even if the card is reinserted.

If load-sharing has been specifically switched off, you see multiple slots with the same label.

10.2.4. nShield PKCS #11 library environment variables

The nShield PKCS #11 library uses the following environment variables:

- `CKNFAST_ASSUME_SINGLE_PROCESS`
- `CKNFAST_ASSURANCE_LOG`
- `CKNFAST_CARDSET_HASH`
- `CKNFAST_CONCATENATIONKDF_X963_COMPLIANCE`
- `CKNFAST_DEBUG`
- `CKNFAST_DEBUGDIR`
- `CKNFAST_DEBUGFILE`
- `CKNFAST_DH_LSB`
- `CKNFAST_FAKE_ACCELERATOR_LOGIN`
- `CKNFAST_HSM_POOL`
- `CKNFAST_LOADSHARING`
- `CKNFAST_LOAD_KEYS`
- `CKNFAST_NO_ACCELERATOR_SLOTS`
- `CKNFAST_NO_SYMMETRIC`
- `CKNFAST_NO_UNWRAP`
- `CKNFAST_NONREMOVABLE`
- `CKNFAST_NVRAM_KEY_STORAGE`
- `CKNFAST_OVERRIDE_SECURITY_ASSURANCES`
- `CKNFAST_RELOAD_KEYS`
- `CKNFAST_SEED_MAC_ZERO`
- `CKNFAST_SESSION_THREADSAFE`
- `CKNFAST_TOKENS_PERSISTENT`
- `CKNFAST_USE_THREAD_UPCALLS`
- `CKNFAST_WRITE_PROTECTED`

If you used the default values in the installation script, you should not need to change any of these environment variables.

You can set environment variables in the file `cknfastrc`. This file must be in the `/opt/nfast/` directory of the client.



The `cknfastrc` file should be saved without any suffix (such as `.txt`).

Each line of the file `cknfastrc` must be of the following form:

```
<variable>=<value>
```



Variables set in the environment are used in preference to those set in the resource file.

Changing the values of these variables after you start your application has no effect until you restart the application.

If the description of a variable does not explicitly state what values you can set, the values you set are normally **1** or **0**, **Y** or **N**.



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

10.2.4.1. CKNFAST_ASSUME_SINGLE_PROCESS

By default, this variable is set to **1**. This specifies that only token objects that are loaded at the time **C_Initialize** is called are visible.

Setting this variable to **0** means that token objects created in one process become visible in another process when it calls **C_FindObjects**. Existing objects are also checked for modification on disc; if the key file has been modified, then the key is reloaded. Calling **C_SetAttributeValues** or **C_GetAttributeValues** also checks whether the object to be changed has been modified in another process and reloads it to ensure the most recent copy is changed.

Setting the variable to **0** can slow the library down because of the additional checking needed if a large number of keys are being changed and a large number of existing objects must be reloaded.

10.2.4.2. CKNFAST_ASSURANCE_LOG

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

10.2.4.3. CKNFAST_CARDSET_HASH

This variable enables you to specify a specific card set to be used in load-sharing mode. If this variable is set, only the virtual smart card slot that matches the specified hash is present (plus the accelerator slot). The hash that you use to identify the card set in **CKN-**

FAST_CARDSET_HASH is the SHA-1 hash of the secret on the card. Use the **nfkminfo** command-line utility to identify this hash for the card set that you want to use: it is listed as **hk1tu**. For more information about using **nfkminfo**, see [nfkminfo: information utility](#).

10.2.4.4. CKNFAST_CONCATENATIONKDF_X963_COMPLIANCE

Sets the correct use of ECDH derive with concatenate KDF using the ANSI X9.63 specification as per the PKCS#11 standard.



The default is ANSI X9.63 to match that of the PKCS #11 Specification.



ECDH derive with concatenate KDF SP800-56a can use the standard PKCS #11 v3 **CKD_SHA[x]_SP800_KDF** values.

10.2.4.5. CKNFAST_DEBUG

This variable is set to enable PKCS #11 debugging. The values you can set are in the range **0** - **11**. If you are using **NFLOG_*** for debugging, you must set **CKNFAST_DEBUG** to **1**.

Value	Description
0	None (default setting)
1	Fatal error
2	General error
3	Fix-up error
4	Warnings
5	Application errors
6	Assumptions made by the nShield PKCS #11 library
7	API function calls
8	API return values
9	API function argument values
10	Details
11	Mutex locking detail

10.2.4.6. CKNFAST_DEBUGDIR

If this variable is set to the name of a writeable directory, log files are written to the speci-

fied directory. The name of each log file contains a process ID. This can make debugging easier for applications that fork a lot of child processes.

10.2.4.7. CKNFAST_DEBUGFILE

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

10.2.4.8. CKNFAST_DH_LSB

If this variable is set the least significant bytes of the result of DH/ECDH key agreement using the `CKM_DH_PKCS_DERIVE`, `CKM_X9_42_DH_DERIVE` or `CKM_ECDH1_DERIVE` mechanisms are taken. This is in line with the PKCS#11 specification. If this variable is not set the most significant bytes will be used. The latter behavior is consistent with Security World software prior to v12.81.

10.2.4.9. CKNFAST_FAKE_ACCELERATOR_LOGIN

If this variable is set, the nShield PKCS #11 library accepts a PIN for a module-protected key, as required by Sun Java Enterprise System (JES), but then discards it. This means that a Sun JES user requesting a certificate protected by a load-shared HSM can enter an arbitrary PIN and obtain the certificate.

10.2.4.10. CKNFAST_HSM_POOL

HSM Pool mode is determined by the state of the `CKNFAST_HSM_POOL` environment variable.

Set the environment variable to `1`, `y` or `Y` to enable HSM Pool mode for the PKCS #11 application, or set to `0`, `n` or `N` to explicitly disable HSM Pool mode for the PKCS #11 application.

HSM Pool mode takes precedence over load-sharing mode. HSM Pool mode only supports module protected keys so do not use `CKNFAST_NO_ACCELERATOR_SLOTS` to disable the accelerator slot.

10.2.4.11. CKNFAST_LOADSHARING

Load-sharing mode is determined by the state of the `CKNFAST_LOADSHARING` environment variable.

To enable load-sharing mode, set the environment variable `CKNFAST_LOADSHARING` to a value that starts with something other than `0`, `n`, or `N` and ensure that the `CKNFAST_HSM_POOL` environment variable is not set. The virtual slot behavior then operates.



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

10.2.4.12. CKNFAST_NO_ACCELERATOR_SLOTS

If this variable is set, the nShield PKCS #11 library does not create the accelerator slot, and thus the library only presents the smart card slots (real or virtual, depending on whether load-sharing is in use).

Do not set this environment variable if you want to use the accelerator slot to create or load module-protected keys.



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

10.2.4.13. CKNFAST_NO_SYMMETRIC

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

10.2.4.14. CKNFAST_NO_UNWRAP

If this variable is set, the nShield PKCS #11 library does not advertise the `c_wrap` and `c_unwrap` commands. You should set this variable if you are using Sun Java Enterprise System (JES) or Netscape Certificate Management Server as it ensures that a standard SSL handshake is carried out. If this variable is not set, Sun JES or Netscape Certificate Management Server make extra calls, which reduces the speed of the library.

10.2.4.15. CKNFAST_NONREMOVABLE

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



Since protection by non-persistent cards is enforced by the HSM, not the library, this variable does not make it possible to use keys after a non-persistent card is removed, or after a timeout expires.

10.2.4.16. CKNFAST_NVRAM_KEY_STORAGE

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

10.2.4.17. CKNFAST_OVERRIDE_SECURITY_ASSURANCES

This variable can be assigned one or more of the following parameters, with an associated value where appropriate, to override the specified security assurances in key operations where this is deemed acceptable:

- `all`
- `none`
- `tokenkeys`
- `longterm [=<days>]`
- `explicitness`
- `import`
- `unwrap_mech`
- `unwrap_kek`
- `derive_kek`
- `derive_xor`
- `derive_concatenate`
- `unwrap_rsa_aes_kwp`
- `weak_<algorithm>`
- `shortkey_<algorithm>=<bitlength>`
- `silent.`

Each parameter specified is separated by a semicolon. Using the command line, enter the following to set the variable:

```
CKNFAST_OVERRIDE_SECURITY_ASSURANCES="<parameter1>;<parameter2>=<value3>"
```

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

```
CKNFAST_OVERRIDE_SECURITY_ASSURANCES=<parameter1>;<parameter2>=<value3>
```

Unknown parameters generate a warning; see [Diagnostic warnings about questionable operations](#).

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

10.2.4.17.1. all

The **all** parameter overrides all security checks and has the same effect as supplying all the other **CKNFAST_OVERRIDE_SECURITY_ASSURANCES** parameters except the **none** parameter.

Using the **all** parameter prevents the library from performing any of the security checks and allows the library to perform potentially insecure operations. This parameter cannot be used with any other parameters.

10.2.4.17.2. none

The **none** parameter does not override any of the security checks and has the same effect as supplying no parameters. Using the **none** parameter allows the library to perform all security checks and warn about potentially insecure operations without performing them. This parameter cannot be used with any other parameters.

10.2.4.17.3. tokenkeys

The **tokenkeys** parameter permits applications to request that insecure keys are stored long-term by the cryptographic hardware and library.

Some PKCS #11 applications create short-term session keys as long-term objects in the cryptographic provider, for which strong protection by the HSM is not important. Therefore, provided that you intend to create long-term keys, the need to set this token does not always indicate a potential problem because the **longterm** keys restriction is triggered automatically. If you set the **tokenkeys** parameter, ensure that your Quality Assurance process tests all of your installation's functionality at least 48 hours after the system was set up to check that the key lifetimes are as expected.

When the **tokenkeys** parameter is set, the effect on the PKCS #11 library is to permit insecure Token keys. By default, any attempts to create, generate, or unwrap insecure keys with **CKA_TOKEN=true** fails with **CKR_TEMPLATE_INCONSISTENT** and a log message that explains the insecurity. When **tokenkeys** is included as a parameter for **CKNFAST_OVERRIDE_SECURITY_ASSURANCES**, attempts to create, generate, or unwrap insecure keys with **CKA_TOKEN=true** are allowed.

10.2.4.17.4. longterm[=days]

The **longterm** parameter permits an insecure key to be used for **days** after it was created. Usually insecure keys may not be used more than 48 hours after their creation. If **days** is not

specified, there is no time limit.



A need to set this variable usually means that some important keys that should be protected by the HSM's security are not secure.

When the **longterm** parameter is set, the PKCS #11 API permits the use of the following functions with an insecure key up to the specified number of **days** after its creation:

- **C_Sign** and **C_SignUpdate**
- **C_Verify** and **C_VerifyUpdate**
- **C_Encrypt** and **C_EncryptUpdate**
- **C_Decrypt** and **C_DecryptUpdate**.

By default these functions fail with **CKR_FUNCTION_FAILED**, or **CKR_KEY_FUNCTION_NOT_PERMITTED**, and a log message that explains the insecurity of these functions when used with an insecure private or secret key more than 48 hours after the creation of the key as indicated by **time()** on the host.

When the **longterm** parameter is set, the functions **C_SignInit**, **C_VerifyInit**, **C_EncryptInit**, and **C_DecryptInit** check the **CKA_CREATION_DATE** against the current time.

10.2.4.17.5. explicitness

The **explicitness** parameter permits applications to create insecure keys without explicitly recognizing that they are insecure. An insecure key is a key that is deemed sensitive, but can be wrapped and extracted from the HSM by any untrusted key. A secure key must have the **CKA_WRAP_WITH_TRUSTED** attribute.



A need to set the **explicitness** parameter does not necessarily indicate a problem, but does usually indicate that a review of the application's security policies and use of the PKCS #11 API should be carried out.

Unless the **explicitness** parameter is set, attempts to create, generate, or unwrap insecure keys with **CKA_SENSITIVE=true**, or to set **CKA_SENSITIVE=true** on an existing key, fail by default with **CKR_TEMPLATE_INCONSISTENT** and a log message explaining the insecurity. However, when the **explicitness** parameter is set, these operations are allowed.

10.2.4.17.6. import

The **import** parameter allows keys that are to be imported into the HSM's protection from insecure external sources to be treated as secure, provided that the application requests security for them. Usually, the library treats imported keys as insecure for the purposes of

checking the security policy of the application. Even though the imported copy may be secure, insecure copies of the key may still exist on the host and elsewhere.

If you are migrating from software storage to hardware protection of keys, you must enable the `import` parameter at the time of migration. You can disable `import` again after migrating the keys.



Setting this variable at any other time indicates that the library regards the key as secure, even though it is not always kept within a secure environment.

When the `import` parameter is set, the PKCS #11 API treats keys that are imported through `C_CreateObject` or `C_UnwrapKey` as secure (provided there is no other reason to treat them as insecure). By default, keys which are imported through `C_CreateObject` or `C_UnwrapKey` without this option in effect are marked as being insecure. Only the setting of the parameter at the time of import is relevant.

10.2.4.17.7. `unwrap_mech`

The `unwrap_mech` parameter allows you to create keys with `CKA_UNWRAP=true` and `CKA_DECRYPT=false`.

By default, when `unwrap_mech` is not supplied as a parameter for `CKNFAST_OVERRIDE_SECURITY_ASSURANCES`, trying to create a key with `CKA_UNWRAP=true` and `CKA_DECRYPT=false` fails with `CKR_TEMPLATE_INCONSISTENT`.

When `CKA_UNWRAP` is set to `true` and `CKA_DECRYPT` is not specified in the template, then `CKA_DECRYPT` is automatically set to `true`.

10.2.4.17.8. `unwrap_kek`

When a key is transferred into the HSM in encrypted form, the key is usually treated as insecure unless the key that was used for the decryption only allows the import and export of keys and not the decryption of arbitrary messages. This behavior is necessary to prevent an unauthorized application from simply decrypting the encrypted key instead of importing it. However, because PKCS #11 wrapping mechanisms are insecure, all unwrapping keys have `CKA_DECRYPT=true`.

By default, keys that are unwrapped with a key that has `CKA_DECRYPT` permission are considered insecure. When the `unwrap_kek` parameter is set, the PKCS #11 API considers keys that are unwrapped with a key that also has `CKA_DECRYPT` permission as secure (provided there is no other reason to treat them as insecure).

10.2.4.17.9. `derive_kek`

By default, keys that have been derived by using `CKM_DES3_ECB_ENCRYPT_DATA` with a key that has `CKA_ENCRYPT` permission are considered insecure. However, when the `derive_kek` parameter is set, the PKCS #11 API considers keys that are derived with a key that has `CKA_ENCRYPT` permission as secure (provided that there is no other reason to treat them as insecure).

10.2.4.17.10. `derive_xor`

Normally, you can only use only extractable keys with `CKM_XOR_BASE_AND_DATA` and, on unextractable keys, only `CKM_DES3_ECB_ENCRYPT_DATA` is allowed by `CKA_DERIVE`. However, when the `derive_xor` parameter is set, the PKCS #11 API also allows such functions with keys that are not extractable and treats them as secure (provided that there is no other reason to treat them as insecure).

10.2.4.17.11. `derive_concatenate`

Normally, you can only use session keys with `CKM_CONCATENATE_BASE_AND_KEY` for use with the operation `C_DeriveKey`. However, when the `derive_concatenate` parameter is set, the PKCS #11 API also allows such functions with keys that are long term (token) keys. The PKCS #11 API treats these keys as secure, provided there is no other reason to treat them as insecure. Even if the `all` parameter is set, if you do not include the `CKA_ALLOWED_MECHANISMS` with `CKM_CONCATENATE_BASE_AND_KEY`, this `C_DeriveKey` operation will not be allowed.

10.2.4.17.12. `unwrap_rsa_aes_kwp`

The `C_UnwrapKey` operation with `CKM_RSA_AES_KEY_WRAP` imports the temporary AES key with an nCore API ACL that permits unwrapping of the wrapped target key by the temporary AES key. When using the `C_UnwrapKey` operation with only a user supplied template (`pTemplate`) it is possible to create this ACL such that it permits a one-time unwrap of only the wrapped target key. When the RSA unwrapping key has `CKA_UNWRAP_TEMPLATE` set it is necessary to construct the ACL when the RSA key is created in order to setup the partitioning guarantees from the `CKA_UNWRAP_TEMPLATE`. The intended wrapped target keys are unknown at this time, which means the ACL must permit a one-time unwrap of any key.

The Security Assurance Mechanism (SAM) considers this scenario insecure by default and therefore the use of the `C_UnwrapKey` operation with `CKM_RSA_AES_KEY_WRAP` is disabled when the RSA unwrapping key has `CKA_UNWRAP_TEMPLATE` set. When the `unwrap_rsa_aes_kwp` parameter is set the SAM enables the `C_UnwrapKey` operation with `CKM_RSA_AES_KEY_WRAP` in this scenario. The RSA unwrapping key must also explicitly allow the `CKM_RSA_AES_KEY_WRAP`

mechanism via `CKA_ALLOWED_MECHANISMS` in addition to setting the `unwrap_rsa_aes_kwp` (or `all`) parameter; otherwise, the `C_UnwrapKey` operation will remain disabled when the RSA unwrapping key has `CKA_UNWRAP_TEMPLATE` set.

10.2.4.17.13. `weak_<algorithm>`

The `weak_<algorithm>` parameter allows you to treat keys used with a weak algorithm as secure. For example, DES is not secure, but setting the parameter `weak_des` means that such keys are considered secure. You can apply the `weak_<algorithm>` parameter to all keys that have a short fixed key length or whose algorithms have other security problems. As a guide, weak algorithms are those whose work factor to break is less than approximately 80 bits.

10.2.4.17.14. `shortkey_<algorithm=bitlength>`

The `shortkey_<algorithm=bitlength>` parameter permits excessively short keys for the specified `<algorithm>` to be treated as secure. The parameter `<bitlength>` specifies the minimum length, in bits, that is to be considered secure. For example, RSA keys must usually be at least 1024 bits long in order to be treated as secure, but `shortkey_rsa=768` would allow 768-bit RSA keys to be treated as secure.

10.2.4.17.15. `silent`

The `silent` parameter turns off the warning output. Checks are still performed and still return failures correctly according to the other variables that are set.

10.2.4.17.16. Diagnostic warnings about questionable operations

When the `CKNFAST_OVERRIDE_SECURITY_ASSURANCES` environment variable is set to a value other than `all`, diagnostic messages are always generated for questionable operations. Each message contains the following elements:

- The PKCS #11 label of the key, if available
- The PKCS #11 identifier of the key, if available
- The hash of the key
- A summary of the problem.

If the problem is not that a questionable operation has been permitted because of a setting in `CKNFAST_OVERRIDE_SECURITY_ASSURANCES` it could be that an operation has failed. In such a case, the setting required to authorize the operation is noted.

By default, these messages are sent to `stderr`. If a file name has been specified in the `CKNFAST_ASSURANCE_LOG` environment variable, diagnostic messages are also written to this file.

If `CKNFAST_DEBUG` is 1 or greater and a file is specified in `CKNFAST_DEBUGFILE`, the PKCS #11 library Security Assurance Mechanism log information is sent to the specified file. These variables must be set whenever `generatekey` or `KeySafe` are used.



If a file is specified in `CKNFAST_ASSURANCES_LOG` and no file is specified in `CKNFAST_DEBUGFILE` (or if `CKNFAST_DEBUG` is 0), diagnostic messages are sent to `stderr` as well as to the file specified in `CKNFAST_ASSURANCES_LOG`.

10.2.4.18. CKNFAST_SEED_MAC_ZERO

Set this variable to use zero padding for the Korean SEED MAC mechanisms (`CK_SEED_MAC` and `CKM_SEED_MAC_GENERAL`). If this variable is not set, or is set to `n`, then the SEED MAC mechanisms will use the default PKCS #5 padding scheme.

10.2.4.19. CKNFAST_SESSION_THREADSAFE

You must set this environment variable to `yes` if you are using the Sun PKCS #11 provider when running nCipherKM JCA/JCE code.

10.2.4.20. CKNFAST_TOKENS_PERSISTENT

This variable controls whether or not the Operator Cards that are created by your PKCS #11 application are persistent. If this variable is set when your application calls the PKCS #11 function that creates tokens, the Operator Card created is persistent.



Use of the nShield PKCS #11 library to create tokens is deprecated, because it can only create 1/1 tokens in FIPS 140-2 Level 2 Security Worlds. Use `KeySafe` or one of the command-line utilities to create OCSs.

10.2.4.21. CKNFAST_USE_THREAD_UPCALLS

If this variable is set and `CKF_OS_LOCKING_OK` is passed to `C_Initialize`, `NFastApp_SetThreadUpcalls` is called by means of `nfast_usencthread`s and only a single `NFastApp_Connection` is used, shared between all threads.

If this variable is set and mutex callbacks are passed to `C_Initialize` but `CKF_OS_LOCKING_OK` is not passed, `C_Initialize` fails with `CKR_FUNCTION_FAILED`. (`NFastApp_SetThreadUp-`

`calls` requires more callbacks than just the mutex ones that PKCS #11 supports.)

If neither mutex callbacks nor `CKF_OS_LOCKING_OK` is passed, this variable is ignored. Only a single connection is used because the application must be single threaded in this case.

10.2.4.22. CKNFAST_LOAD_KEYS

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

10.2.4.23. CKNFAST_WRITE_PROTECTED

Set this variable to make your OCS or softcard (token) write-protected. If a token is write-protected, you cannot:

- Generate certificate, data, and key objects for that token.
- Modify attributes of an existing object.



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

10.2.4.24. CKNFAST_RELOAD_KEYS

Set this variable to enable PKCS #11 key reloading. See section *PKCS #11 with key reloading* in the *Cryptographic API Integration Guide*.

Key reloading requires load sharing-mode to operate, and enables it automatically if `CKNFAST_LOADSHARING` is not set.

10.2.5. Checking the installation of the nShield PKCS #11 library

After you have created a Security World, ensure that the nShield PKCS #11 library has been successfully installed by using the `ckcheckinst` command-line utility.

To verify the installation of the nShield PKCS #11 library, follow these steps:

1. Give the command `ckcheckinst`.

If you have an invalid Security World (for example, if all your HSMs are in the initialization state), `ckcheckinst` quits with the following error message:

```
ckcheckinst: C_Initialize failed rv = 00000006
```

Is the security world initialized? (Use nfkminfo to check)

If your Security World is valid, `ckcheckinst` displays information similar to the following:

```
PKCS#11 library interface version 2.40
flags 0
manufacturerID "nCipher Corp. Ltd "
libraryDescription "nFast PKCS#11 1.## "
implementation version 1.##
Load sharing and Failover enabled

slot Status Label
===== 0 Fixed token "accelerator "
1 Operator card "card2 "
2 Operator card "card3 "
Select slot Number to run library test or 'R'etry or to 'E'xit:
```

In this example output:

- `PKCS #11 library interface version 2.40` refers to the version of the PKCS #11 specification supported
- `implementation version 1.##` refers to the version of the nCipher PKCS #11 library
- `Loadsharing and Failover enabled` is shown if load-sharing has been enabled. Alternatively `Pool mode enabled` is shown if Pool mode has been enabled.

Slots that contain a valid Operator Card are indicated by the status `Operator card` and the card's label. A fixed token is always available and is listed as slot 0.

If you insert a blank card or an unrecognized card (for example, an Operator Card from a different Security World or an Administrator Card), this is indicated in the `Status` column. The corresponding slot number is not available.



If you are using the `preload` command-line utility in conjunction with the nShield PKCS #11 library, you can only see the token that you loaded with the `preload` utility. In load-sharing mode, the loaded card set is used to set the environment variable `CKNFAST_CARDSET_HASH`, so only this card set is visible as a slot.

If there is no card in a slot, `ckcheckinst` displays `No token present` beside the relevant slot numbers.

`ckcheckinst` gives you the following choices:

```
No removable tokens present.
Please insert an operator card into at least one available slot and
enter 'R' retry.
If you have not created an operator card or there are no physical slots, enter a fixed token slot number,
```

or 'E' to exit this program and create a card set before continuing.

2. If there are no available slots with cards in them, you can choose one of the following actions:

- Insert a valid Operator Card, and press **R**
- choose a fixed token slot
- Press **E** to quit, then create an OCS, and run **ckcheckinst** again.

When there is at least one slot with a valid token, input a slot number, and press **Enter**. In a FIPS 140-2 Level 3 compliant Security World, **ckcheckinst** prompts you to enter the passphrase for the selected Operator Card.

3. Type the passphrase, and press **Enter**.

ckcheckinst displays the results of the tests:

```
Test Pass/Failed
-----
1 Generate RSA key pair Pass
2 Generate DSA key pair Pass
3 Encryption/Decryption Pass
4 Signing/Verify Pass
Deleted test keys ok
PKCS11 Library test successful.
```

If any tests fail, **ckcheckinst** displays a message indicating the failure and quits. It does not run any subsequent tests.

If **ckcheckinst** fails:

- Check that the hardserver is running
- Use the **enquiry** and **nfkminfo** world.

If all seems in order, reinstall the nShield library.

10.2.6. How the nShield PKCS #11 library protects keys

Session objects are created on an HSM and never leave that HSM. The following table lists the protection for different types of PKCS #11 token objects:

	Smart card Slot	Accelerator Slot
Private Token Object	Operator Card Set	not supported
Public Token Object	Security World	Security World
Public key	well known HSM key	well known HSM key

Operator Card Set

The object is stored as an nShield key blob encrypted by the OCS key. You must log in to this OCS before you can load this object.

security world

The object is stored as an nShield key blob encrypted by the Security World key. This object can be loaded on to any HSM in the Security World. The nShield PKCS #11 library only allows access if a card from this OCS is present.

well-known module key

Public keys are encrypted under a well-known HSM key. This encryption is for programming convenience only and does not provide security. These keys can be loaded on any nShield HSM.

10.3. nShield native and custom applications

Use the nShield native option for applications that were written using nShield key management software and that expect keys to be both protected by the Security World and stored in the Security World data structure.

Use the **custom** external application option for applications that were written using nShield key management software and that expect their keys to be in standalone files.



KeySafe does not place any restrictions on the OCS that is used to protect nShield native or **custom** application keys. You must make sure that your application is capable of loading the card set.

10.4. CodeSafe applications

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



If you wish to use the SEE to run applications, it must have been ordered and enabled as described in [Enabling optional features](#).

An SEE application is typically a standalone SEE machine that is loaded automatically by the hardserver (for example, a CodeSafe C application).

Check the documentation that your application vendor supplies for information about any signatures that you may require to set up and use the application, as well as for any other

installation and configuration information.

CodeSafe applications are standalone applications, but each CodeSafe C application can consist of multiple parts, and its installation can include several configuration steps. For instructions on installing and configuring each application, see your application vendor's documentation.

To use a standalone application:

1. Ensure that the SEE machine for the application is in the directory `/opt/nfast/ custom-seemachines` on the remote file system.



If an SEE machine has previously been loaded on the HSM, press the Clear button on the front of the unit before proceeding to the next step. This clears the current SEE machine from memory.

2. From the main menu on the front panel of the HSM, select **CodeSafe**.
3. To enable the HSM to publish the SEE World for multiple clients, enter the following information when prompted:
 - The name of the SEE machine file.
 - The name of the user data file, if required.
 - The type of custom SEE machine you are using (select **SEELib** or **BSDlib sockserv**).



This option is only available if you have provided a valid user data file in step 2. If **BSDlib sockserv** is selected, `worldid_pubname`, `postload_prog`, and `postload_args` will be passed to `load _seemachine`. For detailed descriptions of the options in this section, see [load_seemachine](#).

- The ID of the SEE World to create.



This option is only available if you have selected the **SEELib** option in the previous step.



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

10.5. Remotely loading and updating SEE machines

The SEE remote push facility allows the remote deployment of CodeSafe SEE machines to an Connect, negating the need to physically visit the HSM to load or update the SEE machine. This is achieved by editing the configuration file on the RFS for a specific Connect

to specify the new SEE machine, then setting a configuration flag in the config file to **true**.

Before configuring a module to autonomously run an SEE machine and accept updates using the RFS, that module must first be set up to accept remotely-pushed configurations, see [Configuring auto push](#).

For more information about configuring log file storage options, see [Configuring log file storage](#).

To configure an Connect module to autonomously run an SEE machine and accept updates using the RFS:

1. Copy the existing config file to a new file called **config.new**.
2. In the **load_seemachine** section of the **config.new** file for the remote module, add or amend the following settings:

```
pull_rfs=true
machine_file=mymachinename.sar
userdata=myuserdata.sar
worldid_pubname=publ_name
```



These settings specify the type, name and user data of the SEE machine you wish to load. For more information about each setting, see [load_seemachine](#).



For CodeSafe Direct, the **userdata** file must be packed as a SAR file.



The remote module will load the new SEE machine in place of any existing SEE machine. If no **machine_file** value is set, then pushing the config file will remove any existing machines on the unit.

3. In the **sys_log** section of the **config.new** file for the remote module, add or amend the following settings:

```
behaviour=push
push_interval=1
```



These settings control how and where log messages are written. Using the example above, messages will be written to the **sys-tem.log** and **hardserver.log** files of the module, which are accessible using the remote file system. You may wish to revise the **push_interval** to a higher value once the Connect has successfully loaded the new SEE machine.

4. Run `nopclearfail` to clear the module, followed by `enquiry` to check that the module is ready.
5. Run `cfg-pushnethsm` to push the new config file to the module.

To load a new SEE machine to multiple Connect modules, we recommend scheduling down time for each HSM, upgrading them on a per HSM basis. Each Connect configuration file is specific to an individual HSM and each configuration file should be updated separately to load the new SEE machine.

11. Remote Operator

This chapter explains:

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



If you wish to use the Remote Operator feature, you must have enabled it as described in [Enabling optional features](#). The Remote Operator feature must have been ordered for, and enabled on, the nShield module that you intend to use as the remote, unattended module.

11.1. About Remote Operator

The Remote Operator feature enables the contents of a smart card inserted into the slot of one module (the *attended module*) to be securely transmitted and loaded onto another module (an *unattended module*). This is useful when you need to load an OCS-protected key onto a machine to which you do not have physical access (because, for example, it is in a secure area).

For Remote Operator to work, the modules must be in the same Security World. You insert the required cards from the OCS into a slot in the attended module. From this module, the contents of the OCS are transmitted over secure channels to the unattended module, which then loads them. You do not need physical access to the unattended module in order to load the OCS onto it.

The following limitations apply to Remote Operator:

- You cannot access non-persistent card sets remotely
- You cannot use the `createocs` command-line utility to write new cards or card sets remotely.

You can export a slot from an attended module and import a slot to any (unattended) module in the Security World. Before you can import a slot to one module, you must first export it from another module.

11.2. Configuring Remote Operator

This section explains how to configure Remote Operator.

11.2.1. Overview of configuring Remote Operator

Before you can use Remote Operator, you must perform the following initial configuration tasks:

1. Configure the HSMs for Remote Operator.

The HSMs must be in the same Security World, and must have been initialized with remote card set reading enabled.

Both the attended and the unattended HSM must be in operational mode before they can import or export slots. See [Checking and changing the mode on an nShield Connect](#) for more about changing the mode.

2. Configure the HSMs for slot import and export, as appropriate.

Starting from 12.81, you can export and import dynamic slots as Remote Operator slots.

After the initial configuration is complete, to use Remote Operator you must:

1. Create a Remote OCS (that is, an OCS with the correct permissions for Remote Operator).
2. Generate keys that are protected by the Remote OCS.
3. Ensure your application is configured to use keys protected by the Remote OCS.

11.2.2. Configuring HSMs for Remote Operator

1. Ensure both HSMs are initialized into the same Security World; see [Adding or restoring an HSM to the Security World](#).



By default, HSMs are initialized with remote card-set reading enabled. If you do not want an HSM to be able to read remote card sets, you can initialize it by running the `new-world` with the `-S MODULE` (where `MODULE` is the HSM's ID number).

2. For the unattended HSM:
 - a. Check whether the Remote Operator feature is enabled by running the `enquiry` command-line utility. The output for the HSM must include `Remote Share` in its list of Features.
 - b. Check whether the HSM has permission to allow loading of Remote OCSs by selecting **Security World mgmt > Display World info**.
 - c. Check whether the correct software, with permission to receive remote shares, is present by running the `nfkminfo` command-line utility.

The output from this selection must show that **flags** are set to include **ShareTarget**, as in the following example:

```
Module #1
generation 2
state 0x2 Usable
flags 0x10000 ShareTarget
n_slots 3
esn 8851-43DF-3795
hkm1 391eb12cf98c112094c1d3ca06c54bfe3c07a103
```

11.2.3. Configuring slot import and export

For information about the parameters controlled by the hardserver configuration file, see:

- [slot_exports](#)
- [slot_imports](#)
- [slot_mapping](#)

Before you can configure hardservers for Remote Operator, ensure that:

- You have configured the attended and unattended HSMs for Remote Operator as described in [Configuring HSMs for Remote Operator](#).
- Your network firewall settings are correct. See the *Installation Guide* for more information about firewall settings.

When the HSMs have been configured, use one of the following methods to configure slot import and export:

- Use the Connect front panel, see [Configuring slot import and export using the Connect front panel](#).
- Use the **cfg-remoteslots** utility.
- Update the HSM configuration file, see [Configuring hardservers for Remote Operator using the HSM configuration file](#).

11.2.3.1. Configuring slot import and export using the Connect front panel

1. Configure the attended HSM to export a slot by following these steps:
 - a. From the main menu, select **Security World mgmt > Set up remote slots > Export slot**.

Use this option for exporting slot #0 only.

If you need to configure the export of slots other than 0, see [Configuring hard-](#)

[servers for Remote Operator using the HSM configuration file.](#)

- b. Specify the HSM to which the slot is being export by supplying values for:
 - The IP address of the unattended HSM
 - The ESN of the unattended HSM.
2. Configure the unattended HSM to import the slot that you are exporting from the attended HSM by following these steps:
 - a. From the main menu, select **Security World mgmt > Set up remote slots > Import slot**.
 - b. Specify the details of the Remote Operator slot by supplying values for:
 - The IP address of the HSM from which the slot is being exported
 - The ESN of the HSM from which the slot is being exported
 - The ID of the slot on the importing HSM
 - The port to use to connect to the hardserver hosting the attended HSM.

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

```
slotinfo -m 1
```

If slot importation was successful, the output from this command includes the line:

Slot	Type	Token	IC	Flags	Details
#0	Smartcard	present	3	A	
#1	Software Tkn	-	0		
#2	Smartcard	-	0	AR	

The **R** in the **Flags** column indicates that slot **2** is a Remote Operator slot.

Applications running on the unattended machine can now use slot **2** to load OCSs that are presented to slot **0** on the attended machine. If any of the cards require a pass phrase, the application must pass this to the unattended HSM in the usual way.

For the application to be able to load the OCS onto the unattended HSM, it must be able to read the card set files associated with the OCS from the local Key Management Data directory. If the OCS was created on a different machine, you must copy the card set files in the Key Management Data directory onto the unattended machine (either manually or by using client cooperation; for more information, see [Setting up client cooperation](#)).

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

11.2.3.2. Configuring hardservers for Remote Operator using the HSM configuration file

1. On the attended HSM's host machine, configure the hardserver to allow slot 0 of the local HSM (with ESN AAAA-AAAA-AAAA to be exported to a remote HSM (with ESN BBBB-BBBB-BBBB, hosted by the machine with the IP address 222.222.222.222):
 - a. Create a copy of the configuration file as **config.new** in the following directory.

```
/opt/nfast/kmdata/hsm-ESN/config
```

- b. Edit the sections related to slot export in **config.new**:

```
[slot_exports]
# Start of the slot_exports section
# Local slots that the hardserver should allow remote modules to import. Note
# that if a slot which has been remapped in the slot_mapping section is to be
# exported, it must be referred to in this section by its original
# (pre-mapping) local_slotid.
# Each entry has the following fields:
#
# ESN of the local module whose slot is allowed to be exported.
# local_esn=ESN
#
# SlotID of the slot which is allowed to be exported. (default=0)
# local_slotid=INT
#
# IP address of the machine allowed to import the slot or empty to allow all
# machines. (which is the default)
# remote_ip=ADDR
#
# ESN of the module allowed to import the slot or "" to allow all modules
# which are permitted in the security world. (default = "")
# remote_esn=ESN
```

```
[slot_mapping]
# Start of the slot_mapping section
# Slot remapping configuration. Notes for Remote Operator users: If a slot
# which is remapped in this section is also exported in the slot_exports
# section, the local_slotid field in the slot_exports section must be set to
# the original (pre-mapping) local_slotid. When importing that slot in another
# module, the slot_imports section must refer instead to the new
# (post-mapping) remote_slotid.
# Each entry has the following fields:
#
# ESN of the module on which slot 0 will be remapped with another.
# esn=ESN
#
# Slot to exchange with slot 0. Setting this value to 0 means do
# nothing. (default=0)
# slot=INT
```

- c. Run the **cfg-pushnethsm** utility on the updated configuration file, specifying the updated file and the network address of the nShield Connect to load the new configuration.


```
cfg-pushnethsm --address=<module_address> <path_to_config_file>
```

- d. Check that the configuration file has been updated. This can be confirmed using the timestamp on the updated config file.
 - e. Clear the HSM for the changes to take effect, run the **nopclearfail** command:
2. On the unattended module's host machine, configure the hardserver to import slot 0 from the remote attended module (with ESN AAAA-AAAA-AAAA, hosted by the machine with the IP address 111.111.111.111) to the local module (with ESN BBBB-BBB-BBBB).
- a. Edit the sections related to slot import in **config.new**:

```
[slot_imports]
# Start of the slot_imports section
# Remote slots that the hardserver should import to modules on this machine.
# Note that if a remote slot which has been remapped in the slot_mapping
# section on the remote system is to be imported, it must be referred to in
# this section by its new (post-mapping) remote_slotid.
# Each entry has the following fields:
#
# ESN of the local module to import the slot to
# local_esn=ESN
#
# SlotID to use to refer to the slot when it is imported on the local module.
# Setting this value to 0 means it will be automatically assigned to the
# lowest available value. (default=0)
# local_slotid=INT
#
# IP address of the machine hosting the slot to import
# remote_ip=ADDR
#
# Port to connect to on the remote machine
# remote_port=PORT
#
# ESN of the remote module to import the slot from
# remote_esn=ESN
#
# SlotID of the slot to import on the remote module (default=0)
# remote_slotid=INT
```

- b. Run the **cfg-pushnethsm** utility on the updated configuration file, specifying the updated file and the network address of the nShield Connect to load the new configuration.

```
cfg-pushnethsm --address=<module_address> <path_to_config_file>
```

- c. Check that the configuration file has been updated. This can be confirmed using the timestamp on the updated config file.
 - d. Clear the HSM for the changes to take effect, run the **nopclearfail** command:
3. Check the Remote Operator slot configuration:

```
slotinfo -m 1
```

If slot import was successful, the output from this command includes the line:

Slot	Type	Token	IC	Flags	Details
#0	Smartcard	present	3	A	
#1	Software Tkn	-	0		
#2	Smartcard	-	0	AR	

The **R** in the **Flags** column indicates that slot **2** is a Remote Operator slot.

Applications running on the unattended machine can now use slot **2** to load OCSs that are presented to slot **0** on the attended machine. If any of the cards require a pass phrase, the application must pass this to the unattended HSM in the usual way.

For the application to be able to load the OCS onto the unattended HSM, it must be able to read the card set files associated with the OCS from the local Key Management Data directory. If the OCS was created on a different machine, you must copy the card set files in the Key Management Data directory onto the unattended machine (either manually or by using client cooperation; for more information, see [Setting up client cooperation](#)).

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

11.2.4. Using Remote Operator with applications requiring cards in slot 0

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

1. Do one of the following:

- a. Use the **slot_mapping** section in the module configuration file to define a Dynamic Slot to exchange with slot 0 for an HSM and push the updated configuration file to the Connect.

See [nShield Connect and client configuration files](#) for more about module configuration file, [slot_mapping](#) for more about the **slot_mapping** section and `xref:configure.adoc#AboutUserPrivileges,About user privileges` > > for more about editing the module configuration file.

Or:

- b. Use the front panel controls to navigate to **Security World mgmt > Set up dynamic slots > Slot mapping** and follow the instructions on the screen.

You can check the mapping by:

- Running the command:

```
slotinfo -m 1
```

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

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

- The **R** in the **Flags** column indicates that slot #0 is now a Remote Slot



Slot mapping can also be configured for a dynamic remote slot, i.e. a dynamic slot in a different HSM which has been imported to the relevant HSM. The Flags column will contain the flags ARD.

or:

- Using the front panel controls to navigate to **Security World mgmt > Display World**.

When dynamic slots are added to an HSM after the initial configuration was done with only remote slots, the dynamic slots will take precedence over the remote slots. The slot numbers of the remote slots will therefore change. You will have to revise the slot mapping and specify the new slot number of the remote slot.

11.2.5. Using Remote Operator on Remapped Slots

If a slot has been mapped to slot #0 on the attended HSM, it is still possible to export the local slot to an unattended HSM. Further, if the mapped slot is a dynamic slot, it is possible to export it as well. To do this, do the following:

1. On the attended HSM's host machine, configure the hardserver to allow the export of the relevant slot by referring to it by its original slotID.
 - a. To export the local slot, local_slotid=0.
 - b. To export a dynamic slot, local_slotid=2 (or higher if the HSM is configured with multiple dynamic slots).
2. On the unattended HSM's host machine, configure the hardserver to import the relevant slot by referring to it by its new slotID.
 - a. To import the exported local slot, remote_slotid=2 (or higher, same as the slotID

specified in the mapping section of the attended HSM's configuration file).

- b. To import the exported dynamic slot, `remote_slotid=0`.

11.2.6. Configuration Example for Using Remote Administration and Remote Operator Concurrently

Below is an example of the relevant portions of a `hardserver` config file to achieve concurrent usage of Remote Administration and Remote Operator. It is broken up and explained per config file section.

The `dynamic_slots` section allocates exactly 1 dynamic slot to each of modules 1 and 2.

```
[dynamic_slots]
esn=BBBB-BBBB-BBBB
slotcount=1
-----
esn=AAAA-AAAA-AAAA
slotcount=1
```

The `slot_imports` section first imports module 1 slot #0 to module 2 slot #3 and then imports module 1 slot #2 to module 2 slot #4.

```
[slot_imports]
local_esn=AAAA-AAAA-AAAA
remote_ip=127.0.0.1
remote_port=9004
remote_esn=BBBB-BBBB-BBBB
remote_slotid=0
-----
local_esn=AAAA-AAAA-AAAA
remote_ip=127.0.0.1
remote_port=9004
remote_esn=BBBB-BBBB-BBBB
remote_slotid=2
```

The `slot_exports` section allows module 1 slot #0 and module 1 slot #2 to be exported by that module.

```
[slot_exports]
local_esn=BBBB-BBBB-BBBB
local_slotid=0
-----
local_esn=BBBB-BBBB-BBBB
local_slotid=2
```

The `slot_mapping` section swaps module 2 slot #0 and module 2 slot #2.

```
[slot_mapping]
esn=AAAA-AAAA-AAAA
```

```
slot=2
```

After making the changes above to the hardserver configuration file:

1. Push the hardserver configuration file to the nShield Connect by running `cfg-push-nethsm`.
2. Clear the modules by running `nopclearfail`.

This is the expected system configuration output for the relevant modules:

```
slotinfo -m1
Slot Type      Token  IC   Flags  Details
#0  Smartcard   -      0     A
#1  Software Tkn -      0
#2  Smartcard   -      0     AD

slotinfo -m2
Slot Type      Token  IC   Flags  Details
#0  Smartcard   -      0     AD
#1  Software Tkn -      0
#2  Smartcard   -      0     A
#3  Smartcard   -      0     AR
#4  Smartcard   -      0     ARD
```

11.2.7. Using Remote Operator with Remote Administration with Older Versions of the Software

Versions of Remote Operator older than 12.81 do not support its concurrent use with the Remote Administrator feature. In such a case, the following features are not supported:

- Exporting and importing dynamic slots
- Mapping remote slots to slot #0
- Automatic assignment of slotID when importing slots

It is possible to use some of the features when the attended HSM (exporting end) has the new version of the software (12.81+) and the unattended HSM (importing end) has an older version (pre-12.81).

A dynamic slot which has been exported by the attended HSM can be imported to the unattended HSM. Its local slotID will need to be manually specified if the unattended HSM has any dynamic slots configured. This is due to the default import slot (slot #2) being occupied by the dynamic slot. The unattended HSM can remap its dynamic slots to slot #0, but cannot remap any of its imported slots.

11.3. Creating OCSs and keys for Remote Operator

When you have configured the HSMs and slot import and export, you can create Remote OCSs and generate keys protected by them. These Remote OCSs and keys can be used by applications running on the unattended HSM.

For the most part, card sets and keys intended to be used with Remote Operator are similar to their ordinary, non-Remote counterparts.

11.3.1. Creating OCSs for use with Remote Operator

You can generate Remote OCSs by using KeySafe or by running the `createocs` command-line utility with the `-q|--remotely_readable` option specified. The cards in a Remote OCS must be created as persistent; see [Persistent Operator Card Sets](#).

To check whether the card in a slot is from a Remote OCS, select **Security World mgmt > Display World info** from the main menu or run the `nfkminfo` command-line utility. The output displays slot section information similar to the following:

```
Module #1 Slot #0 IC 1
generation      1
phystype        SmartCard
slotlistflags   0x2
state           0x5
Operator flags  0x20000 RemoteEnabled
shareno         1
shares          LTU(Remote)
error           OK
```

In this example output, the `RemoteEnabled` flag indicates the card in the slot is from a Remote OCS.



If you create a Remote OCS on the attended machine, then you must copy the Key Management Data files on the attended machine to the unattended machine.



Both the attended and unattended HSMs must be in the same Security World before you generate a Remote OCS. If you are not using client cooperation, the Key Management Data directories must be manually synchronized after you generate the Remote OCS.



If you already have recoverable keys protected by a non-Remote OCS, you can transfer them to a new Remote OCS by using KeySafe or the `replaceocs` command-line utility.

11.3.2. Loading Remote Operator Card Sets

Once configured, the Remote Operator slots can be used by all the standard nShield libraries. A Remote Operator slot can be used to load any OCSs that have been created to allow remote loading. For more information about the applications to use with remote cards, see [Application interfaces](#). For more information about Remote Operator slots, see [Remote Operator](#).



After an OCS has been inserted into a Remote Operator slot, for each time a given card is inserted, the module only allows each share on that card to be read one time. If there is a second attempt to read shares from that card before the card is reinserted, the operation fails with a `UseLimitsUnavailable` error.

11.3.3. Generating keys for use with Remote Operator

After you have created a Remote OCS, to generate keys protected by it you can run KeySafe or the `generatekey` and `preload` command-line utilities on the unattended module, inserting cards to the slot attached to the attended module. For more information about generating and working with keys, see [Working with keys](#).



If you generate keys protected by a Remote OCS on the attended module, then you must copy the files in the Key Management Data directory on the attended machine to the unattended module.



KeySafe can list imported slots, but cannot use them.

If you already have an OCS-protected key that you want to use, but the protecting OCS is not a Remote OCS, you can use KeySafe to protect the key under a new Remote OCS if the key was originally generated with the key recovery option enabled.

However, if the key was not generated with key recovery enabled, you cannot protect it under a different OCS. In such a case, you must generate a new key to be protected by a Remote OCS.

11.3.4. Configuring the application

After you have configured the HSMs and slot import and export, created a Remote OCS, and generated keys protected by the Remote OCS, configure the application with which you want to use these keys as appropriate for the particular application.

After you have configured the application, start it remotely from the attended machine. Insert cards from the OCS into the attended machine's exported slot as prompted.

11.3.5. Managing Remote Operator slots using the unit front panel

11.3.5.1. Editing Remote Operator slots

You can change the details of a Remote Operator slot. You must always update the details of both the exported slot on the local module and the imported slot on the remote module.

To update an exported a slot on the module:

1. From the main menu, select **Security World mgmt > Set up remote slots > Edit exported slot**.
2. Select the exported slot that you want to update. Slots are identified by the IP address of the remote module.
3. Update the details of the slot.

To update an imported slot on the unit:

1. From the main menu, select **Security World mgmt > Set up remote slots > Edit imported slot**.
2. Select the imported slot that you want to update. Slots are identified by the IP address of the remote module.
3. Update the details of the slot.

11.3.5.2. Deleting Remote Operator slots

You can delete Remote Operator slots.

To delete an exported slot, from the main menu, select **Security World mgmt > Set up remote slots > Delete exported slot** and select the slot you want to delete.

To delete an imported slot, from the main menu, select **Security World mgmt > Set up remote slots > Delete imported slot** and select the slot you want to delete.

12. Working with keys

This chapter explains how to use the facilities we provide to work with keys. There is often more than one way of performing a particular task. The methods available for working with keys are:

- KeySafe
- **generatekey** and related utilities
- The unit front panel.

You cannot generate keys from the front panel on the unit. You can generate keys on the client using the methods described in this chapter and view them on the module.

12.1. Common Criteria CMTS Mode Assigned Keys

Common Criteria CMTS mode includes the concepts of Assigned Keys and General Keys, as defined in EN 419 221-5.

Assigned Keys provide for more restrictive controls which are enforced with ACLs. An Assigned Key is a secret key with a Key Generation Certificate and with the ACL configuration defined in *nShield Solo XC Common Criteria Evaluated Configuration Guide*, specifically:

- The **Reauthorization conditions** and **Key Usage** attributes cannot be changed.
- The **Authorisation Data** attribute can only be changed by presentation of the current **Authorisation Data**, it cannot be changed or reset by an Administrator.
- The key cannot be exported by wrapping with another key.
- The key must be generated. It cannot be imported.

These properties of an Assigned Key enable the sole control that's required for a secret key used to create a digital signature.

A General Key is one that does not meet the criteria for an Assigned Key.

For both Assigned and General Keys in a Common Criteria CMTS Security World it is not possible to export or import as plain text. This is enforced by the HSM.

The ACL configuration defining an Assigned Key is described in the *nShield Solo XC Common Criteria Evaluated Configuration Guide*. Determination of the Assigned status of a key uses the **nfkverify** utility and the Key Generation certificate recorded in the key when it was created.

The `generatekey` and `mkac1x` utilities have been enhanced to offer support for generating Assigned Keys, see [Key generation options and parameters](#) for `generatekey` and the online help for `mkac1x`.

12.2. Generating keys

Whenever possible, generate a new key instead of importing an existing key. Because existing keys have been stored in a known format on your hard disk, there is a risk that the existing key has been compromised. Key material can also persist on backup media.



Some applications can generate keys directly.

When you attempt to generate keys for a Security World that complies with FIPS 140-2 Level 3, you are prompted to insert an Administrator Card or Operator Card. You may need to specify to the application, the slot you are going to use to insert the card. You need to insert the card only once in a session.



For softcard protected key generation, you must use an Operator Card Set.

Generating a key creates both a key and a certificate request for the following application types:

- `embed` (OpenSSL)
- `kpm`

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

12.2.1. Generating keys using the command line

Keys are generated using the command line with the `generatekey` utility. The `--generate` option creates a new key on the host computer that is protected either by the module or by an Operator Card set from the Security World. No key material is stored in an unencrypted form on the host computer.

When you generate a key with `generatekey`, choose a new identifier for the key and use whichever application type is appropriate. The key identifier can only contain digits and lowercase letters; it cannot contain spaces, underscores (`_`), or hyphens (`-`).

You can use `generatekey` in two ways:

- In interactive mode, by issuing commands without parameters and supplying the

required information when prompted by the utility

- In batch mode, by supplying some or all of the required parameters using the command line (**generatekey** prompts interactively for any missing but required parameters).

In interactive mode, you can input **abort** at any prompt to terminate the process.

Batch mode is useful for scripting. In batch mode, if any required parameters are omitted, **generatekey** does not prompt for the missing information but instead will either use available defaults or fail. If you specify one or more parameters incorrectly, an error is displayed and the command fails.

If the Security World was created with audit logging selected then you can request that the usage of a key for cryptographic operations is logged in the audit log. By default only key generation and destruction is logged. For further information see [Audit Logging](#).

To generate a key, use the command:

```
generatekey --generate [OPTIONS] <APPNAME> [<NAME>=<VALUE> ...]
```

In this command:

- **--generate** option specifies that this instance of **generatekey** is generating a key. Other options can be specified to perform tasks such as importing or retargeting keys. To see a list of options run the command **generatekey --help**.
- the **<APPNAME>** parameter specifies the name of the application for which the key is to be generated. For details of the available application types (**APPNAME**), see [Key application type \(APPNAME\)](#).
- The **<NAME>=<VALUE>** syntax is used to specify the properties of the key being generated. For details of the available application types (**APPNAME**), see [Key properties \(NAME=VALUE\)](#).

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

In interactive mode, **generatekey** prompts you for any required parameters or actions that have not been included in the command. When you give the command:

1. Enter parameters for the command, as requested. If you enter a parameter incorrectly, the request for that information is repeated and you can re-enter the parameter.
2. When all the parameters have been collected, **generatekey** displays the final settings. In a FIPS 140-2 Level 3 compliant Security World, you are prompted to insert a card for FIPS authorization if no such card is present.

3. If prompted, insert an Administrator Card or an Operator Card from the current Security World.
4. If you want to protect the key with an OCS, you are prompted to insert the relevant cards and input pass phrases, as required.

12.2.1.1. Example of key generation with `generatekey`

To generate a simple RSA key in batch mode, protected by module protection, use the command:

```
generatekey --generate --batch simple type=rsa size=2048 plainname=keya ident=abcd certreq=yes
```

The `generatekey` utility prompts you to insert a quorum of Operator Cards from the **operatorone** OCS. After you have inserted the appropriate number of cards, `generatekey` generates the key.

Although it is not explicitly specified, the created key is recoverable by default if OCS and softcard replacement is enabled for the Security World.

12.2.2. Generating keys with KeySafe

In order to generate a key with KeySafe, follow these steps:

1. Start KeySafe. (For an introduction to KeySafe and for information on starting the software, see [Using KeySafe](#).)
2. Click the **Keys** menu button, or select **Keys** from the **Manage** menu. KeySafe takes you to the **Keys** panel, which shows the keys in the Security World.
3. Click the **Create** button to open the **Generate Key** panel.
4. Select an application with which you want to use your key from the list, and then click the **Next** button. KeySafe takes you to the **Key Generation Parameters** panel.
5. Select and enter your desired parameters for key generation.

The types of information that you need to provide on the **Key Generation Parameters** panel differs slightly depending on the application you selected on the **Generate Key** panel.

6. When you have supplied your desired key generation parameters, click the **Commit** button.



In order to generate a key protected by a FIPS 140-2 Level 3 compliant Security World, you need authorization from an Operator Card

or Administrator Card from the current Security World. Follow the onscreen instructions.

7. If you choose to generate a key that is protected by a smart card or softcard, KeySafe takes you to a panel from which you can load the protecting card or softcard. Follow the onscreen instructions, inserting any necessary Operator Cards and supplying any pass phrases as needed.
8. KeySafe displays a message indicating that the key has been successfully generated. Click the **OK** button.
9. KeySafe returns you to the **Generate Key** panel, from which you can generate another key or choose another operation.

12.2.3. Generating NVRAM-stored keys

NVRAM key storage provides a mechanism for generating keys stored in a module's non-volatile memory and hence within the physical boundary of an nShield module. You can store only a few keys in this way: the number depends on the memory capacity of the module, the size of the key and whether the key has recovery data associated with it.



We recommend that you *do not store keys in NVRAM unless you must do so to satisfy regulatory requirements*. NVRAM key storage was introduced only for users who must store keys within the physical boundary of a module to comply with regulatory requirements. NVRAM-stored keys provide no additional security benefits and their use exposes your ACS to increased risk. Storing keys in nonvolatile memory also reduces load-balancing and recovery capabilities. Because of these factors, we recommend you always use standard Security World keys unless explicitly required to use NVRAM-stored keys.

When you generate an NVRAM-stored key, you must have sufficient nonvolatile memory available in the module or the command fails.



You need backup and recovery procedures, which must be consistent with regulatory requirements, to protect your NVRAM-stored keys. Do **NOT** use Remote Administration to back-up keys to a smart card, as, in transit, the keys would not be physically protected from access by the host system.



An NVRAM-stored key can only be loaded successfully by using the **pre load** command-line utility on the generating module. Attempts to load such a key on other modules that have NVRAM fail with **UnknownID**

errors.

We provide the `nvr-ram-backup` utility to enable the copying of files, including NVRAM-stored keys, between a module's nonvolatile memory and a smart card.

12.3. Importing keys

Importing a key takes an unprotected key stored on the host and stores it in the Security World in encrypted form.



We recommend generating a new key (or retargeting a key from within the Security World) instead of importing an existing key whenever possible. The import operation does not delete any copies of the key material from the host, and because existing keys have been stored in a known format on your hard disk (and key material can persist on backup media), there is a risk that an existing key has been compromised. It is your responsibility to ensure any unprotected key material is deleted. If a key was compromised before importation, then importing it does not make it secure again.

The following key types can be imported by the tools we provide:

- RSA keys in PEM-encoded PKCS #1 format (from a file). The PEM key that contains the key to import must not require a pass phrase.
- DES, DES2 and Triple DES keys (entered in hex).



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

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

12.3.1. Importing keys from the command line

You can import keys using the `generatekey` utility. To import a key, give the command:

```
generatekey --import [<OPTIONS>] <APPNAME> [<NAME>=<VALUE> ...]
```

This command uses the following options:

Option	Description
<code>--import</code>	This option specifies key importation.
<code><OPTIONS></code>	You can specify particular options when running <code>generatekey</code> that control details of key importation.
<code><APPNAME></code>	This option specifies the name of the application for which the key is to be imported. This must be an application for which <code>generatekey</code> can generate keys.
<code><NAME>=<VALUE></code>	This specifies a list of parameters for the application.

For RSA keys, you can include `pemreadfile=filename` in the command to specify the file name of the PEM file that contains the key. Otherwise, you are prompted for this information during import.

In interactive mode, you are prompted for any required parameters or actions that have not been included in the command:

- Enter parameters, as requested. If you enter a parameter incorrectly, the request for that information is repeated and you can re-enter the parameter.
- If you want to protect the key with an OCS, you are prompted to insert the relevant cards and input pass phrases, as required.
- If prompted, insert an Administrator Card or an Operator Card from the current Security World.

12.3.1.1. Example of key importation with `generatekey`

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

```
generatekey --generatekey --import simple pemreadfile=/opt/projects/key.pem plainname=importedkey ident=abc
protect=module
```

In this example, `generatekey` requires you to input `RSA` for the key type.

Although not explicitly specified, this key is, by default, recoverable if OCS and softcard replacement is enabled for the Security World.

12.3.2. Importing keys with KeySafe

Any user who has write access to the directory that contains the Security World can import a key.

In order to import a key with KeySafe, follow these steps:

1. Start KeySafe. (For an introduction to KeySafe and for information on starting the software, see [Using KeySafe](#).)
2. Click the **Keys** menu button, or select **Keys** from the **Manage** menu. KeySafe takes you to the **Keys** panel.
3. Click **Import** to open the **Import Key** panel.
4. Select the application associated with the key that you want to import, and then click the **Next** button. KeySafe takes you to the **Key Import Parameters** panel.
5. Select and enter the desired parameters for the key that you want to import.

The types of information that you need to provide on the **Key Import Parameters** panel will differ slightly depending on the application you selected on the **Import Key** panel.

6. When you have supplied parameters for the key that you want to import, click the **Commit** button.
7. If you choose to import a key that is protected by a smart card, KeySafe takes you to the **Load Operator Card Set** panel. Follow the onscreen instructions, inserting the required number of Operator Cards and supplying any pass phrases as needed.
8. KeySafe displays a message indicating that the key has been successfully imported. Click the **OK** button.
9. KeySafe returns you to the **Import Key** panel, from which you can import another key or choose another operation.

12.4. Listing supported applications with generatekey

To list supported applications, use the command:

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

12.5. Retargeting keys with generatekey

The **--retarget** option takes an existing key in the Security World and makes it available for use by another application as if it had been expressly generated for use by that application. Because no key material is exposed during retargeting, this operation is as secure as generating a new key.



When you retarget a key, **generatekey** does not remove the original key

from the Security World. If required, you can use KeySafe to discard the original key.

When you retarget a key, you cannot change its protection method. You cannot change the key from module-protected to card-protected, or from card-protected to module-protected.

To retarget a key, use the command:

```
generatekey --retarget [<OPTIONS>] <APPNAME> [from-application=<appname>]
[from-ident=<keyident>]
```

In this command:

Option	Description
<code>--retarget</code>	This option specifies key importation.
<code><OPTIONS></code>	This option specifies any options to include when the command is run. Run the command <code>generatekey --help</code> for details about the available options.
<code><APPNAME></code>	This option specifies the name of the application for which the key is to be generated. This must be an application for which <code>generatekey</code> can generate keys.
<code>from-application=<appname></code>	This option specifies the name of the application with which the key is currently associated.
<code>from-ident=<keyident></code>	This option specifies the identifier of the key to be retargeted. You can find this identifier by using the <code>nfkminfo</code> command-line utility.

If `generatekey` cannot identify the key type for retargeting, you are prompted to specify the key type. Input the key type and press Enter.

12.6. Viewing keys

You can view existing keys in the Security World using KeySafe, the unit front panel, or the `nfkminfo` command-line utility.

12.6.1. Viewing information about keys on the unit front panel

You can view keys that have been created on the client on the same computer as the RFS with SEE machines. You cannot view other keys until they are transferred to the RFS.

To view keys:

1. From the main menu, select **Security World mgmt > Keys > List keys**.
2. Select the application to which the key belongs.
3. Select a key to view its full details.
4. If you wish, select **Verify key ACLs** to verify the key's ACL.

12.6.2. Viewing keys with KeySafe

In order to view a list of keys on the client computer on which you are running KeySafe, follow these steps:

1. Start KeySafe. (For an introduction to KeySafe and for information on starting the software, see [Using KeySafe](#).)
2. Click the **Keys** menu button, or select **Keys** from the **Manage** menu. KeySafe takes you to the **Keys** panel, which lists all the keys in the Security World on this client computer. It displays the name of the key, the application for which it was created, the protection method that was used and whether the key is stored in NVRAM.

If you click a key's listing, KeySafe displays additional information about that key, for example, the application with which the key is used, whether or not the key is recoverable, and the key's name, creation date, hash, instance, and copy ID.

From the **Keys** panel, you can choose to:

- Create a new key (see [Generating keys with KeySafe](#))
- import a key (see [Importing keys with KeySafe](#))
- discard a key from the Security World (see [Discarding keys](#))

12.6.3. Viewing keys using the command line

The **nfkminfo** command-line utility is used to list keys. To list the keys that have been created in the current Security World, use one of the following commands:

```
nfkminfo -k [<APPNAME>[<IDENT>]]
```

```
nfkminfo -l [<APPNAME>[<APPNAME>...]]
```

The **-k|--key-list** option lists keys only. The **-l|--name-list** option lists keys and their names.

With either option, `<APPNAME>` is an optional application name. If you specify an application name, `nfkminfo` lists only the keys for that application. Commonly, `<APPNAME>` is often one of:

- `custom`
- `embed` (currently only usable with your own provider or provider from a previous nShield release)
- `pkcs11`
- `kpm`
- `kps`
- `seeconf`
- `seeinteg`
- `simple`

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



For example, user-defined application names can be created by using the `nfk` library to generate arbitrary keys.

With the `--name-list` option, `<IDENT>` is the key identifier.

The command `nfkminfo --key-list` returns output of the form:

```
Key summary - 4 keys:
AppName appname      Ident <ident> AppName <appname>
Ident <ident> AppName <appname>
Ident <ident> AppName <appname>                Ident <ident>
```

To list information about a specific key, specify the `--key-list` option with an application and key identifier:

```
nfkminfo --key-list <appname> <ident>
```

This command returns output of the form:

```
Key AppName <appname> Ident <ident> BlobKA length  752
BlobPubKA length      316
BlobRecoveryKA length 868
name                  "name"
hash                  hash recovery                Enabled
protection            CardSet
other flags           PublicKey +0x0
cardset               hash_ktBlobKA
format               6 Token
other flags           0x0
```

```

hkm          hash_km hkt          hash_kt hkr          none
BlobRecoveryKA
format       8 Indirect
other flags  0x0
hkm          none
hkt          none
hkr          hash_krBlobPubKA
format       5 Module
other flags  0x0
hkm          hash_km hkt          none
hkr          none
No extra entries

```

To list keys and names, specify the `--name-list` option. The command `nfkminfo --name-list` returns output of the form:

```

Key summary - 30 keys
in format key_<appname>_<ident> '<name>')
key_appname_ident'name '
key_appname_ident'name '
key_appname_ident'name '
key_appname_ident'name '
key_appname_ident'name '
key_appname_ident'name '
key_appname_ident'name '

```

12.7. Verifying Key Generation Certificates with `nfkverify`

The `nfkverify` command-line utility verifies key generation certificates. You can use `nfk-mverify` to confirm how a particular Security World and key are protected. It also returns some information about the Security World and key.

The `nfkverify` utility compares the details in the ACL of the key and those of the card set that currently protects the key.

A key that has been recovered to a different card set shows a discrepancy for every respect that the new card set differs from the old one. For example, a key recovered from a 2-of-1 card set to a 1-of-1 card set has a different card-set hash and a different number of cards, so two discrepancies are reported. The discrepancy is between the card set mentioned in the ACL of the key and the card set by which the key is currently protected (that is, the card set mentioned in the key blobs).

A key that has been transferred from another Security World shows discrepancies and fails to be verified. Entrust recommends that you verify keys in their original Security World at their time of generation.



If you must replace your Security World or card set, Entrust recommends that you generate new keys whenever possible. If you must transfer a key, perform key verification immediately before transferring the


key; it is not always possible to verify a key after transferring it to a new Security World or changing the card set that protects it.

12.7.1. Usage

To verify the key generation certificates from the command line, run the command:

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

Optionally, the command can also include the following:

Option	Description
<code>-h --help</code>	This option displays help for <code>nfmverify</code> .
<code>-V --version</code>	This option displays the version number for <code>nfmverify</code> .
<code>-u --usage</code>	This option displays a brief usage summary for <code>nfmverify</code> .
<code>-m --module=MODULE</code>	This option performs checks with module <code>MODULE</code> .
<code>-f --force</code>	This option forces display of an output report that might be wrong.
<code>-U --unverifiable</code>	<p>This option permits operations to proceed even if the Security World is unverifiable.</p> <div>  <p>If you need the <code>-U --unverifiable</code> option, there may be some serious problems with your Security World.</p> </div>
<code>-v --verbose</code>	This option prints full public keys and generation parameters.
<code>-C --certificate</code>	This option checks the original ACL for the key using the key generation certificate. This is the default.
<code>-L --loaded</code>	These options check the ACL of a loaded key instead of the generation certificate.
<code>-R --recov</code>	This option checks the ACL of the key loaded from the recovery blob.
<code>--allow-dh-unknown-sg-group</code>	This option allows an operation to proceed even if a Diffie-Hellman key is using an unrecognized Sophie-Germain group.

Option	Description
<code>-A --assigned</code>	<p>In a <code>common-criteria-cmts</code> Security World <code>nfkverify</code> will identify keys as Assigned or General, see Common Criteria CMTS Mode Assigned Keys based on the criteria in the <i>nShield Solo XC Common Criteria Evaluated Configuration Guide</i>, and print the classification by default. When considering the key's timeout and usage limits <code>nfkverify</code> will consider these limits against the <code>max-keyusage</code> and <code>max-keytimeout</code> values set on a <code>common-criteria-cmts</code> Security World. If there is a maximum value set on the Security World, any non-zero value less than or equal to this is considered compatible with the reauthorization conditions for an Assigned Key. If the maximum value is not set on the Security World, no value or any value is considered compatible with the reauthorization conditions for an Assigned Key.</p> <p>This option, in a <code>common-criteria-cmts</code> mode Security World, means the <code>nfkverify</code> utility will exit with a non-zero exit code if the key is not an Assigned Key. This supports testing for Assigned Keys programmatically.</p>

12.8. Discarding keys

Discarding a key deletes the information about the key from the host disk. This option is only available in KeySafe.

If you have backup media, you can restore the information and the key becomes usable again. Likewise, if you have copies of the Security World data on several computers, erasing the data on one computer does not remove it from any other computer.

To destroy a key permanently you must either erase the OCS that is used to protect it or erase the Security World completely. There are no other ways to destroy a key permanently.

12.9. Restoring keys

We do not supply tools for restoring a key that has been discarded. However if you have kept a backup of the host data for the Security World, you can restore a key from the backup data.



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

13. Using KeySafe

KeySafe provides a GUI based interface to perform many of the main tasks required to use an nShield Security World. This appendix describes KeySafe, the Security World management tool. It includes information about:

- Starting KeySafe
- Using the graphical user interface (GUI) for KeySafe
- Using buttons to select and run operations
- Using the keyboard to navigate KeySafe
- KeySafe error reporting.

To perform Security World management, card-set management, and key management tasks using KeySafe, see the relevant chapters of this guide.



By default, KeySafe uses the same mechanisms and supports the same features and applications as the `generatekey` utility.

13.1. Setting up KeySafe

1. You must have Java JRE/JDK 1.7, 1.8 or 11. We recommend that you install Java before you install the Security World Software. The Java executable must be on your path.

Java software is available from <http://www.oracle.com/technetwork/java/>. If your security policy does not allow the use of downloaded software, these components can be obtained on removable media from Oracle or from your operating system vendor.

The `keysafe.jar` file must be specified in the Java class path.#



After you have set up the path, check that you are using the correct Java version by running java with the `-version` option.

Example:

```
>>java -version
java version "1.8.0_05"
Java(TM) SE Runtime Environment (build 1.8.0_05-b13)
Java HotSpot(TM) 64-Bit Server VM (build 25.5-b02, mixed mode)
```

2. The Security World Software must be installed.
3. In the configuration file at `$NFAST_KMDATA/config/config`, set the following port values in the `server startup` section:

```
nonpriv_port=9000  
priv_port=9001
```

You must restart the hardserver after this change.



See the Installation Guide for more about ports and firewall settings.

13.2. Starting KeySafe

Ensure that Xwindows is properly configured and running before starting KeySafe.

Start KeySafe by running the `/opt/nfast/bin/ksafe` script (assuming you installed the Security World Software in the default `/opt/nfast/` directory).

13.3. About the KeySafe window

The KeySafe window is divided into two areas:

- The sidebar (on the left), subdivided into:
 - The menu buttons (at the top of the sidebar)
 - The Security World status pane (at the bottom of the sidebar)
- The main panel area (on the right).

This layout is consistent throughout the KeySafe application.

13.3.1. Sidebar


The sidebar provides access to different parts of the KeySafe application (with the menu buttons) and also displays information about both the current Security World and your module or modules (with the Module Status tree).



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

13.3.2. Menu buttons

There are five menu buttons at the top of the sidebar:

Menu button	Description
Introduction	Clicking the Introduction menu button opens the introductory panel that KeySafe displays at startup.
World	<p>Clicking the World menu button opens the World Operations panel, from which you can:</p> <ul style="list-style-type: none"> • Add modules to a Security World • Remove modules from a Security World. <div>  <p>You cannot perform these operations on a module that is not in the pre-initialization mode.</p> </div>
Card Sets	<p>Clicking the Card Sets menu button opens the List Operator Card Sets panel, from which you can:</p> <ul style="list-style-type: none"> • Examine or change an Operator Card Set or its pass phrase • Create a new Operator Card Set • Replace an Operator or Administrator Card Set • Discard an Operator Card Set.
Softcards	<p>Clicking the Softcards menu button opens the List Softcards panel, from which you can:</p> <ul style="list-style-type: none"> • Create a new softcard • Change or recover the pass phrase on a softcard • Discard a softcard
Keys	<p>Clicking the Keys menu button opens the Keys panel, from which you can:</p> <ul style="list-style-type: none"> • Create a key • Import a key • Discard a key • View details of a key.

While KeySafe is executing a command, the menu buttons are disabled. Their normal functionality returns when the command is completed.

13.3.3. Menus

Three menu options are available from the menu bar at the top of the screen:

- **File**
 - **Exit** - displays a dialog asking whether you are sure you wish to quit KeySafe. Click **Yes** (or press the **Enter** key) to close KeySafe. Click **No** to close the dialog and return to your KeySafe session.

- **Manage**

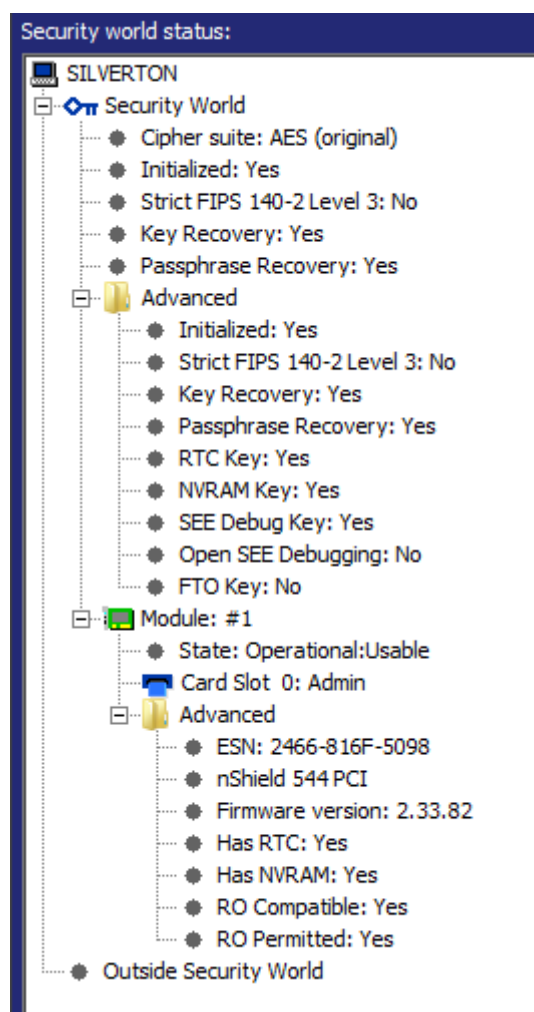
- **Introduction** - opens the **Introduction** panel. See [Introduction button](#).
- **World** - opens the **World Operations** panel. See [World button](#).
- **Card sets** - opens the **List Operator Card Sets** panel. See [Cardsets button](#).
- **Softcards** - opens the **List Softcards** panel. See [Soft Cardsets button](#).
- **Keys** - opens the **Keys** panel. See [Keys button](#).

- **Help**

- **About KeySafe** - opens the **About KeySafe** panel, which displays current version numbers for KeySafe, kmjava and nfjava. You will need to quote these version numbers if you contact Support about KeySafe.

13.3.4. Module Status tree

The Module Status tree, in the lower part of the KeySafe sidebar, displays information about the current Security World and your modules in the form of a tree diagram.



At the top of the Module Status tree is an icon representing the computer on which the running copy of KeySafe is installed. The name of this computer is shown to the right of the icon.

Below the computer icon in the Module Status tree are icons and text identifiers representing the current Security World and your module(s). To the left of each icon is an expand/collapse toggle, or node. By default, when KeySafe starts, these nodes are collapsed and show a minus sign. Click the node to display expanded information about the Security World or module. Click the node again to collapse this information.

13.3.4.1. Security World information

At the top level of the Security World tree, the following information is displayed:

- **Cipher suite** — the type of key protecting the Security World
- **Initialized** — whether the Security World is initialized (Yes or No)
- **Strict FIPS 140-2 Level 3** — whether the Security World is operating at FIPS 140-2 Level 3 (Yes or No)
- **Key Recovery** — whether key recovery is enabled (Yes or No)
- **pass phrase Recovery** — whether pass phrase recovery is enabled (Yes or No). For more information, see [Pass phrase replacement](#).

When the **Advanced** node is expanded, the following additional information is displayed:

- **RTC Key** — whether a real-time clock authorization key has been generated (Yes or No)
- **NVRAM Key** — whether a non-volatile memory authorization key has been generated (Yes or No)
- **SEE Debug Key** — whether SEE debugging has been enabled (Yes or No)
- **Open SEE Debugging** — whether Open SEE debugging has been enabled (Yes or No)
- **FTO Key** — whether a foreign token key has been generated (Yes or No)

13.3.4.2. Module information

Module information may be displayed either inside or outside the Security World. Modules that have not been incorporated into a Security World will be shown beneath the **Outside Security World** node.

At the top level of the Module tree, the following information is displayed:

- The module's state, which is one of the following:

Mode	Description
PreInitMode	The module is in pre-initialization mode.
InitMode	The module is in initialization mode.
Operational:Useable	The module is in the current Security World and useable for key operations.
Operational:Unknown	The mode of the module cannot be determined.
Operational:Uninitialized	The module key is set and the module must be initialized before use.
Operational:Factory	The module key is set to the factory default.
Operational:Foreign	The module is from an unknown Security World.
Operational:AccelOnly	The module is an acceleration-only module.
Operational:Unchecked	Although the module appears to be in the current Security World, KeySafe cannot find a module initialization certificate (a module_ESN file) for this module
Failed	The module has failed.
PreMaintMode	The module is in the pre-maintenance mode.
MaintMode	The module is in the maintenance mode.

- the status of the smart card reader slot(s).



For FIPS 140-2 Level 3 Security Worlds, a **FIPS Auth Loaded** entry shows if an Administrator Card or Operator Card has been inserted to authorize an operation that requires a FIPS key.

The Module status tree has an **Advanced** folder that shows the following details when expanded:

- **ESN** — the module's electronic serial number (ESN), which is a unique identifier. You must quote a module's ESN if you need to contact Support. Keep a record of the ESN(s) associated with your module(s).
- the **HSM type** and model number
- **Firmware version** — the version of the module's firmware
- **Has RTC** — whether the module has a Real Time Clock (RTC)
- **Has NVRAM** — whether the module has nonvolatile memory (NVRAM).
- **RO Compatible** —
- **RO Permitted** —

13.3.5. Main panel area

The KeySafe main panel area is used to display information and options pertaining to a chosen operation. For example, clicking the **World** menu button takes you to the **World Operations** panel in the main panel area.

13.3.5.1. Navigation and command buttons

On each **Operations** panel there are a number of *navigation buttons*. Clicking a navigation button does *not* commit you to an action, but instead selects an operation and loads another panel of additional information and options related to the selected operation. From the **World Operations** panel, for example, clicking the **Erase Module** navigation button does not itself erase a module, but rather loads the **Erase Module** panel.

On the next panel, the **Commit** button executes an operation, while the **Back** button returns to the previous panel. For example, on the **Erase Module** panel, clicking the **Commit** button will erase the module, while clicking the **Back** button will return to the **World Operations** panel.



Clicking the **Commit** button tells KeySafe to write or delete data: it is not necessarily possible to reverse such changes even if you subsequently cancel the operation. In some cases, clicking the **Commit** button causes KeySafe to display a dialog asking you to confirm your command. Such features help prevent you from accidentally destroying your data or keys.

Some panels require that you select options by means of radio buttons or that you enter data into text fields before clicking the **Commit** button. For example, if you click the **Reprogram Module** button on the **World Operations** panel, the next panel prompts you to choose whether the module can receive remote operator card shares.

Input may be in the form of radio buttons (allowing several options, one of which — the *default* — will be already selected) or text boxes (allowing you to enter text: no default value is set). If you click the **Commit** button without having entered data into a mandatory text field, or if KeySafe detects that the information you provided is inconsistent or invalid, KeySafe displays an error message. Click the message's **OK** button, and then provide or correct the necessary information.

After you successfully issue a command by clicking the **Commit** button, the menu buttons are disabled until the requested command is completed.

13.3.5.2. Navigating with the keyboard

The **Tab** key always takes you to the next field or button. If the cursor is not currently active in a text field, pressing the space bar or the **Enter** key activates the currently selected button (as if you had clicked it). Pressing the **Shift-Tab** button combination takes you to the previous field (if any) or deselects an automatically selected button (if any).

13.4. Errors

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

Error message	Possible causes	Suggested solutions
Unable to establish KeySafe session. Please ensure that the hardserver is running and accepting TCP connections. Click OK to exit.	<ul style="list-style-type: none"> The hardserver is unable to receive TCP connections. The server program communicates with clients by using named pipes or TCP sockets. The hardserver is not running, or is physically disconnected. 	<ul style="list-style-type: none"> Check the hardserver configuration file settings: see server_startup. To restart the hardserver: <ol style="list-style-type: none"> Exit KeySafe Restart the server (as described in Stopping and restarting the client hardserver) Restart KeySafe.
Unable to generate key: Error reported by nShield hardware module nFast error: UnknownFlag, in response to GenerateKeyPair	Your hardware or firmware may not be up to date.	<p>To update your firmware:</p> <ol style="list-style-type: none"> Exit KeySafe Update the firmware as described in Upgrading the nShield Connect image file and associated firmware Restart KeySafe <p>The firmware upgrade process destroys all persistent data held in a key-management module. If your security system requires that the persistent data held in a key-management module must survive intact during the upgrade or initialization of the key-management module, a backup and recovery mechanism of your kmdata directory must be implemented.</p>

If you receive any error message titled **Unexpected Error**, contact Support with details of what you were doing, and the exact error message.

14. Supplied utilities

This appendix describes the executable command-line utilities (utilities) that you can use for performing various configuration and administrative tasks related to your module.

These utilities exist in the **bin** subdirectory of your Security World Software installation. Unless noted, all utilities have the following standard help options:



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

14.1. Utilities for general operations

Use the utilities described in this section to:

- Check the module configuration and verify that it functions as expected.
- Obtain statistics for checking the performance of the module.

Utility	Enables you to...
enquiry	<p>Obtain information about the hardserver (Security World Software server) and the modules connected to it.</p> <p>Use this utility to:</p> <ul style="list-style-type: none"> • Check if the software has been installed correctly • Check the firmware version • Check if the Remote Operator feature is enabled • Check if the Serial Console feature is available • Check the hardware status of internal security modules <p>See Testing the installation for more information.</p>
checkmod	Check modulo exponentiations performed on the module against the test data located in the opt/nfast/testdata directory.
cfg-mkdefault	Create a default client configuration file for the hardserver configuration sections.
cfg-remoteslots	Configures Remote Operator slot imports and exports. See Remote Operator .
cfg-reread	Load the hardserver configuration from the configuration file.

Utility	Enables you to...
fet	<ul style="list-style-type: none"> • Activate features • View the status of features • Verify that a feature has been successfully enabled on a connected module <p>To view the status of features, run the tool without a smart card. If a FEM card is not present, or if any of the features are not enabled successfully, the utility prompts you to indicate what to do next.</p> <div>  <p>To enable features, and view the status of or verify features on an nShield Connect unit, use the front panel rather than the fet utility.</p> </div> <p>For more information, see Enabling features with a smart card.</p>
ncdate	View, set, and update the time on a module's real-time clock.
ncversions	<p>Obtain and verify the versions of the Security World Software components that are installed. This utility lists the following information:</p> <ul style="list-style-type: none"> • Versions of all components, irrespective of whether they are installed individually or as part of a component bundle • Version of each component bundle
nfdiag	<p>Obtain information about the module and the host on which it is installed. This diagnostic utility can save information to either a ZIP file or a text file.</p> <p>For more information, see nfdiag: diagnostics utility.</p> <div>  <p>Run this utility only if requested to do so by Support.</p> </div>
nfwarrant	<p>Ensure that a suitable warrant is available to allow a Security World to be dynamically managed using an nShield Solo or nShield Edge.</p> <ul style="list-style-type: none"> • Identify modules that have the appropriate firmware/KLF2 key • Identify modules that need their KLF2 key to be warranted by Entrust • Generate a warrant upgrade request for a specific module, as required • Install an upgraded warrant • List KLF2 warrants

Utility	Enables you to...
<code>nopclearfail</code>	<p>Clear an HSM, put an HSM into the error state, retry a failed HSM, or change the HSM mode.</p> <p>You must use a privileged connection to use this utility with the following parameters:</p> <ul style="list-style-type: none"> • change the mode of the HSM (<code>nopclearfail -I/M/0</code>) • Clear the module (<code>nopclearfail -c</code>)
<code>nvr-ram-backup</code>	Copy files between a module's NVRAM and a smart card, allowing files to be backed up and restored.
<code>nvr-ram-sw</code>	View and modify information about NVRAM areas.
<code>pubkey-find</code>	Obtain information of the public key from a certificate or certificate request (in a Base-64 encoded PEM file).
<code>randchk</code>	Run a universal statistical test on random numbers returned by the module.
<code>rtc</code>	View and set the module's real-time clock.
<code>slotinfo</code>	<ul style="list-style-type: none"> • Obtain information about tokens in a module • Format a smart card
<code>snmpbulkwalk</code> <code>snmpget</code> <code>snmpgetnext</code> <code>snmpset</code> <code>snmptranslate</code> <code>snmpwalk</code>	<p>Obtain system, module, connection and software information from the SNMP agent.</p> <p>For more information, see Using the SNMP command-line utilities.</p>
<code>stattree</code>	Obtain statistics gathered by the Security World Software server and modules. For more information, see stattree: information utility .
<code>sbin/logrotate-hardserver</code>	<p>Archive the existing hardserver log from <code>/opt/nfast/log/hardserver.log</code> and re-open as a fresh log file.</p> <p>When run with no arguments, it will automatically archive the existing log to <code>/opt/nfast/log/archive/hardserver.DATETIME.log</code> (where <code>DATETIME</code> is the current date and time). The directory <code>/opt/nfast/log/archive/</code> is created if it does not already exist.</p> <p>Optionally, a single argument can be provided with the full file name to archive the existing hardserver log to.</p> <p>This script must be run as root.</p>

14.2. Hardware utilities

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

Utility	Enables you to...
<code>fwcheck</code>	Verify the firmware installed on a module.
<code>nfloadmon</code>	Upgrade the module monitor and firmware of nShield Edge and nShield Solo modules.

14.3. Test analysis tools

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



All the listed utilities, except the `floodtest` utility, are supported only on FIPS 140-2 Level 2 Security Worlds.


Utility	Enables you to...
<code>crypttest</code>	Test all defined symmetric cryptographic mechanisms.
<code>des_kat</code>	Perform DES known-answer tests. This utility indicates if any of them fail.
<code>floodtest</code>	Perform hardware speed-testing by using modular exponentiation.
<code>kptest</code>	Test the consistency of encryption and decryption, or of signature and verification, with the RSA and DSA algorithms.
<code>ncthread-test</code>	Stress test modules and test nCore API concurrent connection support.
<code>perfcheck</code>	Run various tests to measure the cryptographic performance of a module. For more information, see perfcheck: performance measurement checking tool .
<code>sigtest</code>	Measure module speed using RSA or DSA signatures or signature verifications.
<code>ncperftest</code>	Test the performance of various crypto commands using attached nShield hardware. Available since v12.10 it contains all the functionality in <code>sigtest</code> and <code>floodtest</code> as well as several new features and greater accuracy and throughput capability in performance management.



14.4. Security World utilities

Use the utilities described in this section to:

- Set up and manage Security Worlds.
- Create and manage card sets and pass phrases.

- Generate keys and transfer keys between Security Worlds.

Utility	Enables you to...
bulk erase	<p>Erase multiple smart cards including Administrator Cards, Operator Cards, and FEM activation cards, in the same session.</p> <div>  <p>Do not use the bulk erase utility to erase Administrator Cards from the current Security World.</p> </div>
cardpp	<p>Change, verify, and recover a pass phrase of an Operator Card. For more information, see:</p> <ul style="list-style-type: none"> • Verifying the pass phrase of a card with cardpp. • Changing known pass phrases with cardpp. • Changing unknown or lost pass phrases.
createocs	<p>Create and erase an OCS. For more information, see:</p> <ul style="list-style-type: none"> • Creating an Operator Card Set from the command line. • Erasing cards from the command line.
initunit	<p>Initialize an nShield module.</p> <p>For more information, see Erasing a module with initunit.</p>
generatekey	<p>Generate, import, or retarget keys. This utility is included in the Core Tools bundle, which contains all the Security World Software utilities. For more information, see:</p> <ul style="list-style-type: none"> • Generating keys with the command line. • Importing keys from the command line. • Example of key generation with generatekey, for an example of key generation in batch mode. • Example of key importation with generatekey, for an example of importing an RSA key. • Listing supported applications with generatekey. • Retargeting keys with generatekey.
kmfile-dump	<p>Obtain key management information from a Security World's key management data file.</p>
migrate-world	<p>Migrate existing keys to a destination Security World. For more information, see Security World migration.</p>

Utility	Enables you to...
<code>mkac1x</code>	<p>Generate non-standard cryptographic keys that can be used to perform specific functions, for example, to wrap keys and derive mechanisms. This utility includes options that are not available with the <code>generate-key</code> utility.</p> <div>  <p>Ensure that you run the <code>mkac1x</code> utility with the options that are appropriate for your security infrastructure. If the appropriate options are not chosen, the security of existing keys might potentially be compromised.</p> </div>
<code>new-world</code>	<p>Create and manage Security Worlds on nShield modules.</p> <p>You must use a privileged connection to use this utility with the following parameter:</p> <ul style="list-style-type: none"> Initialize the HSM (<code>new-world -e/i/l</code>)
<code>nfmcheck</code>	Check Security World data for consistency.
<code>nfminfo</code>	<p>Obtain information about a Security World and its associated cards and keys. For more information, see:</p> <ul style="list-style-type: none"> Displaying information about a Security World with <code>nfminfo</code>. Viewing card sets from the command line. Viewing softcards with <code>nfminfo</code>. Viewing keys using the command line. <code>nfminfo</code>: information utility.
<code>nfmverify</code>	<p>Perform Security World verification.</p> <p>For more information, see Verifying Key Generation Certificates with <code>nfmverify</code>.</p>
<code>postrocs</code>	<p>Transfer PKCS #11 keys to a new card set in the new Security World. When transferring keys by using either the <code>key-xfer-im</code> utility or the <code>migrate-world</code> utility, run the <code>postrocs</code> utility if there are any PKCS #11 keys that are protected by OCSs.</p> <div>  <p>PKCS #11 keys either have <code>keys_pkcs_um</code> or <code>key_pkcs_uc</code> as the prefix.</p> </div>

Utility	Enables you to...
<code>ppmk</code>	<ul style="list-style-type: none"> • Create and manage softcards. Use this utility to: • View details of a softcard • Create and delete a softcard • View, change, and recover the pass phrase of a softcard <p>For more information, see:</p> <ul style="list-style-type: none"> • Creating a softcard with ppmk. • Erasing softcards with ppmk. • Viewing softcards with ppmk. • Verifying the pass phrase of a softcard with ppmk. • Changing known softcard pass phrases with ppmk. • Replacing unknown pass phrases with ppmk.
<code>preload</code>	Load keys into a module before an application is run in another session.
<code>racs</code>	<p>Create a new ACS to replace an existing ACS.</p> <p>For more information, see Replacing an Administrator Card Set with racs.</p>
<code>rocs</code>	<ul style="list-style-type: none"> • Restore an OCS from a quorum of its cards • Restore softcards <p>For more information, see:</p> <ul style="list-style-type: none"> • Replacing OCSs or softcards with rocs. • Using rocs from the command line.

14.5. CodeSafe utilities


Use the following helper utilities to develop and sign SEE machines. For more information about these utilities, see the *CodeSafe Developer Guide*.

Utility	Enables you to...
<code>elftool</code>	Convert ELF format executables into a format suitable for loading as an SEE machine.
<code>hsc_loadseemachine</code>	Load an SEE machine into each module that is configured to receive one, then publishes a newly created SEE World, if appropriate.
<code>loadsee-setup</code>	Set up the configuration of auto-loaded SEE machines.
<code>modstate</code>	View the signed module state.

Utility	Enables you to...
see-sock-serv see-stdioe-serv see-stdioesock-serv see-stdoe-serv	Activate or enable standard IO and socket connections for SEE machines using the bsdlib architecture.
tct2 (Trusted Code Tool)	Sign, pack, and encrypt file archives so that they can be loaded onto an SEE -ready nShield module.

14.6. PKCS #11

Use the following utilities to manage the interfaces between the PKCS #11 library and the module.

Utility	Enables you to...
ckcerttool	Import a certificate as a PKCS #11 CKO_CERTIFICATE object of type CKC_X_509 , and optionally, associate it with the corresponding private key.
ckcheckinst	Verify the installation of the nShield PKCS #11 libraries. For more information, see Checking the installation of the nCipher PKCS #11 library .
ckimportbackend	<p>Generate keys for use with PKCS #11 applications. When you run the generatekey utility to generate PKCS #11 keys, the ckimportbackend utility is executed in the background.</p> <div>  <p>Do not run this utility unless directed to do so by Support.</p> </div>
cknfkmid	View values of attributes of PKCS #11 objects.
ckshahmac	Perform a PKCS #11 test for vendor-defined SHA1_HMAC key signing and verification capabilities.
cksigtest	Measure module signing or encryption speed when used with nShield PKCS #11 library calls.

If you have installed [Cipher Tools](#), you can use the following additional PKCS #11 utilities. For more information about these utilities, see the *Cryptographic API Integration Guide*.

Utility	Enables you to...
ckinfo	View PKCS #11 library, slot, and token information. Use this utility to verify that the library is functioning correctly.

Utility	Enables you to...
<code>cklist</code>	View details of objects on all slots. If invoked with a PIN argument, the utility lists public and private objects. If invoked with the <code>-n (--nopin)</code> option, the utility lists only the public objects. This utility does not output any potentially sensitive attributes, even if the object has <code>CKA_SENSITIVE</code> set to <code>FALSE</code> .
<code>ckmechinfo</code>	View details of the supported PKCS #11 mechanisms provided by the module.
<code>ckrsagen</code>	Test RSA key generation. You can use specific PKCS #11 attributes for generating RSA keys.
<code>cksotool</code>	Create a PKCS #11 Security Officer role, and manage its PIN.


14.7. nShield Connect utilities

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

- Create and manage client configuration files.
- Enroll nTokens with an nShield Connect.
- Set up a Remote File System (RFS) and synchronize Security World data between an nShield Connect and the RFS.
- Administer an nShield Connect remotely
- Configure NTP.


Utility	Enables you to...
<code>anonkneti</code>	View the ESN and <code>HK_{NETI}</code> key hash from a module identified by its IP address. For more information, see Configuring the remote file system (RFS) .
<code>cfg-pushnethsm</code>	Copy a specified configuration file from a remote file system to the file system on a specified module. For more information, see: <ul style="list-style-type: none"> • Remote configuration of additional clients. • About user privileges.
<code>cfg-pushntp</code>	Configure time synchronisation on the nShield Connect, using NTP. For more information, see Configuring NTP in the nShield Connect .

Utility	Enables you to...
<code>config-serverstartup</code>	<p>Edit the <code>[server_startup]</code> section of the configuration file for the client's hard server to enable or disable TCP sockets.</p> <p>For more information, see:</p> <ul style="list-style-type: none"> • After software installation. • config-serverstartup.
<code>nethsmadmin</code>	<p>Administer an nShield Connect remotely. Options include:</p> <ul style="list-style-type: none"> • Check the Security World files on a specified nShield Connect • Copy Security World files from the RFS to the nShield Connect • Command the specified nShield Connect to reboot • Command the nShield Connect to upgrade using the specified nShield Connect image file from its RFS • Retrieve a list of nShield Connect image files available on the RFS • Retrieve a list of feature certificates available on the RFS for a specified nShield Connect • Command the nShield Connect to apply a specified feature certificate from the RFS • Erase the Security World on the nShield Connect and re-initialize the HSM. • Get the date and time on the nShield Connect • Set the date and time on the nShield Connect. <p>You must use a privileged connection to use this utility with the following parameters:</p> <ul style="list-style-type: none"> • Reboot the HSM (<code>nethsmadmin -r</code>) • Erase the Security World (<code>nethsmadmin -e</code>) • Upgrade the HSM firmware (<code>nethsmadmin -i</code>) <p>For more information, see:</p> <ul style="list-style-type: none"> • Using nethsmadmin to copy a Security World to an nShield Connect and check the current version. • Upgrading the nShield Connection image file and firmware from a privileged client. • Using nethsmadmin to reboot an nShield Connect. • Remotely enabling dynamic feature certificates including nShield Connect client licenses.

Utility	Enables you to...
<code>nethsmenroll</code>	<p>Edit the local hardserver configuration file to add the specified nShield Connect unit. As an alternative to hand-editing a client's configuration file, you can run this utility on a client to configure it to access an nShield Connect.</p> <p>For more information, see:</p> <ul style="list-style-type: none"> • nethsm_imports. • Configuring the unit to use the client. • nethsmenroll.
<code>ntokenenroll</code>	<p>Enroll a locally attached nToken with an nShield Connect unit. This utility installs the Electronic Serial Number (ESN) of the nToken within the client configuration file and displays the module's ESN and the hash of the key to be used in nToken authentication.</p> <p>For more information, see Configuring the unit to use the client.</p>
<code>rfs-setup</code>	<p>Create a default RFS hardserver configuration. Run this utility when you first configure the RFS.</p> <p>For more information, see:</p> <ul style="list-style-type: none"> • Configuring the remote file system (RFS). • Setting up client cooperation.
<code>rfs-sync</code>	<p>Synchronize the Security World data between a cooperating client and the RFS. This utility is run on the client.</p> <p>For more information, see:</p> <ul style="list-style-type: none"> • Setting up client cooperation. • rfs-sync. <div>  <p>You can use this utility with nShield modules if an nShield Connect unit is present.</p> </div>

14.8. Developer-specific utilities

Use the following utilities to ensure that the HSMs are functioning as expected and to test the cryptographic functionality at the nCore level.

Utility	Enables you to...
<code>pollbare</code>	<p>Obtain information about state changes. The functionality of this test utility depends on whether the server or an HSM supports nCore API poll commands.</p> <div>  <p>To know if your server or HSM supports nCore API poll commands, run the <code>enquiry</code> utility.</p> </div>

Utility	Enables you to...
<code>trial</code>	Test the nCore API commands. You can use this utility interactively or from a script file.

14.9. Utilities that require a privileged connection

You must be a privileged user, that is, use a privileged connection to the HSM, to run certain utilities with certain parameters.

Utility	Use case
<code>nopclearfail -I/M/O</code>	Change the mode of the HSM
<code>nopclearfail -c</code>	Clear the module
<code>nethsmadmin -r</code>	Reboot the HSM
<code>nethsmadmin -e</code>	Erase the Security World
<code>nethsmadmin -i</code>	Upgrade the HSM firmware
<code>new-world -e/i/l</code>	Initialize the HSM

15. Preload Utility

15.1. Overview

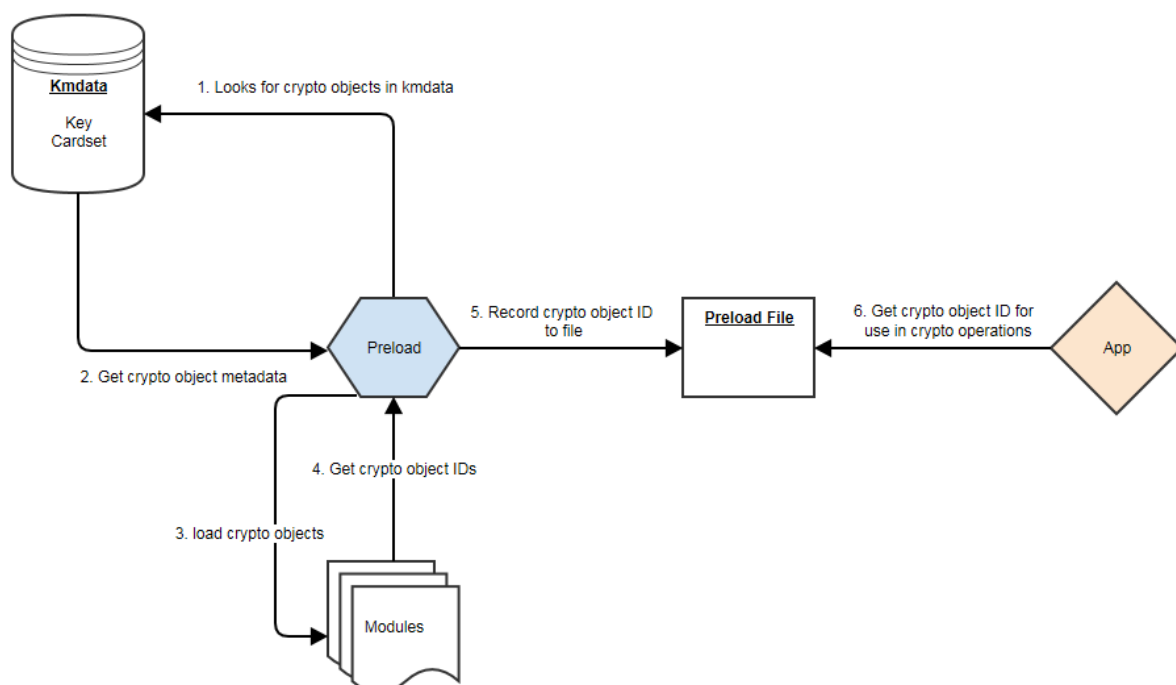
The preload utility loads persistent cryptographic objects (keys/OCS/softcards) onto a chosen set of modules, then makes those objects available for use by applications. This removes the need for applications to load keys/cards themselves, and allows for easy sharing of keys/cards between multiple applications.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

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

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

The preload binary can be found in `/opt/nfast/bin`. This binary calls the **preload.py** script found in `/opt/nfast/python/scripts`

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



15.2. Using Preload

15.2.1. Preload Commands

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

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

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

Preload has a choice of 3 commands:

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



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

The preload session remains open, and thus the preloaded keys remain loaded, as long as at least one instance of preload continues to run. If/when the final preload instance terminates, all loaded objects will be cleaned up.

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

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

15.2.2. Preload file location

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

15.2.3. Preload Command Line Arguments

Argument	Effect
--version	show program's version number and exit

Argument	Effect
<code>-h, --help</code>	show help message and exit
<code>-m MODULE_NUMBER, --module=MODULE_NUMBER</code>	Load on specified module (may be repeated; default = all).
<code>-c IDENT, --cardset=IDENT</code>	Load all cardsets matching <code>IDENT</code> . If <code>IDENT</code> looks like a hash it will be interpreted as that, otherwise it will be interpreted as a name. If it is definitely a name, use <code>--cardset-name</code> .
<code>--cardset-name=NAME</code>	Load cardset(s) named <code>NAME</code> .
<code>-s IDENT, --softcard=IDENT</code>	Load all softcards matching <code>IDENT</code> . If <code>IDENT</code> looks like a hash it will be interpreted as that, otherwise it will be interpreted as a name.
<code>--softcard-name=NAME</code>	Load softcard(s) named <code>NAME</code> .
<code>-o, --any-one</code>	Load a single cardset.
<code>-i, --interactive</code>	Load cardsets interactively until told to stop.
<code>-A APP, --appname=APP</code>	Choose the <code>appname</code> for subsequent <code>-K</code> options.
<code>-K IDENT, --key-ident=IDENT</code>	Load keys with ident matching <code>IDENT</code> .
<code>-n PATTERN, --name-pattern=PATTERN</code>	Load keys with name matching <code>PATTERN</code> . Use <code>*</code> for wildcard.
<code>--name-exact=NAME</code>	Load keys with name <code>NAME</code> .
<code>-M, --module-prot</code>	Load all module protected keys, in addition to any others requested.
<code>--no-cardset-keys</code>	Do not automatically load keys protected by requested cardsets. Deprecated.
<code>--no-token-keys</code>	Do not automatically load keys protected by requested tokens.
<code>--admin=KEYS</code>	Load admin keys (separate with commas, or use <code>all</code>).
<code>--list-admin</code>	List available admin key names (for <code>--admin</code>).
<code>-F, --require-fips</code>	Require FIPS-auth to be loaded.
<code>-N, --no-fips</code>	Do not record FIPS auth, even if available. (overrides <code>-F</code>).
<code>-H, --high-availability</code>	High availability mode.
<code>--polling-interval=POLLING_INTERVAL</code>	Interval (s) between polls for changes to the module list (default=60). High availability mode only.
<code>-f PRELOAD_FILE, --preload-file=PRELOAD_FILE</code>	Use specified preloaded objects file, instead of the default.
<code>-R, --reload-everything</code>	Reload keys and tokens that are already loaded.
<code>--show-key-info</code>	Display key information for keys as they are loaded.
<code>-l, --file-logging</code>	Log to file.

Argument	Effect
<code>-S, --no-stderr-logging</code>	Do not log to <code>stderr</code> , this is independent of file logging.
<code>--log-file=LOG_FILE</code>	The file destination for the log, defaults to <code>preload_%pid.log</code> in the <code>nfast</code> log directory.
<code>--log-level=LOG_LEVEL</code>	The log level to log, options: <code>DEBUG</code> , <code>INFO</code> , <code>WARNING</code> , <code>ERROR</code> . Default is <code>INFO</code> , if unrecognized option it will fall back to default.

15.2.4. Pattern Matching

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

Wildcard	Definition
<code>*</code>	matches everything
<code>?</code>	matches any single character
<code>[seq]</code>	matches any character in seq
<code>[!seq]</code>	matches any character not in seq

It is advised that all arguments that using wildcards are surrounded by quotations to ensure that they are passed to preload as intended. For example, to load all keys whose names start with `keyname`, the following pattern could be used:

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

15.3. Preload File

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

Each entry has the following format:

Element	Description
<code>Hash</code>	the <code>sha1</code> hash of the crypto object
<code>module</code>	The module which this object is present
<code>objectid</code>	The id reference as a <code>M_KeyID</code>
<code>generation</code>

Example **nfkminfo** output with preloaded crypto objects:

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

By default the preload file location is **/tmp**:

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

This location can be changed by using the command line option **-f PRELOAD_FILE**, **--preload-file=PRELOAD_FILE**.

15.4. Softcard Support

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

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

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

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

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

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

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

15.4.1. No Cardset Keys

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

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

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

15.5. FIPS Auth

FIPS Auth can be made available via preload.

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

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

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

15.6. Admin Keys

15.6.1. Listing

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

This should be run without a command:

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

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

15.6.2. Loading

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

Also note that the logical token of the admin key is preloaded alongside the key itself. E.G. `kfips` and `ltfips`.

15.7. High Availability

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

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

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

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

15.7.1. Prerequisites for high availability mode

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

Managing OCS cardset-protected keys requires the following:

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

15.7.2. Differences from legacy behaviour

When running in high availability mode, certain behaviours may differ from those outside of high availability mode. This includes the prompts for PIN entry, error messages, etc. This is due to a necessary difference in implementation between the two modes, and is expected.

15.7.3. Conditions for Management/Reloading

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



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

15.7.4. Merged Keys in the Preload File

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

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

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

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

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

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

To make **nfkminfo** show the preloaded objects, run it as a **subprocess** as part of the **preload** command. (see above section on using preload)



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

15.7.5. Polling Interval

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

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

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

By default the polling interval is 60 seconds.

15.7.6. Key timeouts and use limits

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

In high availability mode, there are situations where OCS/keys that have previously timed out, or reached maximum use limits, may be reloaded (and thus their limits reset) without user interaction. In general within high availability mode keys that have timed out or

reached their use limits will be left in place, unusable, respecting the limits. However if the module containing those keys reboots or resets then, upon the module's return, preload will notice that the keys are not loaded and will load them. This reloading of keys will necessarily reset timeouts and use limits. If the timeout on an OCS has reached its limit, any keys protected by that OCS will not be reloaded on newly-indoctrinated modules in the security world.

15.7.7. Multiple Preload instances in high availability mode

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

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

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

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

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

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

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

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

15.7.7.1. Managing Keys

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

For Example, Security World crypto objects:

crypto object	name	protected by
Softcard	softcard1	N/a

crypto object	name	protected by
Key	simple_softcard1	softcard1
Key	simple_module1	module

First preload process started:

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

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

```
$> preload -H nfkmfinfo
...
Pre-Loaded Objects ( 3): objecthash module objectid generation
29235f2a0b77fc1e18641b0820fe3c93e030a02e 4294967295 0x44313d41 1
5bccb6f540802ef1da3828f6b8b0f3fc985272e6 2 0x44313d47 1
5bccb6f540802ef1da3828f6b8b0f3fc985272e6 1 0x44313d46 1
...
$> nfkmfinfo -k simple simplesoftcard1
...
name                "simple_softcard1"
hash                 29235f2a0b77fc1e18641b0820fe3c93e030a02e
...
$> nfkmfinfo -s
...
Operator logical token hash      name
5bccb6f540802ef1da3828f6b8b0f3fc985272e6 softcard1
```

Second preload process started:

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

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

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

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

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

15.7.8. FIPS Auth in High Availability mode

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

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

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

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

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

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

15.7.9. PKCS #11 and JCE

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

15.7.9.1. Use PKCS #11 for key reloading

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

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

set or softcard protected keys), where `<IDENT>` is the identifier of the card set or soft-card, respectively.



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



Using preload in high availability mode with Operator Card Sets has a set of restrictions, see [Overview](#).

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

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

For more information, see section *PKCS_11 with key reloading* in the *Cryptographic API Integration Guide*.

15.7.10. Unsupported options

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

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

15.8. Logging

By default `preload` logs to `stderr`.

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

e.g.

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

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

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

The default file location for logs is `/opt/nfast/logs/preload_log_pid.log`.

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

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

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

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

15.9. Using preloaded objects - Worked example

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

Python:

```
import nfkm

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

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

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

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

Python:

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

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

Python:

```
import nfkm

conn = nfkm.connection(existingobjects="")

world_info = nfkm.getinfo(conn)

print world_info.existingobjects
```

Result:

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

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

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

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

print conn.transact(cmd)
```

Result:

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

16. Environment variables

This appendix describes the environmental variables used by Security World Software:

Variable	Description
<code>KERNEL_HEADERS</code>	This variable allows you to specify the path to kernel headers (if, for example, they are not in the default directory). It is necessary for the configuration script to be able to find the kernel headers when building the PCI driver during software installation.
<code>NFAST_CERTDIR</code>	This variable specifies the path to the dynamic feature enabling Feature Certificates directory. You only need to change the value of this variable if you move the Installation directory. See <code>NFAST_HOME</code> , <code>NFAST_KMDATA</code> , and <code>NFAST_LOGDIR</code> .
<code>NFAST_DEBUG</code>	This variable enables debug logging for the PKCS #11 library. You must set <code>NFAST_DEBUG</code> equal to a value in the range 1 – 7 for debug messages to be logged (see Logging, debugging, and diagnostics . For more information, see also Logging and debugging information for PKCS #11 .
<code>NFAST_DEBUGSYSLOG</code>	This variable redirects debug logging to syslog. The value of the environment variable should be one of the syslog facilities to be used. Prefixing the facility name with <code>+</code> enables traditional logging and syslog simultaneously.
<code>NFAST_HOME</code>	This variable specifies the path to the Installation directory, which is set by the Security World Software installation script. You only need to change the value of this variable if you move the Installation directory. See <code>NFAST_KMDATA</code> , <code>NFAST_CERTDIR</code> , and <code>NFAST_LOGDIR</code> .
<code>NFAST_KMDATA</code>	This variable sets the location of the Key Management Data directory. You only need to change the value of this variable if you move the Key Management Data directory. See <code>NFAST_HOME</code> , <code>NFAST_CERTDIR</code> , <code>NFAST_LOGDIR</code> , and <code>NFAST_KMLOCAL</code> .
<code>NFAST_KMLOCAL</code>	This variable specifies the location of the Key Management and Security World Data directory. If this environment variable is not set, by default the module looks for the Security World data in the <code>local</code> subdirectory of the Key Management Data directory. See <code>NFAST_KMDATA</code> .
<code>NFAST_LOGDIR</code>	This variable specifies the location of the Log Files directory. You only need to change the value of this variable if you move the Log Files directory. See <code>NFAST_HOME</code> , <code>NFAST_KMDATA</code> , and <code>NFAST_CERTDIR</code> .

Variable	Description
NFAST_USER_LOGDIR	This variable specifies the location of log files that are specific to each user.
NFAST_NFKM_TOKENSFILE NFAST_NFKM_TOKENSSELECT	This variable sets the default values for a file in which ClientID and KeyIDs are stored by the preload command-line utility.
NFAST_SEE_MACHINEENCKEY_DEFAULT	This variable is the name of the SEEConf key needed to decrypt SEE-machine images. Running the command loadmache --encryptionkey='IDENT (or 'loadmache --unencrypted) overrides any value set by this variable.
NFAST_SEE_MACHINEENCKEY_<module>	This variable is the name of the SEEConf key needed to decrypt the SEE-machine image targeted for the specified module. It overrides NFAST_SEE_MACHINEENCKEY_DEFAULT for the specified module. Running the command loadmache --encryptionkey=<IDENT> (or loadmache --unencrypted) overrides any value set by this variable.
NFAST_SEE_MACHINEIMAGE_DEFAULT	This variable is the path of the SEE machine image to load on to any module for which a specific image is not defined. Supplying the machine-filename parameter when running the loadmache command-line utility overrides this variable. This variable is not affected when running the loadsee-setup or hsc_loadseemachine utilities.
NFAST_SEE_MACHINEIMAGE_<module>	This variable is the path of the SEE machine image to load on to the specified module. If set, this variable overrides the use of NFAST_SEE_MACHINEIMAGE_DEFAULT for the specified module. Supplying the <machine-filename> parameter when running the loadmache command-line utility overrides the NFAST_SEE_MACHINEIMAGE_<module> variable. This variable is not affected when running the loadsee-setup or hsc_loadseemachine utilities.
NFAST_SEE_MACHINESIGHASH_DEFAULT	This variable is the default key hash of the vendor signing key (seeinteg) that signs SEE machine images. This variable is only required if you are using a dynamic SEE feature with an encrypted SEE machine. Running the command loadmache --sighash=<HASH> any value set in this variable.
NFAST_SEE_MACHINESIGHASH_<module>	This variable is the key hash of the vendor signing key (seeinteg) that signs SEE machine images for the specified module. It overrides NFAST_SEE_MACHINESIGHASH_DEFAULT for the specified module. This variable is only required if you are using a dynamic SEE feature with an encrypted SEE machine. Running the command loadmache --sighash=<HASH> any value set in this variable.

Variable	Description
NFAST_SERVER NFAST_PRIVSERVER	<p>If these variables are set in the hardserver's environment, the values specify the pathnames of the UNIX domain sockets that the hardserver uses for ordinary/privileged client connections to the hardserver.</p> <p>These variables are available for this purpose for backward compatibility only; you should configure sockets in the hardserver configuration file, see server_startup. If you set these variables they override the values in hardserver configuration file.</p>
NFAST_SERVER_PORT NFAST_SERVER_PRIVPORT	<p>If these variables are set in the hardserver's environment, the values specify the TCP port numbers that the nFast server uses for connections over TCP sockets.</p> <p>These variables are available for this purpose for backward compatibility only: you should configure ports in the hardserver configuration file, as described in server_startup. If you set these variables, they override the values in the hardserver configuration file.</p>
NFAST_SERVER_SYSLOG	<p>This variable sets server logging to syslog. The value of the environment variable should be one of the syslog facilities to be used. Prefixing the facility name with + enables traditional logging and syslog simultaneously.</p>
NFLOG_CATEGORIES	<p>This variable is used to filter log messages by supplying a colon-separated list of allowable message categories; see Logging, debugging, and diagnostics. If no value is supplied, all message categories are logged.</p>
NFLOG_SEVERITY	<p>This variable is used to filter log messages by supplying a minimum severity level to be logged; see Logging, debugging, and diagnostics. If no value is supplied, the default severity level is WARNING.</p>
NFLOG_DETAIL	<p>This variable is used to filter log messages by supplying a bit mask of detail flags; see Logging, debugging, and diagnostics. The default is time+severity+writeable.</p>
NFLOG_FILE	<p>This variable is used to specify a filename (or file descriptor) in which log messages are to be written; see Logging, debugging, and diagnostics. The default is stderr (the equivalent of file descriptor &2).</p>

17. Logging, debugging, and diagnostics

This appendix describes the settings and tools you can use to access the logging and debugging information generated by the Security World Software. You are also shown how to obtain system information using the `nfdiag` command-line utility.

17.1. Logging and debugging

The nShield Connect and the applications that use it generate log files. You can view log files using the front panel. Application log messages are handled on the client.



The current release of Security World Software uses controls for logging and debugging that differ from those used in previous releases. However, settings you made in previous releases to control logging and debugging are still generally supported in the current release, although in some situations the output is now formatted differently.



Some text editors, such as Notepad, can cause NFLOG to stop working if the NFLOG file is open at the same time as the hardserver is writing the logs.

17.1.1. Environment variables to control logging

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

NFLOG_FILE

This environment variable specifies the name of a file (or a file descriptor, if prefixed with the `&` character) to which logging information is written. The default is `stderr` (the equivalent of `&2`).

Ensure that you have permissions to write to the file specified by `NFLOG_FILE`.

NFLOG_SEVERITY

This environment variable specifies a minimum severity level for logging messages to be written (all log messages less severe than the specified level are ignored). The level can be one of (in order of greatest to least severity):

1. `FATAL`
2. `SEVERE`

3. **ERROR**
4. **WARNING**
5. **NOTIFICATION**
6. ``DEBUG`N` (where *N* can be an integer from 1 to 10 inclusive that specifies increasing levels of debugging detail, with 10 representing the greatest level of detail, although the type of output is depends on the application being debugged).



The increasingly detailed information provided by different levels of ``DEBUG`N` is only likely to be useful during debugging, and we recommend not setting the severity level to ``DEBUG`N` unless you are directed to do so by Support.

The default severity level is **WARNING**.

NFLOG_DETAIL

This environment variable takes a hexadecimal value from a bitmask of detail flags as described in the following table (the **logdetail** flags are also used in the `hardserver` configuration file to control `hardserver` logging; see: [server_settings](#)):

Hexadecimal flag	Function	logdetail flags
0x00000001	This flag shows the external time (that is, the time according to your machine's local clock) with the log entry. It is on by default.	external_time
0x00000002	This flag shows the external date (that is, the date according to your machine's local clock) with the log entry.	external_date
0x00000004	This flag shows the external process ID with the log entry.	external_pid
0x00000008	This flag shows the external thread ID with the log entry.	external_tid
0x00000010	This flag shows the external time_t (that is, the time in machine clock ticks rather than local time) with the log entry.	external_time_t
0x00000020	This flag shows the stack backtrace with the log entry.	stack_backtrace
0x00000040	This flag shows the stack file with the log entry.	stack_file
0x00000080	This flag shows the stack line number with the log entry.	stack_line

Hexadecimal flag	Function	logdetail flags
0x00000100	This flag shows the message severity (a severity level as used by the NFLOG_SEVERITY environment variable) with the log entry. It is on by default.	msg_severity
0x00000200	This flag shows the message category (a category as used by the NFLOG_CATEGORIES environment variable) with the log entry.	msg_categories
0x00000400	This flag shows message writeables, extra information that can be written to the log entry, if any such exist. It is on by default.	msg_writeable
0x00000800	This flag shows the message file in the original library. This flag is likely to be most useful in conjunction with Security World Software-supplied example code that has been written to take advantage of this flag.	msg_file
0x00001000	This flag shows the message line number in the original library. This flag is likely to be most useful in conjunction with example code we have supplied that has been written to take advantage of this flag.	msg_line
0x00002000	This flag shows the date and time in UTC (Coordinated Universal Time) instead of local time.	options_utc
0x00004000	This flag shows the full path to the file that issued the log messages.	options_fullpath
0x00008000	This flag includes the number of microseconds in the timestamp.	options_time_us
0x00010000	This flag enables logging of potentially secret values in generic stub log output.	msg_secrets

NFLOG_CATEGORIES

This environment variable takes a colon-separated list of categories on which to filter log messages (categories may contain the wild-card characters ***** and **?**). If you do not supply any values, then all categories of messages are logged. This table lists the available categories:

Category	Description
nflog	Logs all general messages relating to nflog.

Category	Description
nflog-stack	Logs messages from <code>StackPush</code> and <code>StackPop</code> functions.
memory-host	Logs messages concerning host memory.
memory-module	Logs messages concerning module memory.
gs-stub	Logs general generic stub messages. (Setting this category works like using the <code>dbg_stub</code> flag with the logging functionality found in previous Security World Software releases.)
gs-stubbignum	Logs bignum printing messages. (Setting this category works like using the <code>dbg_stubbignum</code> flag with the logging functionality found in previous Security World Software releases.)
gs-stubinit	Logs generic stub initialization routines. (Setting this category works like using the <code>dbg_stubinit</code> flag with the logging functionality found in previous Security World Software releases.)
gs-dumpenv	Logs environment variable dumps. (Setting this category works like using the <code>dbg_dumpenv</code> flag with the logging functionality found in previous Security World Software releases.)
nfkm-getinfo	Logs <code>nfkm-getinfo</code> messages.
nfkm-newworld	Logs messages about world generation.
nfkm-admin	Logs operations using the Administrator Card Set.
nfkm-kmdata	Logs file operations in the <code>kmdata</code> directory.
nfkm-general	Logs general NFKM library messages.
nfkm-keys	Logs key loading operations.
nfkm-preload	Logs <code>preload</code> operations.
nfkm-ppmk	Logs softcard operations.
serv-general	Logs general messages about the local hardserver.
serv-client	Logs messages relating to clients or remote hardservers.
serv-internal	Logs severe or fatal internal errors.
serv-startup	Logs fatal startup errors.
servdbg-stub	Logs all generic stub debugging messages.
servdbg-env	Logs generic stub environment variable messages.
servdbg-underlay	Logs messages from the OS-specific device driver interface
servdbg-statemach	Logs information about the server's internal state machine.
servdbg-perf	Logs messages about the server's internal queuing.

Category	Description
servdbg-client	Logs external messages generated by the client.
servdbg-messages	Logs server command dumps.
servdbg-sys	Logs OS-specific messages.
pkcs11-sam	Logs all security assurance messages from the PKCS #11 library.
pkcs11	Logs all other messages from the PKCS #11 library.
rqcard-core	Logs all card-loading library operations that involve standard message passing (including slot polling).
rqcard-ui	Logs all card-loading library messages from the current user interface.
rqcard-logic	Logs all card-loading library messages from specific logics.

You can set a minimum level of hardserver logging by supplying one of the values for the **NFLOG_SEVERITY** environment variable in the hardserver configuration file, and you can likewise specify one or more values for the **NFLOG_CATEGORIES** environment variable. For detailed information about the hardserver configuration file settings that control logging, see [server_settings](#).



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

17.1.2. Logging and debugging information for PKCS #11

In order to get PKCS #11 logging and debugging output, you must set the **CKNFAST_DEBUG** variable. A debug output file (with path) can also be set using the **CKNFAST_DEBUGFILE** variable. These variables can be set in the environment or **/opt/nfast/cknfastrc** file. Normally settings in the environment should override the same settings (if any) in the **cknfastrc** file. You can specify any appropriate PKCS #11 categories using the **NFLOG_CATEGORIES** environment variable.

To produce PKCS #11 debug output, the **CKNFAST_DEBUG** variable can be a given value from 1 through to 11, where the greater the value the more detailed debug information is provided. A value of 7 is a reasonable compromise between too little and too much debug information. A value of 0 switches the debug output off.

This environment variable takes a colon-separated list of categories on which to filter log messages (categories may contain the wildcards characters ***** and **?**).

The following table maps PKCS #11 debug level numbers to the corresponding **NFLOG_SEVERITY** value:

PKCS #11 debug level	PKCS #11 debug meaning	NFLOG_SEVERITY value	Output in log
0	DL_None	NONE	
1	DL_EFatal	FATAL	"Fatal error:"
2	DL_EError	ERROR	"Error:"
3	DL_Fixup	WARNING	"Fixup:"
4	DL_Warning	WARNING	"Warning:"
5	DL_EApplic	ERROR	"Application error:"
6	DL_Assumption	NOTIFICATION	"Unsafe assumption:"
7	DL_Call	DEBUG2	">> "
8	DL_Result	DEBUG3	"< "
9	DL_Arg	DEBUG4	"> "
10	DL_Detail	DEBUG5	"D "
11	DL_DetailMutex	DEBUG6	"DM "

17.1.3. Debugging information for Java

This section describes how you can specify the debugging information generated by Java.

17.1.3.1. Setting the Java debugging information level

In order to make the Java generic stub output debugging information, set the Java property **NFJAVA_DEBUG**. The debugging information for **NFJAVA**, **NFAST**, and other libraries (for example, **KMJAVA**) can all use the same log file and have their entries interleaved.

You set the debugging level as a decimal number. To determine this number:

1. Select the debugging information that you want from the following list:

```
NONE = 0x00000000 (debugging off)
MESS_NOTIFICATIONS = 0x00000001 (occasional messages including important errors)
MESS_VERBOSE = 0x00000002 (all messages)
MESS_RESOURCES = 0x00000004 (resource allocations)
FUNC_TRACE = 0x00000008 (function calls)
FUNC_VERBOSE = 0x00000010 (function calls + arguments)
REPORT_CONTEXT = 0x00000020 (calling context e.g ThreadID and time)
FUNC_TIMINGS = 0x00000040 (function timings)
NFJAVA_DEBUGGING = 0x00000080 (Output NFJAVA debugging info)
```

2. Add together the hexadecimal value associated with each type of debugging information.

For example, to set `NFJAVA_DEBUGGING` and `MESS_NOTIFICATIONS`, add `0x00000080` and `0x00000001` to make `0x00000081`.

3. Convert the total to a decimal and specify this as the value for the variable.

For example, to set `NFJAVA_DEBUGGING` and `MESS_NOTIFICATIONS`, include the line:

```
NFJAVA_DEBUG=129
```

For `NFJAVA` to produce output, `NFJAVA_DEBUG` must be set to at least `NFJAVA_DEBUGGING + MESS_NOTIFICATIONS`. Other typical values are:

- **255**: All output
- **130**: nfjava debugging and all messages (`NFJAVA_DEBUGGING` and `MESS_VERBOSE`)
- **20**: function calls and arguments and resource allocations (`FUNC_VERBOSE` and `MESS_RESOURCES`)

17.1.3.2. Setting Java debugging with the command line

You can set the Java debug options by immediately preceding them with a `-D`. Use the `NJAVA_DEBUGFILE` property to direct output to a given file name, for example:

```
java -DNFJAVA_DEBUGFILE=myfile -DNFJAVA_DEBUG=129 -classpath .... classname
```



Do not set `NFJAVA_DEBUG` or `NFJAVA_DEBUGFILE` in the environment because Java does not pick up variables from the environment.

If `NFJAVA_DEBUGFILE` is not set, the standard error stream `System.err` is used.



Set these variables only when developing code or at the request of Support.



Debug output contains all commands and replies sent to the hardserver in their entirety, including all plain texts and the corresponding cipher texts as applicable.

17.2. Diagnostics and system information



Besides the diagnostic tools described in this section, we also supply a

performance tool that you can use to test Web server performance both with and without an nShield HSM. This tool is supplied separately. If you require a copy, contact your Sales representative.

17.2.1. nfdiag: diagnostics utility

The **nfdiag** command-line utility is a diagnostics tool that gathers information about the system on which it is executed. It can save this information to either a **.zip** file or a text file.

Under normal operating conditions, you do not need to run **nfdiag**. You can run **nfdiag** before contacting Support and include its output file with any problem report.

17.2.1.1. Usage



Run **nfdiag** with the standard **-h|--help** option to display information about the options and parameters that control the program's behavior.

The **nfdiag** command-line utility is an interactive tool. When you run it, it prompts you to supply the following information:

Option	Actions to take
which application(s) you are using	Identify all application software installed on the machine on which any problem with the nShield product occurs.
what APIs you are using	Describe any custom software, especially any interaction it has with the nShield security system.
a description of the problem	Include as much detail as possible, including any error messages you have seen.
a Support ticket number (if you have one)	When you contact Support you are supplied with a Support ticket number. Enter this number to help Support expedite the collection of any information you have sent.
a contact email address	Supply an email address that has as few e-mail/spam filters as possible so that any additional files that Support sends to you are not blocked. We use the e-mail address you supply here <i>only</i> for communication directly related to your problem report.
a contact name	Enter your name (or the name of an appropriate person for contact by Support).
a contact telephone number	Include the appropriate country and any region code for your contact telephone number.

Option	Actions to take
any additional diagnostic file(s)	<p>By default, nfdiag asks if you want to supply any additional diagnostic files. If you do not want to supply additional diagnostic files, type N, or Enter to continue.</p> <p>If you want to supply additional diagnostic files:</p> <ul style="list-style-type: none"> • Type Y. • When prompted, supply the name of the file to be attached. Attached files must be in plain text format. • After you have supplied the name of a file to attach, nfdiag asks again if you want to supply any additional diagnostic files. Repeat the process to attach additional files. • When you do not want to attach any more files, type N, or Enter to continue.



Except for a Support ticket number, **nfdiag** requires non-NULL answers to all its prompts for information.

Supplying this information helps **nfdiag** capture as much relevant information as possible for any problem report to Support. As you supply information at each prompt in turn, press **Enter** to confirm the information and continue to the next prompt. Information you supply cannot extend over multiple lines, but if you need to supply this level of information, you can include it in additional attached files, as described.

By default, **nfdiag** runs in verbose mode, providing feedback on each command that it executes and which log files are available. If the system is unable to execute a command, the verbose output from **nfdiag** shows where commands are stalling or waiting to time out.

At any time while **nfdiag** is running, you can type **Ctrl-C** to cancel its current commands and re-run it.

17.2.1.2. Output

After you have finished supplying information for each required prompt, **nfdiag** generates a plain text output file and displays its file name.

If the file **opt/nfast/log/logfile** exists, **nfdiag** automatically includes this file in its output. If the file **opt/nfast/log/logfile** does not exist, **nfdiag** warns you that it could not process this file. This warning does not affect the validity of the generated output file.

When complete, this output file contains the following:

- The information supplied interactively to **nfdiag** when run

- Details about the client machine
- Details about any environment variables
- Output from the following command-line utilities:
 - `enquiry`
 - `stattree`
 - `ncversions`
 - `nfkminfo`
- The contents of the following log files (if they are available):
 - `hardserver.log`
 - `keysafe.log`
 - `cmdadp.log`
 - `ncsnmpd.log`
- Any attached diagnostic files

Because the contents of the output file are plain text, they are human readable. You can choose to view the output file to ensure no sensitive information has been included.



The `nfdiag` utility does not capture any pass phrases in the output file.

17.2.1.3. nShield Connect tamper log

The nShield Connect tamper log contains:

- The date and time of any tamper events
- Whether the tamper detection functionality has been disabled or re-enabled.

You view the tamper log using the nShield Connect front panel, by selecting **System > System information > View tamper log**. You cannot erase the tamper log.

17.2.2. nfkminfo: information utility

The `nfkminfo` utility displays information about the Security World and the keys and card sets associated with it.

17.2.2.1. Usage

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

```
nfkminfo -k|--key-list [<APPNAME> [<IDENT>]]
```

```
nfkminfo -l|--name-list [<APPNAME> [<APPNAME>...]]
```

```
nfkminfo [-c|--cardset-list][[-s|--softcard-list] [<TOKENHASH>]
```

```
nfkminfo --cardset-list [<TOKENHASH>] --key-list [<APPNAME>[<APPNAME>]]|--name-list <APPNAME>[<IDENT>...]]
```

17.2.2.1.1. Security World options

-w|--world-info

This option specifies that you want to display general information about the Security World. These options are the default and need not be included explicitly.

-r|--repeat

This option displays the information repeatedly. There is a pause at the end of each set of information. The information is displayed again when you press Enter.

-p|--preload-client-id

This option displays the preloaded client ID value, if any.

17.2.2.1.2. Key, card set, and softcard options

-k|--key-list [<APPNAME>[<APPNAME>]]

This option lists keys without key names. If **<APPNAME>** is specified, only keys for these applications are listed.

-l|--name-list [<APPNAME>[<IDENT>]]

This option lists keys with their names. If **<APPNAME>** is specified, only keys for these applications are listed. If **<IDENT>** is listed, only the keys with the specified identifier are listed.

-c|--cardset-list [<TOKENHASH>]

If **<TOKENHASH>** is not specified, this option lists the card sets associated with the Security World. The output is similar to this:

```
Cardset list - 1 cardsets: (P)ersistent/(N)ot, (R)emoteable/(L)ocal-only
Operator logical token hash      k/n timeout name <hash>          1/1 none-PL <name>
```

If **<TOKENHASH>** is specified, these options list the details of the card identified by *hash*. The output is similar to this:

```
Cardset
name      "name"
k-out-of-n 1/1
flags     Persistent PINRecoveryForbidden(disabled) !RemoteEnabled
timeout   none
card names ""
hkltu     hash
gentime 2005-10-14 10:56:54
Keys protected by cardset hash:
AppName app Ident keyident
AppName app Ident keyident
...      ...      ...      ...
```

-s|--softcard-list **TOKENHASH**

This option works like the **-c|--cardset-list** option, except it lists softcards instead of card sets. If **<TOKENHASH>** is not specified, this option lists the softcards associated with the Security World.

17.2.2.2. Security World output info

If you run **nfkminfo** with the **-w|--world-info** option, it displays information similar to that shown in these examples:

```
generation 1
state      0x70000 Initialised Usable Recovery !PINRecovery
!ExistingClient !RTC !NVRAM !FTO !SEEDebug
n_modules  1
hkns0      hash_kns0
hkm         hash_km
hkmwk      hash_knwk
hkcre      hash_kre
hkra       hash_kra
ex.client  none
...
```

```
...
Module #1
generation 1
state      0x1 Useable
flags      0x10000 ShareTarget
n_slots    2
esn        34F3-9CB4-753B
hkm1       hash_km1
Module #1 Slot #0 IC 11
generation 1
phystype    SmartCard
slotlistflags 0x2
state       0x4 Operator
flags       0x20000 RemoteEnabled
shareno     2
```



```
shares
error      OK
Cardset
name       "fred"
k-out-of-n 1/2
flags      NotPersistent
timeout     none
card names "" ""
hkltu      hash_kt
Module #1 Slot #1 IC 0
generation 1
phystype    SmartCard
slotlistflags 0x2 SupportsAuthentication
state       0x4 Admin
flags       0x10000 Pass phrase
shareno     1
shares      LTNso(PIN) LTM(PIN) LTR(PIN) LTNV(PIN) LTRTC(PIN) LTDSEE(PIN)
LTFTO(PIN)
error      OK
No Cardset

No Pre-Loaded Objects
```

17.2.2.2.1. World

nfkminfo reports the following information about the Security World:

generation

This indicates the internal number.

state

This indicates the status of the current world:

Initialised	This indicates that the Security World has been initialized.
Usable	This indicates that there is at least one usable HSM in this Security World on this host.
!Usable	This indicates that there are no usable HSMs in this Security World on this host.
Recovery	This indicates that the Security World has the OCS and softcard replacement and the key recovery features enabled.
!Recovery	This indicates that the Security World has the OCS and softcard replacement and the key recovery features disabled.

AdminAuthRequired	This indicates that additional authorization is required for the following operations: <ul style="list-style-type: none"> • Key generation • Public key import • Operator cardset creation • Softcard creation. This authorization is supplied by presenting any operator or administration card from the same Security World. A pass phrase is not required:
ExistingClient	This indicates that there is a Client ID set, for example, by preload . This Client ID is given in the ex.client output if the --preload-client-id flag was supplied.
!ExistingClient	This indicates that no Client ID is set. The ex.client output will be empty.
AlwaysUseStrongPrimes	This indicates that the Security World always generates RSA keys in a manner compliant with FIPS 186-3.
!AlwaysUseStrongPrimes	This indicates that the Security World leaves the choice of RSA key generation algorithm to individual clients.
SEEDebug	This indicates that the Security World has an SEE Debugging delegation key.
!SEEDebug	This indicates the Security World has no SEE Debugging delegation key.
SEEDebugForAll	This indicates no authorization is required for SEE Debugging.
PINRecovery	This indicates that the Security World has the pass phrase replacement feature enabled.
!PINRecovery	This indicates that the Security World has the pass phrase replacement feature disabled.
FTO	This indicates that the Security World has an FTO delegation key.
!FTO	This indicates that the Security World has no FTO delegation key.
NVRAM	This indicates that the Security World has an NVRAM delegation key.
!NVRAM	This indicates that the Security World has no NVRAM delegation key.
RTC	This indicates that the Security World has an RTC delegation key.
!RTC	This indicates that the Security World has no RTC delegation key.
AuditLogging	This indicates that Audit Logging is enabled for this Security World.
!AuditLogging	This indicates that Audit Logging is not enabled for this Security World.

n_modules

This indicates the number of nShield HSMs connected to this computer.

hknso

This indicates the SHA-1 hash of the Security Officer's key.

hkm

This indicates the SHA-1 hash of the Security World key.

hkmwk

This indicates the SHA-1 hash of a dummy key used to load the Administrator Card Set (the dummy key is the same on all HSMs that use Security Worlds and is not secret).

hkre

This indicates the SHA-1 hash of the recovery key pair.

hkra

This indicates the SHA-1 hash of the recovery authorization key.

ex.client

This indicates the **ClientID** required to use any pre-loaded keys and tokens.

k-out-of-n

This indicates the values of *K* and *N* for this Security World.

other quora

This indicates the number (quora) of Administrator Cards (*K*) required to perform certain other functions as configured for this Security World.

ciphersuite

This indicates the name of the Cipher suite that the Security World uses.

Mode

none	This indicates that the Security World is in an unregulated mode. The Security World can be configured to meet the needs of your security policy. This includes, but is not limited to, creating a Security World that is compliant with FIPS140-2 Level 2.
fips1402level3	This indicates that the Security World is in a mode compliant with FIPS 140-2 Level 3.
commoncriteriaacmts	This indicates that the Security World is in a mode compliant with Common Criteria Protection Profile EN 419 221-5, for Cryptographic Modules for Trust Services.

Assigned Keys

<code>max usage</code>	This indicates the maximum key usage reauthorization condition for Assigned Keys. (common-criteria-cmts mode only).
<code>max timeout</code>	This indicates the maximum key timeout reauthorization condition for Assigned Keys (common-criteria-cmts mode only).

17.2.2.2.2. Module

For each HSM in the Security World, `nfkminfo` reports:

`generation`

This indicates the version of the HSM data.

`state`

This indicates one of the following:

<code>PreInitMode</code>	This indicates that the HSM is in the pre-initialization state.
<code>InitMode</code>	This indicates that the HSM is in the initialization state.
<code>Unknown</code>	This indicates that the HSM's state could not be determined.
<code>Usable</code>	This indicates that the HSM is programmed in the current Security World and can be used.
<code>Uninitialized</code>	This indicates that the HSM does not have the Security Officer's key set and that the HSM must be initialized before use.
<code>Factory</code>	This indicates that the HSM has module key zero only and that the Security Officer's key is set to the factory default.
<code>Foreign</code>	This indicates that the HSM is from an unknown Security World.
<code>AccelOnly</code>	This indicates that the HSM is acceleration only.
<code>Unchecked</code>	This indicates that, although the HSM appears to be in the current Security World, <code>nfkminfo</code> could not find a module initialization certificate (a <code>module_<ESN></code> file) for this HSM.
<code>Failed</code>	This indicates that the HSM has failed. For internal security modules running firmware enquiry utility for further information about the failure reason, or look for hardware errors in the hardserver log.
<code>MaintMode</code>	This indicates that the HSM is in the maintenance state.

`flags`

This displays `ShareTarget` if the HSM has been initialized to allow reading of remote card

sets.

`n_slots`

This indicates the number of slots on the HSM (there is one slot for each physical smart card reader, one slot for each soft token, one slot for each available Remote Operator slot and one slot for each associated Dynamic Slots).

`esn`

This indicates the electronic serial number of the HSM (if the HSM is not in the `Usable` state, the electronic serial number may not be available).

`hkml`

This indicates the hash of the HSM signing key (if the HSM is not in the `Usable` state, this value may not be available).

17.2.2.2.3. Slot

For each slot on the HSM, `nfkminfo` reports:

`IC`

This indicates the insertion count for this slot (which is 0 if there is no card in the slot).

`generation`

This indicates the version of the `slotinfo` structure.

`phystype`

This indicates the type of slot, which can be one of:

- `SmartCard`
- `SoftToken`

`slotlistflags`

These are flags describing the capabilities of the slot. Single letters in parentheses are the flag codes reported by the `slotinfo` utility:

<code>0x2</code>	<p>(A) <code>SupportsAuthentication</code></p> <p>This indicates that the slot supports token-level challenge-response authentication.</p>
<code>0x40000</code>	<p>(R) <code>RemoteSlot</code></p> <p>This indicates that the slot is a Remote Operator slot that has been imported from a remote HSM.</p>

0x80000	(D) DynamicSlot This indicates that it is a Dynamic Slot.
0x100000	(a) Associated This indicates that a Remote Administration Client has associated a card reader with this
0x200000	(t) TimedOut This indicates that no response has been received from the smartcard in this Dynamic Slot within the configured timeout.
0x400000	(f) SecureChannelFailed This indicates that the secure channel between the HSM and the smartcard in this Dynamic Slot has failed in some way.

state

This can be one or more of the following flags:

Blank	This indicates that the smart card in the reader is unformatted.
Admin	This indicates that the smart card in the reader is part of the Administrator Card Set.
Empty	This indicates that there is no smart card in the reader.
Error	This indicates that the smart card in the reader could not be read (the card may be from a different Security World).
Operator	This indicates that the smart card in the reader is an Operator Card.

flags

This displays **pass phrase** if the smart card requires a pass phrase.

shareno

This indicates the number of the card within the card set.

shares

If the card in the slot is an Operator Card, no values are displayed for **shares**.

If the card in the slot is an Administrator Card, values are displayed indicating what key shares are stored on the card. Each share is prefixed with the letters **LT** (Logical Token), and the remaining letters identify the key (for example, the value **LTNSO** indicates that a share of K_{NSO} , the Security Officer's key, is stored on the card).

error

This indicates the error status encountered if the smart card could not be read:

OK	This indicates that there were no errors.
TokenAuthFailed	This indicates that the smart card in the reader failed challenge response authentication (the card may come from a different Security World).
PhysTokenNotPresent	This indicates that there is no card in the reader.

If you purchased a developer kit, you can refer to the relevant developer documentation for a full list of error codes.

17.2.2.2.4. Card set

If there is an Operator Card in the reader, `nfkminfo` reports:

`name`

This indicates the name given to this card set.

`k-out-of-n`

This indicates the values of *K* and *N* for this card.

`flags`

This displays one or more of each of the following pairs of flags:

NotPersistent	This indicates that the Operator Card is not persistent.
Persistent	This indicates that the Operator Card is persistent.
NotRemoteEnabled	This indicates that the card in the slot is not from a Remote Operator Card Set.
RemoteEnabled	This indicates that the card in the slot is from a Remote Operator Card Set.
PINRecoveryForbidden(disabled)	This indicates that the card in the slot does not have pass phrase replacement enabled. This is always true if pass phrase replacement is disabled for the Security World.
PINRecoveryRequired(enabled)	This indicates that the card in the slot does have pass phrase replacement enabled.

`timeout`

the period of time in seconds after which the HSM automatically removes the Operator Card Set. If timeout is set to `none`, the Operator Card Set does not time out.

`card`

lists the names of the cards in the set, not all software can give names to individual cards in a set.

hk1tu

the SHA-1 hash of the secret on the card.

17.2.3. perfcheck: performance measurement checking tool

Use the **perfcheck** command-line utility to run various tests measuring the cryptographic performance of an nShield HSM.



Run **perfcheck** with the standard **-h|--help** option to display information about the options and parameters that control the program's behavior.

The available tests are grouped into suites:

- **kx** (key exchange)
- **keygen** (key generation)
- **signing** (signing)
- **verify** (verification)
- **enc** (encryption)
- **dec** (decryption)
- **misc** (miscellaneous).

To see the list of tests available in a particular suite, run a command of the form:

```
perfcheck --list suite
```

For example, to list all the **signing** tests, run the command:

```
perfcheck --list signing
>>> Suite 'signing' -- Signing (222 tests)
>>> 1 - DSA using RIPEMD160 with 512-bit p and 160-bit q.
>>> 2 - DSA using RIPEMD160 with 1024-bit p and 160-bit q.
>>> 3 - DSA using RIPEMD160 with 2048-bit p and 160-bit q.
>>> 4 - DSA using RIPEMD160 with 3072-bit p and 160-bit q.
>>> ...
```

In the output, each listed test in the suite is identified with a number.

You can reference a test either by its number or by its name:

- by test number:

```
perfcheck suite:test_number
```


To use test 16 of the `signing` suite:

```
perfcheck signing:16
```

- by test name:

```
perfcheck "exact name"
```

Example:

```
perfcheck "signing:RSA using RSAPKCS1 with 2048-bit n."
```

The test numbers change between releases. If you want to rerun tests for comparison, reference the tests by their names.

`perfcheck` prints the results of individual tests to output as it goes along, and then prints a full report at the end. By default, `perfcheck` runs each test three times for both minimum and maximum queue sizes, and then collates the results in the final report. See `--help` for the options to adjust this behavior.

Optionally, `perfcheck` can write its output to a directory in multiple formats using the `--out putdir` option to specify a directory name. This will create a new subdirectory under the specified directory to write the output. The `--nosubdir` option can be added as well to write output to the specified directory directly, in which case that directory must not already exist. The output directory will contain `perfcheck.html`, `perfcheck.txt`, `perfcheck.csv`, and `perfcheck.json` files that contain the report in HTML, text, CSV, and JSON format respectively. JSON files that contain the detailed results of individual tests will also be written to the output directory.

Output reports from test suites include the following information about each test:

Value	Description
Queue	<p>This value is the number of outstanding jobs in the queue when the test was run.</p> <p>By default, most tests run both with a queue of 1, and with a fully maxed out module queue, to give an indication of both one-at-a-time performance and the bandwidth for the operation. The queue can be set differently using the <code>--queue</code> option, in which case only that queue length will be run with, except for some <code>misc</code> suite tests which set their own queue.</p>

Value	Description
Rate (Units/s)	<p>This value is a measure of throughput. It is calculated by dividing the number of repetitions by total time.</p> <p>If a test has been rerun to improve accuracy, as is the case by default, then this is the mean across all the runs.</p> <p>Some tests, for example enc, set the Unit to something other than an operation, for example KB, to indicate the amount of data that can be encrypted.</p>
Min latency (ms)	This value is the time in milliseconds that the quickest individual job across all the test runs took to round-trip.
Mean latency (ms)	<p>This value is the mean time in milliseconds that jobs took to round-trip.</p> <p>If a test has been rerun, this is the mean of the mean latency values from each run.</p>
Max latency (ms)	This value is the time in milliseconds that the slowest individual job across all the test runs took to round-trip.
CV (%)	<p>This value is the coefficient of variation expressed as a percentage of the mean latency. It gives an indication of the variability in the time it takes individual jobs to complete.</p> <p>If a test has been rerun, this is the mean of the CV (%) values from each run.</p>
Min rate (tps)	<p>This is the estimated lower bound of the throughput for this queue size in transactions per second.</p> <p>The value becomes more accurate if more test runs of the same test are done. When it is compared against Mean rate (tps) and Max rate (tps), Min rate (tps) gives an indication of the variability between runs.</p>
Mean rate (tps)	<p>This is a measure of throughput. Unlike Rate (Units/s), it is expressed in transactions per second, that is, as the number of jobs that round-trip per second.</p> <p>Mean rate (tps) is included for comparison against the Min rate (tps) and Max rate (tps) figures.</p>
Max rate (tps)	<p>This is the estimated upper bound of the throughput for this queue size in transactions per second.</p> <p>The value becomes more accurate if more test runs of the same test are done. When it is compared against Min rate (tps) and Mean rate (tps), Max rate (tps) gives an indication of the variability between runs.</p>

Value	Description
Reps	<p>This value is the number of repetitions that were actually carried out, that is, the number of jobs that were round-tripped over all tests of this operation for this queue size.</p> <p>If a test was rerun, this is the sum of the repetitions for each run. The target repetitions for an individual run can be set using the <code>--repetitions</code> option but note that in most cases more repetitions will be run depending on the <code>--accuracy</code> setting provided that the timeout is not reached. It is recommended to set <code>--accuracy</code> rather than <code>--repetitions</code> to control the accuracy of the test instead of adjusting the repetitions.</p>

17.2.3.1. How perfcheck calculates statistics

When an nCore command is submitted to an HSM by a client application, it is processed as follows:

1. The command is passed to the client hardserver.
2. The client hardserver encrypts the command.
3. The encrypted command is sent to the unit hardserver over the network.
4. The unit hardserver decrypts the command and queues it.
5. When the internal security module is free, the command is submitted from the hardserver queue.
6. The command is executed by the HSM, and the reply is given to the unit hardserver.
7. The unit hardserver encrypts the command.
8. The unit hardserver sends the command back to the client hardserver over the network.
9. The client hardserver decrypts the reply and queues it.
10. When the client application is ready, the queued reply is returned to it.

Because an HSM can execute several commands at once, throughput is maximized by ensuring there is always at least one command in the unit hardserver queue (so that there are always commands available to give to the HSM).

The `perfcheck` utility sends multiple simultaneous nCore commands to keep the HSM busy. It can send more commands if a required number of repetitions has not yet been reached.

After sending some initial commands, `perfcheck` begins marking commands with the time at which are submitted; when a command comes back with a timestamp, `perfcheck` checks the amount of time needed to complete the command and updates the values for `Std dev` and `Latency`. The value of `Total time` is the amount of time from sending the first job to

receiving the final one.

17.2.4. stattree: information utility

The **stattree** utility returns the statistics gathered by the hardserver and HSMs.

17.2.4.1. Usage

```
stattree [<node> [<node> [...]]]
```

17.2.4.2. Output

Running the **stattree** utility displays a snapshot of statistics currently available on the host machine. Statistics are gathered both by the hardserver (relating to the server itself, and its current clients) and by each attached HSM.

Times are listed in seconds. Other numbers are integers, which are either real numbers, IP addresses, or counters. For example, a result **-CmdCount 74897** means that there have been 74,897 commands submitted.

A typical fragment of output from **stattree** looks like this:

```
+PerModule:
  +#1:
    +ModuleObjStats:
      -ObjectCount      5
      -ObjectsCreated   5
      -ObjectsDestroyed  0
    +ModuleEnvStats:
      -MemTotal          15327232
      -MemAllocKernel    126976
      -MemAllocUser      0
    +ModuleJobStats:
      -CmdCount          169780
      -ReplyCount        169778
      -CmdBytes          3538812
      -ReplyBytes        4492764
      -HostWriteCount    169772
      -HostWriteErrors   0
      -HostReadCount     437472
      -HostReadErrors    0
      -HostReadEmpty     100128
      -HostReadDeferred  167578
      -HostReadTerminated 0
      -PFNIssued         102578
      -PFNRejected       1
      -PFNCompleted      102577
      -ANIssued          1
      -CPULoadPercent    0
```

PerModule, **ModuleObjStats**, and **ModuleEnvStats** are node tags that identify classes of statistics. **1** identifies an instance node.

ObjectCount, **MemTotal**, and the remaining items at the same level are statistics **IDs**. Each has a corresponding value.

If **<node>** is provided, **stattree** uses the value given as the starting point of the tree and displays only information at or below that node in the tree. Values for **<node>** can be numeric or textual. For example, to view the object counts for local module number 3:

```
$ stattree PerModule 3 ModuleObjStats
+#PerModule:
  +#3:
    +ModuleObjStats:
      -ObjectCount      6
      -ObjectsCreated    334
      -ObjectsDestroyed  328
```

The value of **<node>** must be a node tag; it must identify a node in the tree and not an individual statistic. Thus, the following command does not work:

```
$ stattree PerModule 3 ModuleObjStats ObjectCount
+#PerModule:
  +#3:
    +ModuleObjStats:
      Unable to convert 'ObjectCount' to number or tag name.
```

17.2.4.2.1. Node tags

These hold statistics for each HSM:

Category	Contains
ModuleJobStats	This tag holds statistics for the Security World Software commands (jobs) processed by this HSM.
ModulePCISStats	This tag does not apply to an nShield Edge.
ServerGlobals	Aggregate statistics for all commands processed by the hardserver since it started. The standard statistics (as described below) apply to the commands sent from the hardserver to HSMs. Commands processed internally by the server are not included here. The Uptime statistic gives the total running time of the server so far.

Category	Contains
Connections	Statistics for connections between clients and the hardserver. There is one node for each currently active connection. Each node has an instance number that matches the log message generated by the server when that client connected. For example, when the hardserver message is Information: New client #24 connected , the client's statistics appear under node #24 in the stattree output.
PerModule	Statistics kept by the HSMs. There is one instance node for each HSM, numbered using the standard HSM numbering. The statistics provided by each HSM depend on the HSM type and firmware version.
ModuleObjStats	Statistics for the HSM's Object Store, which contains keys and other resources. These statistics may be useful in debugging applications that leak key handles, for example.
ModuleEnvStats	General statistics for the HSM's operating environment.
HostEnvStats	Environmental statistics for the HSM.
HostSysInfo	Further statistics for the HSM.

17.2.4.2.2. Statistics IDs

ID	Value
Uptime	The length of time (in seconds) since an HSM was last reset, the hardserver was started, or a client connection was made.
CmdCount	The total number of commands sent for processing from a client to the server, or from the server to an HSM. Contains the number of commands currently being processed.
ReplyCount	The total number of replies returned from server to client, or from HSM to server.
CmdBytes	The total length of all the command blocks sent for processing.
ReplyBytes	The total length of all the reply blocks received after completion.
CmdMarshalErrors	The number of times a command block was not understood when it was received. A nonzero value indicates either that the parties at each end of a connection have mismatched version numbers (for example, a more recent hardserver has sent a command to a less recent HSM that the HSM does not understand), or that the data transfer mechanism is faulty.
ReplyMarshalErrors	The number of times a reply was not understood when it was received. A nonzero value indicates either that the parties at each end of a connection have mismatched version numbers (for example, a more recent hardserver has sent a command to a less recent HSM that the HSM does not understand), or that the data transfer mechanism is faulty.

ID	Value
ClientCount	The number of client connections currently made to the server. This appears in the hardserver statistics.
MaxClients	The maximum number of client connections ever in use simultaneously to the hardserver. This gives an indication of the peak load experienced so far by the server.
DeviceFails	The number of times the hardserver has declared a device to have failed. The hardserver provides a diagnostic message when this occurs.
DeviceRestarts	The number of times the hardserver has attempted to restart an HSM after it has failed. The hardserver provides a Notice message when this occurs. The message does not indicate that the attempt was successful.
QOutstanding	The number of commands waiting for an HSM to become available on the specified client connection. When an HSM accepts a command from a client, this number decreases by 1 and DevOutstanding increases by 1. Commands that are processed purely by the server are never included in this count.
DevOutstanding	The number of commands sent by the specified client that are currently executing on one or more HSMs. When an HSM accepts a command from a client, this number increases by 1 and QOutstanding decreases by 1. Commands that are processed purely by the server are never included in this count.
LongOutstanding	The number of LongJobs sent by the specified client that are currently executing on one or more HSMs. When an HSM accepts a LongJobs command from a client, this number increases by 1 and QOutstanding decreases by 1. Commands that are processed purely by the server are never included in this count.
RemoteIPAddress	The remote IP address of a client who has this connection. A local client has the address 0.0.0.0.
HostWriteCount	The number of write operations (used to submit new commands) that have been received by the HSM from the host machine. One write operation may contain more than one command block. The operation is most efficient when this is the case.
HostWriteErrors	The number of times the HSM rejected the write data from the host. A nonzero value may indicate that data is being corrupted in transfer, or that the hardserver/device driver has got out of sync with the HSM's interface.
HostWriteBadData	Not currently reported by the HSM. Attempts to write bad data to the HSM are reflected in HostWriteErrors .
HostWriteOverruns	Not currently reported by the HSM. Write overruns are reflected in HostWriteErrors .
HostWriteNoMemory	Not currently reported by the HSM. Write failures due to a lack of memory are reflected in HostWriteErrors .

ID	Value
HostReadCount	The number of times a read operation to the HSM was attempted. The HSM can defer a read if it has no replies at the time, but expects some to be available later. Typically the HSM reports HostReadCount in two places: the number under ModuleJobStats counts a deferred read twice, once when it is initially deferred, and once when it finally returns some data. The number under ModulePCISStats counts this as one operation.
HostReadErrors	The number of times a read to an HSM failed because the parameters supplied with the read were incorrect. A nonzero value here typically indicates some problem with the host interface or device driver.
HostReadEmpty	The number of times a read from the HSM returned no data because there were no commands waiting for completion. In general, this only happens infrequently during HSM startup or reset. It can also happen if PauseForNotifications is disabled.
HostReadUnderruns	Not currently reported by the HSM.
HostReadDeferred	The number of times a read operation to the HSM was suspended because it was waiting for more replies to become available. When the HSM is working at full capacity, a sizeable proportion of the total reads are likely to be deferred.
HostReadTerminated	The number of times an HSM had to cancel a read operation which has been deferred. This normally happens only if the clear key is pressed while the HSM is executing commands. Otherwise it might indicate a device driver, interface, or firmware problem.
PFNIssued	The number of PauseForNotifications commands accepted by the HSM from the hardserver. This normally increases at a rate of roughly one every two seconds. If the hardserver has this facility disabled (or a very early version), this does not occur.
PFNRejected	The number of PauseForNotifications commands rejected by the HSM when received from the hardserver. This can happen during HSM startup or reset, but not in normal use. It indicates a hardserver bug or configuration problem.
PFNCompleted	The number of PauseForNotifications commands that have been completed by the HSM. Normally, this is one less than the PFNIssued figure because there is normally one such command outstanding.
ANIssued	The number of Asynchronous Notification messages issued by the HSM to the hardserver. These messages indicate such things as the clear key being pressed and the HSM being reset. In later firmware revisions inserting or removing the smartcard or changing the non-volatile memory also generate asynchronous notifications.
ChanJobsIssued	The number of fast channel jobs issued to the HSM. The fast channel facility is unsupported on current HSMs. This number should always be 0.
ChanJobsCompleted	The number of fast channel jobs completed by the HSM. The fast channel facility is unsupported on current HSMs. This number should always be 0.

ID	Value
CPULoadPercent	The current processing load on the HSM, represented as a number between 0 and 100. Because an HSM typically contains a number of different types of processing resources (for example, main CPU, and RSA acceleration), this figure is hard to interpret precisely. In general, HSMs report 100% CPU load when all RSA processing capacity is occupied; when performing non-RSA tasks the main CPU or another resource (such as the random number generator) can be saturated without this statistic reaching 100%.
HostIRQs	On PCI HSMs, the total number of interrupts received from the host. On current HSMs, approximately equal to the total of HostReadCount and HostWriteCount .
ChanJobErrors	The number of low-level (principally data transport) errors encountered while processing fast channel jobs. Should always be 0 on current HSMs.
HostDebugIRQs	On PCI HSMs, the number of debug interrupts received. This is used only for driver testing, and should be 0 in any production environment.
HostUnhandledIRQs	On PCI HSMs, the number of unidentified interrupts from the host. If this is nonzero, a driver or PCI bus problem is likely.
HostReadReconnect	On PCI HSMs, the number of deferred reads that have now completed. This should be the same as HostReadDeferred , or one less if a read is currently deferred.
ObjectsCreated	The number of times a new object has been put into the object store. This appears under the HSM's ModuleObjStats node.
ObjectsDestroyed	The number of items in the HSM's object store that have been deleted and their corresponding memory released.
ObjectCount	The current number of objects (keys, logical tokens, buffers, SEE Worlds) in the object store. This is equal to ObjectsCreated minus ObjectsDestroyed . An empty HSM contains a small number of objects that are always present.
CurrentTempC	The current temperature (in degrees Celsius) of the HSM main circuit board. First-generation HSMs do not have a temperature sensor and do not return temperature statistics.
MaxTempC	The maximum temperature recorded by the HSM's temperature sensor. This is stored in non-volatile memory, which is cleared only when the unit is initialized. First-generation HSMs do not have a temperature sensor and do not return temperature statistics.
MinTempC	The minimum temperature recorded by the HSM's temperature sensor. This is stored in non-volatile memory, which is cleared only when the unit is initialized. First-generation HSMs do not have a temperature sensor and do not return temperature statistics.
MemTotal	The total amount of RAM (both allocated and free) available to the HSM. This is the installed RAM size minus various fixed overheads.

ID	Value
<code>MemAllocKernel</code>	The total amount of RAM allocated for kernel (that is, non-SEE) use in an HSM. This is principally used for the object store (keys, logical tokens, and similar) and for big-number buffers.
<code>MemAllocUser</code>	The total amount of RAM allocated for user-mode processes in the HSM (0 for non-SEE use). This includes the size of the SEE Machine image, and the total heap space available to it.
<code>SystemFans</code>	The fan speed (RPM) for each fan in the HSM.

17.3. How data is affected when a module loses power and restarts

nShield modules use standard RAM to store many kinds of data, and data stored in such RAM is lost in the event that a module loses power (either intentionally, because you turned off power to it, or accidentally because of a power failure).

Therefore, after restoring power to a module, you must reload any keys that had been loaded onto it before it lost power. After reloading, the `KeyIDs` are different.

Likewise, after restoring power to a module, you must reload any cards that were loaded onto it before it lost power.

However, data stored in NVRAM is unaffected when a module loses power.



If you are using multiple nShield modules in the same Security World, and have the same key (or keys) loaded onto each module as part of a load-sharing configuration, loss of power to one module does not affect key availability (as long as at least one other module onto which the keys are loaded remains operational). However, in such a multiple-module system, after restoring power to a module, you must still reload any keys to that module before they can be available from that module.

18. nShield Connect and client configuration files

This appendix describes the configuration files that store the current configuration of an nShield Connect or client.

The *module configuration files* are stored on the internal file system of the nShield Connect. They are updated automatically when the nShield Connect is configured from the front panel. The configuration files are also exported automatically to the remote file system, where they can be edited manually and reloaded on the nShield Connect, if required. For more information, see [Client Software and module configuration](#).

The *client configuration files* are stored in the client's file system. The client's hardserver can also be set up using environment variables. Environment variable settings override settings in the client configuration files. For more information, see [Setting environment variables](#).

18.1. Location of client configuration files

The client configuration files are in the directory `opt/nfast/kmdata/config/` on the client computer's file system. This directory can contain the following files:

File	Description
<code>config</code>	The master configuration file. This contains the current configuration for the client. It is always present in the directory.
<code>config-default</code>	The default configuration file. This can be used to restore factory settings for the client. It is created by the <code>cfg-mkdefault</code> utility.

You can also save backup copies of configuration files in this directory.

18.2. Location of module configuration files

When you configure the nShield Connect from the front panel, the configuration files are exported automatically to the remote file system defined in the `rfs_client` section of the module configuration. The exported files are in the directory `/opt/nfast/kmdata/hsm-ESN/config`.

In this path, *ESN* is the electronic serial number of the network nShield Connect from which the files were exported. This directory contains the master configuration file `config`, which specifies the current configuration for the nShield Connect. It is always present in the direc

tory.

The remote file system information is also contained in the client configuration file section `remote_file_system`.

18.3. Structure of configuration files

The configuration files are text files. They must contain only characters with ASCII values between 32 and 127, and the tab, line break, and return characters.

Lines starting with a `#` character are comments and are ignored. Some comments that document the configuration options are generated by the configuration process. You can add your own comments, but in some cases they may later be overwritten.

A configuration file begins with a single line that specifies the version of the file syntax. This syntax-version line has the format:

```
syntax-version=n
```

In this syntax-version line example, *n* represents the version of the syntax in which the file is written. The system can process a file with a lower syntax version than the one it uses, but not one with a higher version.

After the syntax-version line, the rest of the configuration file consists of sections that can be edited to control different aspects of nShield Connect or client behavior. Each section begins with its name in square brackets, as in this example:

```
[slot_imports]
```

Module configuration files and client configuration files have some sections in common, and some specific to the type of file:

Both	Module file only	Client file only
<code>server_settings</code>	<code>nethsm_settings</code>	<code>module_settings</code>
<code>server_remotecomms</code>	<code>nethsm_eth</code>	<code>server_performance</code>
<code>server_remotecomms_ipv6</code>	<code>nethsm_eth_ipv6</code>	
<code>server_startup</code>	<code>nethsm_gateway</code> <code>nethsm_gateway_ipv6</code>	<code>nethsm_imports</code>
<code>load_seemachine</code>	<code>nethsm_bond</code>	<code>rfs_sync_client</code>

Both	Module file only	Client file only
slot_imports	nethsm_route nethsm_route_ipv6	remote_file_system
slot_exports	nethsm_eth1_enable	remote_administration_service_startuptartup
dynamic_slots	nethsm_bond_enable	
slot_mapping	nethsm_enable	
dynamic_slot_timeouts	cosmod	
auditlog_settings	hs_clients	
	rfs_client	
	sys_log	
	remote_sys_log	
	config_op	
	ui_lockout	

You can update the parameters defined in most of these sections to configure the way that the hardserver handles secure transactions between the nShield Connect and applications that run on the client.



Some sections are updated automatically and should not be edited manually. For more information, see the descriptions of individual sections.

In each section, the bracketed name is followed by a specified set of fields. Each field is on a separate line. Each field begins with its name, followed by an equals sign (=) and a value of the appropriate type. White space can be included at either end of the line (for example, in order to indent lines as an aid to clarity).

Some types of field are grouped into entries. An entry is a set of fields of different types that define an instance of an object (for example, a particular client as distinct from other clients). Entries in the same section are separated by a line that contains one or more hyphens (-). Blank lines and comments are allowed between the fields in an entry.

Strings are case sensitive in the section names and field names.



Multiple clients can be added to one configuration file by separating each client entry from the next with a line consisting of one or more hyphens.

If a particular section is not present in the configuration file, it is assumed to have no

entries.

18.4. Sections only in module configuration files

18.4.1. nethsm_settings

The `nethsm_settings` section defines settings specific to the nShield Connect. The section contains the following fields:

Field	Description
<code>enable_impauth_resilience</code>	When set to the default <code>yes</code> value, this field enables impauth resilience for the module. Setting this field to <code>no</code> disables impauth resilience.
<code>impauth_resilience_timeout</code>	<p>This field specifies the interval of time that must pass for a resumable impauth resilience session to expire. In the event of network errors, clients can reconnect and resume use of keys and other loaded objects until the specified interval has passed (after which all previously loaded objects must be reloaded).</p> <p>Specify this interval either in the form <code>N t</code>, where <code>N</code> is an integer and <code>t</code> is <code>s</code> for seconds, <code>m</code> for minutes, <code>h</code> for hours, <code>d</code> for days, or <code>w</code> for weeks, or as <code>never</code> (in which case sessions never expire). If you specify <code>N</code> but not <code>t</code>, the seconds are assumed. The default setting <code>604800</code> (604800 seconds is 1 week).</p>

18.4.2. nethsm_eth

The `nethsm_eth` section defines the Ethernet interfaces for IPv4 for the nShield Connect. Each interface is defined by an entry as follows:

Field	Description
<code>iface</code>	The identifier for the interface. This value must be <code>1</code> or <code>0</code> .
<code>addr</code>	The IP address of the nShield Connect. The default is <code>0.0.0.0</code> .
<code>netmask</code>	The net mask for the nShield Connect. The default is <code>0.0.0.0</code> .
<code>gateway</code>	This field is retained for backwards compatibility only. The IP address of the gateway is stored in the <code>nethsm_gateway</code> section and this field is set to <code>0.0.0.0</code> .
<code>linkspeed</code>	<p>This field specifies the link speed setting. It can be one of <code>auto/1Gb</code> (nShield Connect only), <code>10BaseT</code>, <code>10BaseT-FDX</code>, <code>100BaseTX</code>, or <code>100BaseTX-FDX</code>.</p> <p>We recommend that you accept the default <code>auto/1Gb</code> or <code>auto</code> setting. On the nShield Connect, this setting can auto-negotiate 1Gb Ethernet.</p>

18.4.3. nethsm_eth_ipv6

The **nethsm_eth_ipv6** section defines the Ethernet interfaces for IPv6 for the nShield Connect. Each interface is defined by an entry as follows:

Field	Description
iface	The identifier for the interface. This value must be 1 or 0 .
addr	The static IPv6 address for this interface. The default is :: . If SLAAC is enabled, this address is ignored.
netmask	The subnet prefix length of the static IPv6 address for the interface. The default is 64 .

18.4.4. nethsm_gateway

The **nethsm_gateway** section defines the default IPv4 Ethernet gateway for the nShield Connect. There is one field, as follows:

Field	Description
gateway	The IPv4 address of the gateway for the nShield Connect. The default is 0.0.0.0 .

18.4.5. nethsm_gateway_ipv6

The **nethsm_gateway_ipv6** section defines the default IPv6 Ethernet gateway for the nShield Connect. There is one field, as follows:

Field	Description
gateway	The IPv6 address of the gateway for the nShield Connect. The default is :: .
linklocal_if	The ethernet interface (0 or 1) to use if the IPv6 default gateway address is a link-local address. The information is not used if the IPv6 default gateway is not a link-local address (default 0).

18.4.6. nethsm_bond

The **nethsm_bond** section defines the HSM bond interface, used for network bonding. The bond interface's address and netmask configuration are inherited from the **eth0** (**iface=0**) configuration. Each entry has the following fields:

Field	Description
<code>mode</code>	<p>Possible values:</p> <ul style="list-style-type: none"> • <code>802.3ad</code> • <code>active-backup</code> Default: <code>802.3ad</code>.
<code>miimon</code>	<p>The MII link monitoring frequency in milliseconds.</p> <p>Range: <code>0 - 10000</code>.</p> <p>Default: <code>100</code>.</p>
<code>primary</code>	<p>Primary device. The specified device will always be the active slave while it is available.</p> <p>Possible values:</p> <ul style="list-style-type: none"> • <code>eth0</code> • <code>eth1</code> <p>Default: <code>eth0</code>.</p> <p>Only valid for <code>active-backup</code> mode.</p>
<code>resend_igmp</code>	<p>The number of IGMP membership reports to be issued after a failover event.</p> <p>Range: <code>0 - 255</code>.</p> <p>Default: <code>1</code>.</p> <p>A value of <code>0</code> prevents the IGMP membership report from being issued in response to the failover event.</p> <p>Only valid for <code>active-backup</code> mode.</p>
<code>lacp_rate</code>	<p>The rate in which the Ethernet interface asks the link partner to transmit LACPDU packets in <code>802.3ad</code> mode.</p> <p>Possible values:</p> <ul style="list-style-type: none"> • <code>slow</code> • <code>fast</code> <p>Default: <code>slow</code>.</p> <p>Only valid for <code>802.3ad</code> mode.</p>

Field	Description
<code>xmit_hash_policy</code>	<p>The transmit hash policy to use for slave selection in <code>802.3ad</code> mode.</p> <p>Possible values:</p> <ul style="list-style-type: none"> • <code>layer2</code> • <code>layer2+3</code> • <code>encap2+3</code> <p>Default: <code>layer2</code>.</p>

The process of network bonding, including all the fields above, are described in <https://www.kernel.org/doc/Documentation/networking/bonding.txt>.

18.4.7. nethsm_route

The `nethsm_route` section defines the static IPv4 routes for the nShield Connect. Each route is defined by an entry as follows:

Field	Description
<code>addr</code>	The IPv4 address of the route destination. The default is <code>0.0.0.0</code> .
<code>masklen</code>	The length to mask for the route.
<code>gateway</code>	The IPv4 address of the gateway for the route. The default is <code>0.0.0.0</code> .

18.4.8. nethsm_route_ipv6

The `nethsm_route_ipv6` section defines the static IPv6 routes for the nShield Connect. Each route is defined by an entry as follows:

Field	Description
<code>addr</code>	Routable IPv6 address block. The default is <code>::</code> .
<code>masklen</code>	The number of bits to mask for the subnet address prefix, that is, the number in the range of <code>1</code> to <code>128</code> after the <code>/</code> of an address in CIDR format. The default is <code>64</code> .
<code>gateway</code>	Route gateway. The default is <code>::</code> .
<code>linklocal_if</code>	The ethernet interface (<code>0</code> or <code>1</code>) to use if the IPv6 default gateway address is a link-local address. The information is not used if the IPv6 default gateway is not a link-local address (default <code>0</code>).

18.4.9. nethsm_eth1_enable

The `nethsm_eth1_enable` section defines if the eth1 interface is enabled.

Field	Description
<code>enable</code>	The indicator of whether the eth1 interface is enabled or disabled (default 'no').

18.4.10. nethsm_bond_enable

The `nethsm_bond_enable` section defines if the bond interface is enabled.

Field	Description
<code>enable</code>	The indicator of whether the bond interface is enabled or disabled (default 'no').

18.4.11. nethsm_enable

The `nethsm_enable` section defines whether IPv4 and or IPv6 are enabled or disabled for the interfaces of the unit. The enable fields are:

Field	Description
<code>iface</code>	The ethernet interface (<code>0</code> or <code>1</code>) to which the following fields apply.
<code>enable_ipv4</code>	Indicator of whether the IPv4 protocol on the interface is enabled (default: <code>yes</code>).
<code>enable_ipv6</code>	Indicator of whether the IPv6 protocol on the interface is enabled (default: <code>no</code>).
<code>ipv6_conf_addr</code>	Indicator of whether the interface uses IPv6 static addresses (<code>static</code>) or SLAAC (<code>slaac</code>). The default is <code>static</code> .

18.4.12. cosmod

The `cosmod` section defines the input configuration for the nShield Connect. The configuration is defined by an entry as follows:

Field	Description
<code>keymap</code>	The selected layout for the keyboard connected to the nShield Connect front panel. This value is either <code>UK</code> or <code>US</code> .

18.4.13. hs_clients

The **hs_clients** section defines the clients that are allowed to connect to the nShield Connect. Each client is defined by an entry as follows:

Field	Description
addr	<p>Either the IP address of the client or 0.0.0.0, ::, or blank if the Connect is to accept clients identified by their keyhash instead of their IP address.</p> <p>If no value is set (the field is blank), or if it is set to 0.0.0.0 or ::, only HKNETI identification is allowed.</p> <p>The default is blank.</p> <p>0.0.0.0 or ::, and blank result in the same behavior. You cannot enter these values in the front-panel user interface. You can only use them in the configuration file and you will have to use the correct key hash for identification if no IP address is configured.</p>
clientperm	<p>The type of connection permitted from the client.</p> <p>This is one of the following:</p> <ul style="list-style-type: none"> • unpriv (non-privileged connections) • priv (privileged connections) • priv_lowport (privileged connections on ports lower than 1024) <p>The default is unpriv.</p>
keyhash	<p>The hash of the KNETI key that the client should present to authenticate itself.</p> <p>Both software based authentication and nToken authentication are supported.</p> <p>For nToken authentication, a value must also be provided for the esn field.</p> <p>The default is 40 zeros, which means that no key authentication is required.</p>
esn	<p>The ESN of the client's nToken. (Only applicable if nToken authentication is used.)</p>
timelimit	<p>Obsolete. This should be set to 0 or unset in new configuration entries. This will be removed in a future release.</p>
datalimit	<p>Obsolete. This should be set to 0 or unset in new configuration entries. This will be removed in a future release.</p>

18.4.14. rfs_client

The **rfs_client** section defines the remote file system where the module configuration is backed up and the master copy of the Security World data is located, as follows:

Field	Description
<code>remote_ip</code>	The IP address of the RFS.
<code>remote_port</code>	The port number on which the RFS hardserver is listening.

18.4.15. sys_log

The `sys_log` section defines how the nShield Connect logging works:

Field	Description
<code>behaviour</code>	<p>The way the log is written. How the log is written is decided by setting either of the following:</p> <ul style="list-style-type: none"> • <code>log</code> • <code>push</code> <p>If <code>log</code> is set, the log is written only to the file system of the nShield Connect. It is lost if the nShield Connect is rebooted. Logging stops when the file system is full. If <code>push</code> is set, the log is automatically appended to the log file in the RFS or remote syslog server at the interval specified in <code>push_interval</code>.</p>
<code>push_interval</code>	The number of minutes between the log being appended when <code>behaviour</code> is set to <code>push</code> . The default is <code>60</code> . The minimum is 1.

18.4.16. remote_sys_log

The `remote_sys_log` section defines a remote syslog server to send the nShield Connect logging (system.log and hardserver.log) to.

Field	Description
<code>remote_syslog_ip</code>	The IP address of the remote syslog server to send logs to. The default is <code>0.0.0.0</code> .
<code>remote_sys_log_port</code>	The UDP port of the remote syslog server to send logs to. The default is <code>514</code> .

18.4.17. config_op

The `config_op` section defines whether a client computer is allowed to update the module configuration automatically (to push a configuration) from files on the client:

Field	Description
<code>push</code>	Whether the client is allowed to push configuration files to the nShield Connect. This is decided by setting one of the following: <ul style="list-style-type: none"> • <code>ON</code> • <code>OFF</code> If <code>on</code> , the client specified in the <code>remote_ip</code> section is allowed to update the configuration of the nShield Connect remotely.
<code>remote_ip</code>	The IP address of the client that is allowed to push config files. If no value is set, or if it is set to <code>0.0.0.0</code> or <code>::</code> , any IP address can push on a new config file.
<code>remote_keyhash</code>	The hash of the key with which the authorized client is to authenticate itself, or (as the default) 40 zeros to indicate no key authentication required.



The default value for `remote_keyhash` (40 zeros) specifies that no authentication should occur. We recommend specifying a key hash in place of this default.

18.4.18. ui_lockout

The `ui_lockout` section defines whether you can lock the nShield Connect using login settings.

To be compatible with UI lockout, the OCS card(s):

- Must be persistent.
- Must not be remoteable.
- Do not need a passphrase, but if a passphrase is configured, it has to be used.

Field	Description
<code>lockout_mode</code>	Controls front panel access to the nShield Connect. Set to: <ul style="list-style-type: none"> • <code>locked</code> Enables UI lockout without requiring a logical token. • <code>locked_lt</code> Enables UI lockout with a logical token (OCS) (requires a valid <code>ltui_hash</code> to be set). • <code>unlocked</code> No UI lockout (default).

Field	Description
<code>ltui_hash</code>	The hash of the logical token (LTUI) required to authorize access to the nShield Connect menu structure when <code>lockout_mode</code> is set to <code>locked_lt</code> .
<code>panel_poweroff</code>	This controls the function of the Power button on the nShield Connect front panel when it is in operational mode. The default setting of <code>yes</code> enables the Power button. When set to <code>no</code> , the Power button is disabled.

18.5. Sections in both module and client configuration files

18.5.1. server_settings


The `server_settings` section defines the settings for the client hardserver you can modify while the hardserver is running.



These flags are used by the `NFLOG_DETAIL` environment variable (see [Environment variables to control logging](#)).

The section contains the following fields:

Field	Description
<code>loglevel</code>	<p>This field specifies the level of logging performed by the hardserver. It takes a value that is one of the following:</p> <ul style="list-style-type: none"> <code>info</code> <code>notice</code> <code>client</code> <code>remoteserver</code> <code>error</code> <code>serious</code> <code>internal</code> <code>startup</code> <code>fatal</code> <code>fatalinternal</code> <p>The default is <code>info</code>. For more information, see Logging, debugging, and diagnostics. If the <code>NFAST_SERVERLOGLEVEL</code> environment variable is set, it overrides any <code>loglevel</code> value set in the configuration file.</p> <div> <p><code>NFAST_SERVERLOGLEVEL</code> is a legacy debug variable.</p> </div>

Field	Description
<code>logdetail</code>	This field specifies the level of detail logged by the hardserver. You can supply one or more flags in a space-separated list. For more information about the flags, see the table below.
<code>connect_retry</code>	This field specifies the number of seconds to wait before retrying a remote connection to a client hardserver. The default is 10.
<code>connect_maxqueue</code>	This field specifies the maximum number of jobs which can be queued on the hardserver. The default is 4096: this is also the maximum value. Setting <code>connect_maxqueue</code> to a high value allows high throughput, but may cause long latency if the hardserver goes down.
<code>connect_broken</code>	This field specifies the number of seconds of inactivity allowed before a connection to a client hardserver is declared broken. The default is 90.
<code>connect_keepalive</code>	This field specifies the number of seconds between <code>keepalive</code> packets for remote connections to a client hardserver. The default is 10.
<code>accept_keepidle</code>	This field specifies the number of seconds before the first <code>keepalive</code> packet for remote incoming connections. The default is 30. Ideally, <code>accept_keepalive</code> should be at least twice the value of the <code>connect_keepalive</code> setting on the unattended machines.
<code>accept_keepalive</code>	<p>This field specifies the number of seconds between <code>keepalive</code> packets for remote incoming connections. The socket will be closed after up to ten consecutive probe failures. The default is 10.</p> <p>Ideally, <code>accept_keepalive</code> should be a value such that $(10 * \text{accept_keepalive}) > \text{connect_broken}$ on the unattended machine. Using the default values for both these fields will fulfil this requirement.</p>
<code>connect_command_block</code>	When the nShield Connect has failed, this field specifies the number of seconds the hardserver should wait before failing commands directed to that HSM with a <code>NetworkError</code> message. For commands to have a chance of succeeding after the nShield Connect has failed this value should be greater than that of <code>connect_retry</code> . If it is set to 0, commands to an nShield Connect are failed with <code>NetworkError</code> immediately. The default is 35.
<code>max_pci_if_vers</code>	This field specifies the maximum PCI interface version number. If <code>max_pci_if_vers</code> is set to 0 (the default), there is no limit.
<code>enable_remote_mode</code>	<p>If this field is set to <code>yes</code> (the default value) in the module configuration file, nShield Connect mode changing using the <code>nopclearfail</code> utility is enabled. If set to <code>no</code>, mode changing using the <code>nopclearfail</code> utility is disabled.</p> <div>  <div>Do not set <code>enable_remote_mode</code> in the client configuration file.</div> </div>

Field	Description
<code>enable_remote_reboot</code>	If this field is set to yes (the default value) in the module configuration file, the nShield Connect remote reboot using the nethsmadmin utility is enabled. If set to no , remote reboot using the nethsmadmin utility is disabled. Run cfg-push-nethsm to push the new config file to the module.
<code>enable_remote_upgrade</code>	If this field is set to yes (the default value) in the module configuration file, the nShield Connect remote upgrade using the nethsmadmin utility is enabled. If set to no , remote upgrade using the nethsmadmin utility is disabled. Run cfg-pushnethsm to push the new config file to the module.

These flags are those used by the **NFLOG_DETAIL** environment variable (see [Environment variables to control logging](#)).

You can supply a number of flags with the **logdetail** field, which specifies the level of detail logged by the hardserver (see the table above). Supply the flags in a space separated list:

Flag	Description
<code>external_time</code>	This flag specifies the external time (that is, the time according to your machine's local clock) with the log entry.
<code>external_date</code>	This flag specifies the external date (that is, the date according to your machine's local clock) with the log entry.
<code>external_tid</code>	This flag specifies the external thread ID with the log entry.
<code>external_time_t</code>	This flag specifies the external time_t (that is, the time in machine clock ticks rather than local time) with the log entry.
<code>stack_backtrace</code>	This flag specifies the stack backtrace with the log entry.
<code>stack_file</code>	This flag specifies the stack file with the log entry.
<code>stack_line</code>	This flag specifies the message line number in the original library. This flag is likely to be most useful in conjunction with example code we have supplied that has been written to take advantage of this flag.
<code>msg_severity</code>	This flag specifies the message severity (a severity level as used by the NFLOG_SEVERITY environment variable) with the log entry.
<code>msg_categories</code>	This flag specifies the message category (a category as used by the NFLOG_CATEGORIES environment variable) with the log entry.
<code>msg_writeable</code>	This flag specifies message writeables, and extra information that can be written to the log entry, if any such exist.
<code>msg_file</code>	This flag specifies the message file in the original library. This flag is likely to be most useful in conjunction with example code we have supplied that has been written to take advantage of this flag.

Flag	Description
<code>msg_line</code>	This flag specifies the message line number in the original library. This flag is likely to be most useful in conjunction with example code we have supplied that has been written to take advantage of this flag.
<code>options_utc</code>	This flag showing the date and time in UTC (Coordinated Universal Time) instead of local time.
<code>unix_file_descriptor_max</code>	This field sets the number of file descriptors the hardserver must be capable of having open concurrently on Linux. The value must be an integer. If <code>unix_file_descriptor_max</code> is set to 0 (the default), the value will be ignored by the hardserver. If it is set to a positive value, the hardserver will refuse to start if the file descriptor hard limit on the system is less than that value. This configuration entry can be used to make sure that the hardserver is capable of satisfying the maximum number of hardserver connections that applications may make use of.

18.5.2. server_remotecomms

The `server_remotecomms` section defines the remote IPv4 communication settings for the client hardserver. These are read only at hardserver start-up.

Any changes made to these fields in the Connect module config file should be followed by a reboot of the nShield Connect.

It is not recommended that the port number be changed. If it is changed, the port number must match in the configuration of peers. For example, if the port number is changed in the nShield Connect configuration, the `remote_port` field of the `nethsm_imports` section of the client-side hardserver config file must be updated to match. The port number can also be specified with the `-P` parameter when running `nethsmenroll`.

This section contains the following fields:

Field	Description
<code>impath_port</code>	<p>This field specifies the port on which the hardserver listens for incoming <code>impath</code> connections. The default is 9004. Setting this field to 0 specifies that the hardserver does not listen for incoming connections (do not set to 0 on an nShield Connect).</p> <p>Make sure that firewall settings are consistent with port settings.</p> <p>If you change this field you will need to re-enroll your client with the new port value, see Enrolling the client from the command line.</p>

Field	Description
<code>impath_addr</code>	This field specifies the IPv4 address at which the hardserver listens for incoming impath connections. If this field is set to <code>inaddr_any</code> (which is the default), the hardserver listens on all IP addresses.
<code>impath_interface</code>	<p>This field specifies a string representing the name of the Ethernet interface on which the hardserver listens. This field is only examined if <code>impath_addr</code> is set to <code>inaddr_any</code>.</p> <p>By default, the <code>impath_interface</code> field is empty, which means that the hardserver listens on all interfaces. If the string is not recognized as the name of one of the interfaces of the nShield Connect, the hardserver does not listen.</p>

18.5.3. server_remotecomms_ipv6

The `server_remotecomms_ipv6` section defines the remote IPv6 communication settings for the client hardserver. These are read only at hardserver start-up.

Any changes made to these fields in the Connect module config file should be followed by a reboot of the nShield Connect.

It is not recommended that the port number be changed. If it is changed, the port number must match in the configuration of peers. For example, if the port number is changed in the nShield Connect configuration, the `remote_port` field of the `nethsm_imports` section of the client-side hardserver config file must be updated to match. The port number can also be specified with the `-P` parameter when running `nethsmenroll`.

This section contains the following fields:

Field	Description
<code>impath_port</code>	<p>This field specifies the port on which the hardserver listens for incoming impath connections. The default is 9004. Setting this field to 0 specifies that the hardserver does not listen for incoming connections (do not set to 0 on an nShield Connect).</p> <p>Make sure that firewall settings are consistent with port settings.</p> <p>If you change this field you will need to re-enroll your client with the new port value, see Enrolling the client from the command line.</p>
<code>impath_addr</code>	This field specifies the IPv6 address at which the hardserver listens for incoming impath connections. If this field is set to <code>::</code> (which is the default), the hardserver listens on all IP addresses.

Field	Description
<code>impath_interface</code>	<p>This field specifies a string representing the name of the Ethernet interface on which the hardserver listens. This field is only examined if <code>impath_addr</code> is set to <code>::</code>.</p> <p>By default, the <code>impath_interface</code> field is empty, which means that the hardserver listens on all interfaces. If the string is not recognized as the name of one of the interfaces of the nShield Connect, the hardserver does not listen.</p> <p>Setting this field to <code>0</code> will disable IPv6 in the hardserver.</p>

18.5.4. server_startup

The `server_startup` section defines the settings for the hardserver that are loaded at start-up. Any changes you make to the settings in this section do not take effect until after you restart the hardserver. For more information, see [Stopping and restarting the client hardserver](#).

The section contains the following fields:


Field	Description
<code>unix_socket_name</code>	This field specifies the name of the socket to use for non-privileged connections on Linux. The default is <code>/dev/nfast/nserver</code> . If the <code>NFAST_SERVER</code> environment variable is set, it overrides any value set for <code>unix_socket_name</code> in the hardserver configuration file. For more information about environment variables, see Environment variables .
<code>unix_privsocket_name</code>	This field specifies the name of the socket to use for privileged connections on Linux. The default is <code>/dev/nfast/privnserver</code> . If the <code>NFAST_PRIVSERVER</code> environment variable is set, it overrides any value set for <code>unix_privsocket_name</code> in the hardserver configuration file.
<code>nt_pipe_name</code>	This field is not used on Linux.
<code>nt_pipe_users</code>	This field is not used on Linux.
<code>nt_privpipe_name</code>	This field is not used on Linux.
<code>nt_privpipe_users</code>	This field is not used on Linux.
<code>nonpriv_port</code>	<p>This field specifies the port on which the hardserver listens for local non-privileged TCP connections. The value <code>0</code> (which is the default) specifies</p> <p>Make sure that your network firewall settings are correct. See the Installation Guide for more about firewall settings.</p> <p>If the <code>NFAST_SERVER_PORT</code> environment variable is set, it overrides any value set for <code>nonpriv_port</code> in the hardserver configuration file.</p>

Field	Description
<code>priv_port</code>	<p>This field specifies the port on which the hardserver listens for local privileged TCP connections. The value <code>0</code> (which is the default) specifies none. Java clients default to connecting to port 9001.</p> <p>If the <code>NFAST_SERVER_PRIVPORT</code> environment variable is set, it overrides any value set for <code>priv_port</code> in the hardserver configuration file.</p>

18.5.5. load_seemachine

The `load_seemachine` section of the hardserver configuration file defines SEE machines that the nShield Connects connected to the client should load and, if required, start for use by other clients. Each SEE machine is defined by the following fields:

Field	Description
<code>module</code>	This field specifies the nShield Connect on to which to load the SEE machine. The value must be an integer. A nShield Connect with this ID must be configured on the client computer.
<code>machine_file</code>	This field specifies the file name of the SEE machine.
<code>userdata</code>	This field specifies the <code>userdata</code> file name to pass to the SEE machine on start-up. If this field is blank (" "), the SEE machine is loaded but not started. By default, this field is blank.
<code>worldid_pubname</code>	This field specifies the <code>PublishedObject</code> name to use for publishing the <code>KeyID</code> of the started SEE machine. If this field is blank (" "), the <code>KeyID</code> is not published. This field is ignored if the value of the <code>userdata</code> field is blank.
<code>postload_prog</code>	<p>This field specifies the program to run after loading the SEE machine in order to perform any initialization required by the SEE machine or its clients. The specified program must accept an argument of the form <code>-m module#</code>.</p> <p>To run <code>see-sock-serv</code> directly on the nShield Connect, set this field to <code>sock-serv</code>.</p>
<code>postload_args</code>	<p>This field specifies arguments to pass to the program specified by the <code>postload_prog</code> field. The argument <code>-m module#</code> is automatically passed as the first argument. The <code>postload_args</code> field is ignored if <code>postload_prog</code> is not specified or is blank.</p> <p>To run <code>see-sock-serv</code> directly on the nShield Connect, set this field to <code>-p pub-name</code>.</p>

Field	Description
<code>pull_rfs</code>	<p>This field specifies whether the SEE machine name and userdata should be pulled from the RFS. The default is <code>no</code>; set to <code>yes</code> to pull the SEE machine and userdata from the RFS before loading on the remote nShield Connect.</p> <p>This field will be ignored if set on client machine configurations.</p> <div>  <p>This field will not be added to existing configuration files if you are upgrading an image. If you require the new functionality enabled by this field, you can add the field to the <code>load_seemachine</code> section of your existing configuration file.</p> </div>

18.5.6. slot_imports

The `slot_imports` section defines slots from remote nShield Connects that will be available to the client. Each slot is defined by the following fields:

Field	Description
<code>local_esn</code>	This field specifies the ESN of the local nShield Connect importing the slot.
<code>local_slotid</code>	This field specifies the <code>SlotID</code> to use to refer to the slot when it is imported on the local nShield Connect. The default is 2.
<code>remote_ip</code>	This field specifies the IP address of the machine that hosts the slot to import.
<code>remote_port</code>	This field specifies the port for connecting to the nShield Connect.
<code>remote_esn</code>	This field specifies the ESN of the remote nShield Connect from which to import the slot.
<code>remote_slotid</code>	This field specifies the <code>SlotID</code> of the slot to import on the remote nShield Connect. The value of this field must be an integer. The default is 0.

18.5.7. slot_exports

The `slot_exports` section defines the slots on the HSMs connected directly to the client that the client hardserver should export for other network HSMs to import. Each local slot has an entry for each network nShield Connect that can import it, consisting of the following fields:

Field	Description
<code>local_esn</code>	This field specifies the ESN of the local nShield Connect whose slot can be imported by a network nShield Connect.

Field	Description
<code>local_slotid</code>	This field specifies the <code>SlotID</code> of the slot that is allowed to be exported. The value must be an integer. The default is 0.
<code>remote_ip</code>	This field specifies the IP address of the nShield Connect that is allowed to import the slot. Keep this field blank to allow all machines. The default is blank.
<code>remote_esn</code>	This field specifies the ESN of the nShield Connect allowed to import the slot. Leave the value blank to allow all permitted nShield Connects in the Security World. The default is blank.

18.5.8. dynamic_slots

The `dynamic_slots` section defines the number of Dynamic Slots that each HSM supports.

Field	Description
<code>esn</code>	ESN of the HSM to be configured with Dynamic Slots.
<code>slotcount</code>	The number of Dynamic Slots that the HSM is to support. If set to 0 (default) the HSM does not support Remote Administration.

18.5.9. slot_mapping

The `slot_mapping` section defines, for each specified HSM, a slot that is exchanged with slot 0 of the HSM. Slot 0 becomes a Dynamic Slot and the local slot becomes the specified slot number. This enables applications and utilities that only support slot 0 to use Remote Administration.

Field	Description
<code>esn</code>	ESN of the HSM to which the mapping is applied.
<code>slot</code>	<p>The slot number to be swapped with slot 0, so that:</p> <ul style="list-style-type: none"> Slot 0 refers to a Dynamic Slot The specified slot number refers to the local slot of the HSM. <p>If <code>slot</code> is set to 0 (default) there is no slot mapping.</p>

18.5.10. dynamic_slot_timeouts

The `dynamic_slot_timeouts` section defines timeout values that are used to specify expected smartcard responsiveness for all HSMs associated with the relevant host or client, when using the Remote Administration.

Field	Description
<code>round_trip_time_limit</code>	<p><code>round_trip_time_limit</code> > 5s + network latency time</p> <p>Round trip (HSM to smartcard and back) time limit in seconds. The card is regarded as removed, if no response has been received within the allowed time. Expected network delays need to be taken into account when setting this. The default is ten seconds.</p>
<code>card_remove_detect_time_limit</code>	<p><code>card_remove_detect_time_limit</code> >= <code>round_trip_time_limit</code> *2</p> <p>Maximum number of seconds that can pass without a response from the smart card, before it is regarded as removed and all the keys that it protects are unloaded. Lower values increase network traffic. The default is 30 seconds.</p>

18.6. Sections only in client configuration files

18.6.1. module_settings

The `module_settings` section defines the settings for the nShield Connect that can be changed while the hardserver is running. The section contains the following fields:

Field	Description
<code>esn</code>	This field specifies the electronic serial number of the nShield Connect.
<code>priority</code>	This field specifies the priority of the nShield Connect. The value for this field can be an integer from 1 (highest) to 100 (lowest). The default is 100.

18.6.2. server_performance

The `server_performance` section defines the performance settings for the client hardserver. These are read only at hardserver start-up. This section contains the following fields:

Field	Description
<code>enable_scaling</code>	This field determines whether multi-threaded performance scaling is enabled or not. If this field is set to auto (or not set), the hardserver automatically chooses the best option for the available hardware (enabled when using nShield Connects, for which scaling is currently optimized, and disabled if using nShield Solos or any other local devices). It can explicitly be enabled by setting to yes , and explicitly disabled by setting to no .

Field	Description
<code>target_concurrency</code>	This field allows the level of concurrency to be tuned. The value must be an integer and will only come into effect when multi-threaded performance scaling is enabled. If <code>target_concurrency</code> is set to <code>0</code> (the default), the value will be automatically configured by the hardserver based on the available number of physical CPU cores. The target concurrency configured is written to the hardserver log.

18.6.3. nethsm_imports

The `nethsm_imports` section defines the network nShield Connects that the client imports. It can also be set up by the `nethsmenroll` utility. Each nShield Connect is defined by the following fields:

Field	Description
<code>local_module</code>	This field specifies the ModuleID to assign to the imported nShield Connect. The value must be an integer. An nShield Connect with this ID must not be already configured on the client computer.
<code>remote_ip</code>	This field specifies the IP address of the nShield Connect to import.
<code>remote_port</code>	This field specifies the port for connecting to the nShield Connect.
<code>remote_esn</code>	This field specifies the ESN of the imported nShield Connect.
<code>keyhash</code>	This field specifies the hash of the key that the nShield Connect should use to authenticate itself.
<code>timelimit</code>	Obsolete. This should be set to <code>0</code> or unset in new configuration entries. This will be removed in a future release.
<code>datalimit</code>	Obsolete. This should be set to <code>0</code> or unset in new configuration entries. This will be removed in a future release.
<code>privileged</code>	The value in this field specifies whether the client can make a privileged connection to the nShield Connect. The default is <code>0</code> , which specifies no privileged connections. Any other value specifies privileged connections.
<code>ntoken_esn</code>	This field specifies the ESN of this client's nToken, if an nToken is installed.



The default value for `remote_keyhash` (40 zeros) specifies that no authentication should occur. We recommend that you set a specific key hash in place of this default.

18.6.4. rfs_sync_client

This section defines which remote file system the client should use to synchronize its key management data:

Field	Description
<code>remote_ip</code>	The IP address of the RFS against which to synchronize.
<code>remote_port</code>	This field specifies the port for connecting to the RFS.
<code>use_kneti</code>	Setting this option to <code>yes</code> to use a local module KNETI instead of the default hardserver's software KNETI to authenticate this client to the RFS.
<code>local_esn</code>	This is only required if <code>use_kneti</code> is set to <code>yes</code> . It is the ESN of the local module used for authentication.

18.6.5. remote_file_system



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

The `remote_file_system` section defines a remote file system on the client by listing the nShield Connects allowed to access the file system on this client. Each nShield Connect is defined by an entry consisting of the following fields:

Field	Description
<code>remote_ip</code>	This field specifies the IP address of the remote nShield Connect that is allowed to access the file system on this client.
<code>remote_esn</code>	This field specifies the ESN of the remote nShield Connect allowed to access the file system on this client.
<code>keyhash</code>	This field specifies the hash of the key with which the client must authenticate itself to the nShield Connect. The default is 40 zeros, which means that no key authentication is required.
<code>native_path</code>	This field specifies the local file name for the volume to which this entry corresponds.
<code>volume</code>	This field specifies the volume that the remote host would access to use this entry.
<code>allow_read</code>	If this field is set to <code>yes</code> , it means that a remote server is allowed to read the contents of the file. The default is <code>no</code> .
<code>allow_write</code>	If this field is set to <code>yes</code> , it means that a remote server is allowed to write to the file. The default is <code>no</code> .
<code>allow_list</code>	If this field is set to <code>yes</code> , it means that a remote server is allowed to list the contents of the file. The default is <code>no</code> .

Field	Description
<code>is_directory</code>	If this field is set to <code>yes</code> , it means that this entry represents a directory. The default is <code>no</code> .
<code>is_text</code>	If this field is set to <code>yes</code> , it means that line endings should be converted to and from the Linux convention for transfers.



If you upgrade from an earlier software version to v12 and are using Remote Administration, you need to manually add the following sections to your configuration file.

18.6.6. remote_administration_slot_server_startup

The `remote_administration_slot_server_startup` section defines the communication settings that are applied at start-up to the Remote Administration.

Field	Description
<code>port</code>	Which port to use to connect to the Remote Administration. The default is 9005.

19. Cryptographic algorithms

19.1. Symmetric algorithms

Symmetric Algorithms				
Algorithm	FIPS approved in a v1 or v2 Security World	FIPS approved in a v3 Security World	Key type	Supported by generatekey
AES	Y	Y	AES or Rijndael	Y
Arcfour	N	N	Arcfour	N
ARIA	N	N	Aria	N
Camellia	N	N	Camellia	N
CAST 256	N	N	CAST256	N
DES	N	N	DES	N
DES2	Y	N	DES2	Y
Triple DES	Y	N ¹	Triple DES	Y
MD5 HMAC	N	N	HMACMD5	N
RIPEMD160 HMAC	N	N	HMACRIPEMD160	N
SEED	N	N	SEED	N
SHA-1 HMAC	Y	Y	HMACSHA1	Y
SHA-224 HMAC	Y	Y	HMACSHA224	N
SHA-256 HMAC	Y	Y	HMACSHA256	Y
SHA-384 HMAC	Y	Y	HMACSHA384	Y
SHA-512 HMAC	Y	Y	HMACSHA512	Y
Tiger HMAC	N	N	HMACTiger	N

¹ Not FIPS 140-2 approved for encryption operations, but available for decryption operations.

19.2. Asymmetric algorithms

Asymmetric Algorithms

Algorithm	FIPS approved in a v1 or v2 Security World	FIPS approved in a v3 Security World ¹	Key type	Supported by generatekey
Diffie-Hellman	Y	Y	DH or DHEx	Y
DSA	Y	Y	DSA	Y
ECDH	Y ²	Y ²	ECDH or EC ³	Y
ECDSA	Y ⁴	Y ⁴	ECDSA or EC	Y
ECIES	N	N	ECDH or EC	N
Ed25519	N	N	Ed25519	N
El Gamal	Y	Y	DH	Y
KCDSA	N	N	KCDSA	N
RSA	Y	Y	RSA	Y
X25519	N	N	X25519	N

¹ Some insecure key sizes are non-FIPS 140-2 approved.

² FIPS 140-2 approval is only for use with ECDH keys, not with EC keys, but see ³ for exception.

³ FIPS 140-2 allows an EC key to be used as an ECDH key for a single use-case. In this use case, an ECDH key is allowed to perform a single signing of a Certificate Signing Request (CSR), so that a certificate for the ECDH key can be generated.

⁴ FIPS 140-2 approval is only for use with ECDSA keys, not with EC keys.

19.3. FIPS information

The latest guidance from the National Institute of Standards and Technology (NIST) is that

- A module is only operating in a FIPS approved mode when it uses FIPS 140-2 approved algorithms, and the algorithms are used with keys of an FIPS approved key length or elliptic curve.

When a module is initialized into FIPS 140-2 Level 3 mode, you are only offered FIPS-approved algorithms. If you have a Security World created to comply with FIPS 140-2 Level 3 and have any protocols that use algorithms not approved by FIPS, you must either migrate to a FIPS 140-2 Level 2 Security World or change your protocols. If you have a Security World created to comply with FIPS 140-2 Level 3 and have existing long-term keys for

unapproved algorithms, then these keys cannot be used with the current firmware. In such a case, we recommend that you either migrate your Security World to a FIPS 140-2 Level 2 Security World or replace these keys with approved keys before upgrading to the current firmware.

These changes do not affect Security Worlds that were created to comply with FIPS 140-2 Level 2, nor do they affect systems that use the nShield module to protect long-term keys but perform encryption with session keys on the host (as is the case with the nShield MSCAPI, and PKCS #11 libraries).



Some algorithms that are shown are not FIPS-approved for encryption or signing operations but may still be available for decryption or verification operations.

To obtain more details on the specific algorithms that are FIPS approved for use in the HSM, refer to the nShield Security Policy for the particular FIPS CMVP certified nShield product that you are using. The FIPS CMVP certificates for nShield products can be found at <https://csrc.nist.gov/projects/cryptographic-module-validation-program/validated-modules/search>, and the Security Policy is linked from the FIPS CMVP certificate.

19.4. Compatibility with v1 and v2 Security Worlds

If the firmware on an HSM is upgraded to v12.50 or later, a v1 or v2 Security World that was FIPS 140-2 Level 3 compliant will no longer be FIPS compliant.

It is possible to create a FIPS-conforming Security World from a host server that is running Security World v12.50 or later as long as the HSM is running v12.50 or later firmware. However your solution won't be FIPS certifiable unless you are running the exact version of firmware that has been FIPS 140-2 certified.

These changes do not affect Security Worlds that were created to comply with FIPS 140-2 Level 2, nor do they affect systems that use the nShield module to protect long-term keys but perform encryption with session keys on the host (as is the case with the nShield MSCAPI, and PKCS #11 libraries).

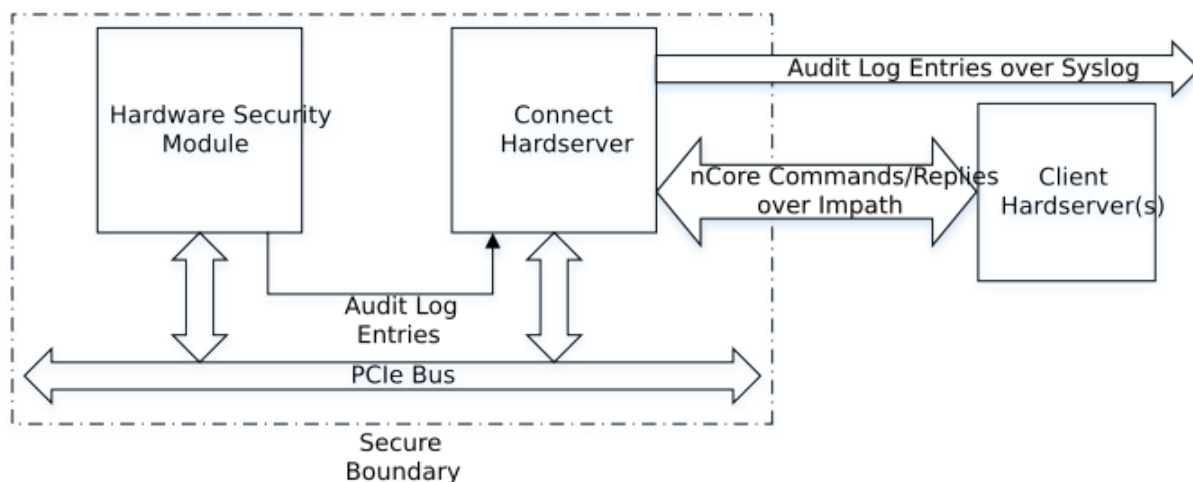
20. Audit Logging

This Appendix describes the Audit Logging capability available for the nShield product line. This capability provides the following features:

- Logs generated and signed on the nShield HSM
- Tamper detection
- Deletion Detection
- Administrative operations are logged
- Key lifetime events are logged
- Per key usage events are optionally logged
- Optional key usage logging
- Public key verification of audit logs
- Compatibility with syslog and Security Information and Event Management (SIEM).

20.1. Architecture

Audit logging is implemented on the nShield Connect. The audit logging functionality is on the embedded nShield Solo or nShield Solo XC card. The image below shows nShield Connect or Connect XC implementation. The audit log entries are generated on the module, the hardserver acts as a relay to a syslog infrastructure.



Given the architecture described above, the Audit Logging capability is implemented in the HSM (embedded Solo/Solo XC in the case of a nShield Connect). This means that the logging is at the level of nCore commands as processed by the HSM. The hardserver layer implements a higher-level abstraction which is presented to application clients. The information used to manage this such as Client Identifiers etc. is not available to the HSM and therefore cannot be logged.

20.2. Audit Logging Implementation

The Audit logging functionality is based on that described in RFC-5848 (<https://tools.ietf.org/html/rfc5848>). This describes a mechanism also known as syslog-sign that adds the following capabilities to syslog:

- Origin authentication
- Public verification
- Message integrity
- Replay resistance
- Message sequencing
- Detection of missing messages.

It is implemented on top of a standard syslog data stream and does not use any additional protocol. The syslog-sign logging scheme adds a number of control messages to the log entries that are to be audited. These are also implemented as syslog messages. The following sections outline the audit log entries that are present in the syslog data stream for nShield Audit Logging. The signing mechanism used is DSA with a 3072 bit key and SHA-256 as the hashing mechanism.

20.2.1. Audit log Entry

This is the log message from the entity being audited. It is in a standard format and includes operation specific data required to provide an auditing capability. As each log message is generated on the HSM, a digest operation is performed on it and the digest buffered in the HSM.

20.2.2. Signature Block

When sufficient digests have been accumulated (N), a Signature Block is generated as a standard log entry containing the following:

- Digests of the previous N messages
- Information to allow the digests to be matched with their respective log entries
- A signature across the digests and other information.

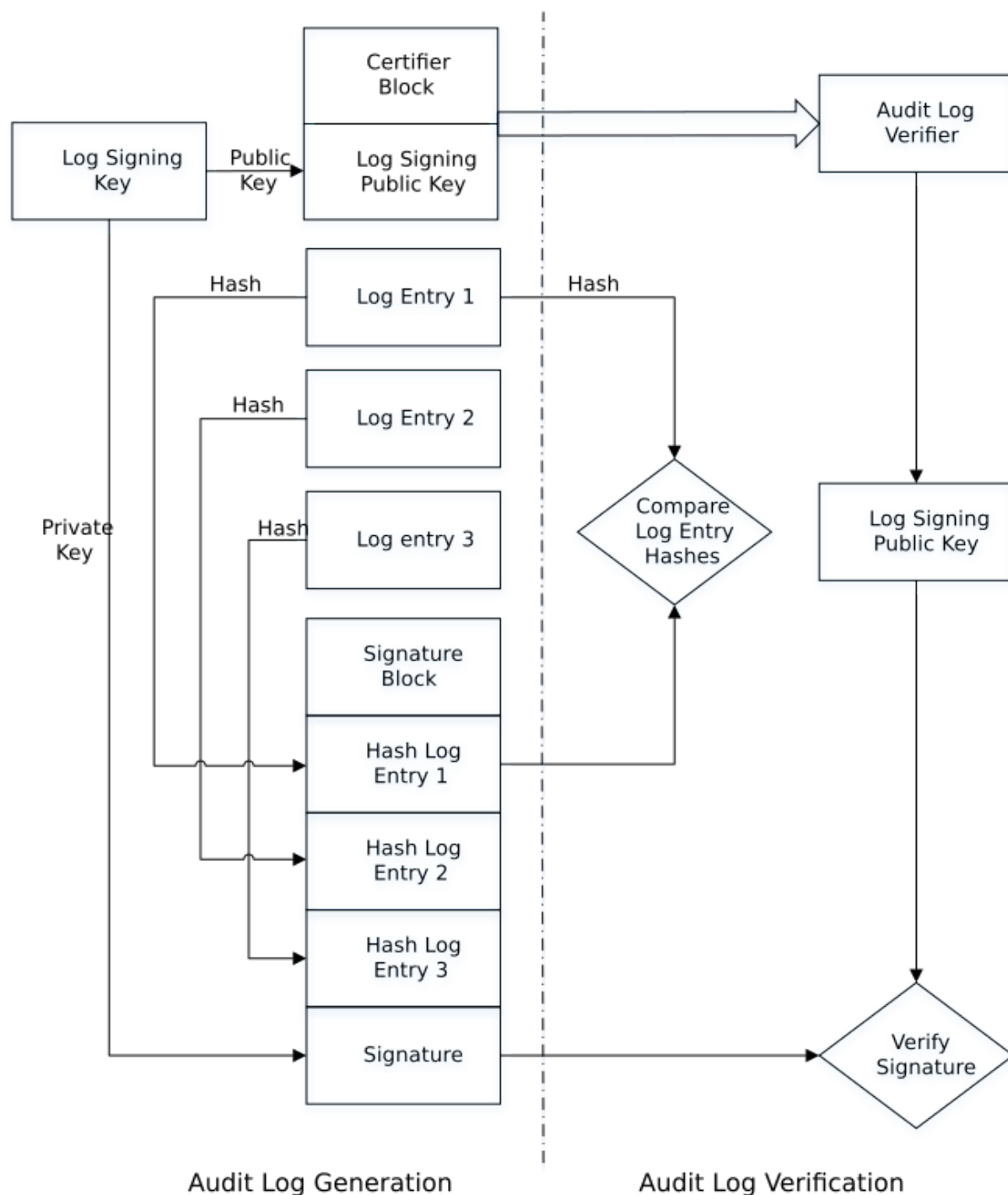
The number of messages is dictated by the transport medium, the size of the digests, the size of the signature and the size of other data contained in the message. There is a limit to the size of messages that can be transported over syslog. The signature is performed using a log signing key. This key is generated and the private half is held securely in the HSM.

20.2.3. Certifier Block

Verification of the Signature Blocks requires that you, or the application performing the verification, has access to the public half of the log signing key. The Certifier Block provides a mechanism for the log verifier to get access to this key. The key is packaged in a form allowing the source of the audit logs to be verified. As the size of this information may be too large for the syslog transport medium it can be broken down into Certifier Block Fragments which are compatible with the syslog transport mechanism. When all of these fragments are received by the log verifier, it can reconstruct the public half of the log signing key and perform any consistency checks and origin verification that is needed.

20.2.4. Audit Log Verification process

Given the public half of the log signing key, a Signature block and its corresponding log entries, the verifier can check the signature on the Signature Block. When this is verified, the log entry digests in the Signature Block are implicitly verified. The integrity of the corresponding log entries can be verified by performing a digest on received log entries and comparing them to the corresponding verified digests in the Signature block. The image below shows the basics of this approach. For more information, see [Audit Log Verification](#).



20.3. Configuring Audit Logging

Audit Logging is enabled per Security World and is configured on creation of the Security World. See [Creating a Security World using new-world](#). A Security World is created with Audit Logging enabled if either the `--audit-logging` or `-G` options are passed to the `new-world` command or the Security World is created in the `common-criteria-cmts` mode. This requires that the HSM is capable of audit logging and Security World creation will fail if the HSM does not support Audit Logging. Additional HSMs are indoctrinated into an Audit Log-

ging enabled Security World using the `new-world` command with the `--program` or `-l` options. The HSM must be capable of Audit Logging. If it is not capable the indoctrination will fail. Therefore all HSMs in an Audit Logging Security World are set to Audit Logging.

Configuring an Audit Logging Security World performs the following actions:

1. Audit Logging is set as enabled for this Security World and is recorded in the world file.
2. The HSM is initialized and:
 - A flag indicating the Audit Logging status is recorded in the HSM's EEPROM
 - A 3072bit DSA HSM specific Audit Logging Signing Key (KAL) is created and persisted in the HSM's EEPROM
 - A Certifier Block containing the public half of KAL is generated and sent to the log receiver via the hardserver.

When a new HSM is indoctrinated into an Audit Logging Security World the world file specifies that this is an Audit Logging Security World. The same actions as for initializing an HSM when the Audit Logging Security World was created are performed. This means that:

- All HSMs in an Audit Logging Security World have Audit Logging enabled
- Each HSM has a distinct Audit Logging Signing Key.

The Audit Logging entries are delivered over syslog using UDP transport. This transport must be configured before Audit Logging is enabled in order to collect the initial messages. The following section on configuring syslog describe the steps necessary to enable syslog for Audit Logging.

20.3.1. Confirming Audit Logging configuration

Audit Logging can be confirmed for the Security World using the `nfkminfo` command:



The Real Time Clock (RTC) on the HSM must be set and the setting confirmed after creating the Security World or indoctrinating an HSM into the Security World. The RTC on the HSM is used to timestamp audit log entries.

```
[audit1@myhost ~]$ /opt/nfast/bin/nfkminfo
World
generation 2
state      0x37270009 Initialised Usable Recovery !PINRecovery !ExistingClient RTC NVRAM
           FTO AlwaysUseStrongPrimes !DisablePKCS1Padding !PpStrengthCheck AuditLogging SEEDebug
n_modules  1
hkns0      4703d2cde058e88ce4c4bf8afe8e25d436b2f864
hkm        90284ba3e2b2019769d2fa6fc38532fae637bfe0 (type Rijndael)
hkmwk      c2be99fe1c77f1b75d48e2fd2df8dffc0c969bcb
hkrc       09928bfc675103b4f2fd937cce1cef8764ada20c
hkra       66679c64c5255b78b6bf06b30ea4f6634a222dd5
```

```

hkmc      aaa221d75803108a88534ad6dd1816a104720d0f
hkrtc     7dc8ab24343bd0bd2e24e071a2b8172aed0ecf43
hknv      f19fe4c416aa773e5518543543f7cfa3924a12a7
hkndsee   6729cbafd7e4c2f8e579d3e9d25c04c72b4f602d
hkfto     184d8c282814bba505b1147ed7bce4dc6b678967
hknull    0100000000000000000000000000000000000000
ex.client  none
k-out-of-n 1/1
other quora m=1 r=1 nv=1 rtc=1 dsee=1 fto=1
createtime 2018-05-03 19:22:17
nso timeout 10 min
ciphersuite Dlf3072s256mAEScSP800131Ar1
min pp     0 chars
mode       none
Module #1

```

Audit Logging is indicated by the presence of **AuditLogging** in the state line of the command's output. If Audit Logging was not enabled this line would show **!AuditLogging**. This indication applies to the Security World.

The enquiry command shows the Audit Logging status of the HSMs.

```

[audit1@myhost ~]$ /opt/nfast/bin/enquiry
Server:
enquiry reply flags  none
enquiry reply level  Six
.....
Module #1:
enquiry reply flags  none
enquiry reply level  Six
serial number       1111-2222-3333
mode                 operational
version              12.60.2
speed index          4512
rec.
queue                19..152
level one flags       Hardware HasTokens SupportsCommandState
version string        12.60.2-0-12d710d3920737e4d29426b9cf88eb0f21b12dd0 2018/05/03 12:00:21 UTC
checked in            000000005aeaf9d5 Thu May 3 08:00:21 2018
level two flags       none
max.
write size            8192
level three flags     KeyStorage
level four flags      OrderlyClearUnit HasRTC HasNVRAM HasNSOPermsCmd ServerHasPollCmds FastPollSlotList HasSEE
HasKLF HasShareACL HasFeatureEnable HasFileOp HasPCIPush HasLongJobs ServerHasLongJobs AESModuleKeys NTokenCmds
JobFragmentation LongJobsPreferred Type2Smartcard ServerHasCreateClient HasInitialiseUnitEx AlwaysUseStrongPrimes
Type3Smartcard HasKLF2 DevelopmentFirmware
module type code      7
product name          nC2023E/nC3423E/nC4033E
device name           #1 nFast PCI device, bus 3, slot 0.
EnquirySix version    7
impath kx groups      DHPrime1024 DHPrime3072
feature ctrl flags     LongTerm
features enabled       ForeignTokenOpen RemoteShare GeneralSEE KISAAlgorithms StandardKM CodeSafeSSL EllipticCurve
ECCMQV
version serial        28
rec.
LongJobs queue        18
SEE machine type       PowerPCSF
supported KML types    DSAp1024s160 DSAp3072s256
active modes           AuditLogging UseFIPSAApprovedInternalMechanisms AlwaysUseStrongPrimes
hardware status        OK

```

The HSM's Audit Logging status is indicated by the presence of **AuditLogging** in the **active modes** line.

20.3.2. Disabling Audit Logging

Audit Logging is set for the lifetime of the Security World. It can be disabled on an HSM if that HSM is removed from the Security World and reinitialized using **initunit**.

20.3.3. Configuring syslog



Prior to enabling Audit Logging on a Security World it is necessary to configure the syslog transport used by the Audit Logging facility.

The following example shows the relevant entries in the hardserver configuration file. Having set these entries it is necessary to restart the hardserver for them to be recognized.

If this configuration is being applied to a nShield Connect HSM, the configuration needs to be pushed to the Connect and the Connect hardserver restarted, see [nShield Connect utilities](#).

```
[auditlog_settings]
# Start of the auditlog_settings section
# Hardserver settings for audit logging.
# Each entry has the following fields:
#
# The port number Auditlogging server listens to .
auditlog_port=514
#
# IP Address of the Auditlogging server
auditlog_addr=WWW.XXX.YYY.ZZZ
#
# Copy auditlog to hardserver log. (default=no)
# auditlog_copy_hslog=ENUM
```

Field	Description
auditlog_port	This is the UDP destination port for Audit Logging syslog messages. The default is 514.
auditlog_addr	This is the IP address of the host to which the Audit Logging syslog messages should be delivered. The default is 0.0.0.0.
auditlog_copy_hslog	<p>This indicates that the syslog messages from Audit logging should be copied to the hardserver's log file. This provides some degree of redundancy but means that the hardserver's log file may grow significantly. The default is no.</p> <p>It is not recommended on nShield Connect HSMs.</p>

It is recommended that the syslog transport is checked before creating an Audit Logging Security World. This can be accomplished by sending a log message to the configured host and port using a command such as **logger** if it's available on your operating system, and the implementation of this command is capable of sending messages to a syslog server over UDP.

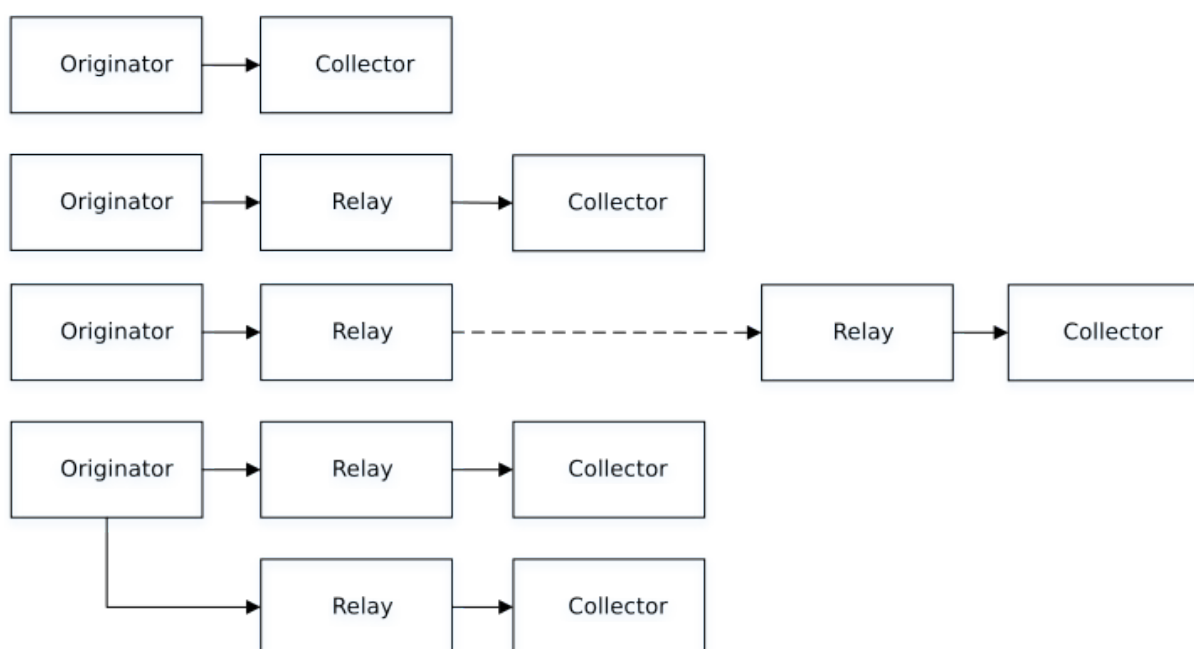
20.4. Log distribution

The nShield Audit Logging capability uses the RFC-3164 (<https://tools.ietf.org/html/rfc3164>) standard for distributing audit log messages. All audit log messages will have the following header prepended. This header is applied by the hardserver before sending the message and does not form part of the signed audit log messages. The signed portion of the audit log message starts at the **CEF:0** CEF identifier and continues to the end of the message.

```
<134>MMM DD HH:MM:SS hostname CEF:0.....
```

The PRI value of this header **<134>** indicates the facility **local0** and a severity of **info**.

The syslog infrastructure used for Log distribution is out of the scope of this guide and your responsibility to implement. Log distribution for Audit Logging uses syslog as the transport medium which allows a standard protocol and message format to be used for the Audit Logging messages. This transport is compatible with SIEM systems and the wider syslog infrastructure in an organization can be used to further distribute or process the log stream. There are many possible configurations of syslog deployment, as shown below.



20.5. Configuring log distribution

The actual implementation of the syslog infrastructure is at your discretion. Verification of the log messages requires that the verifying application has access to the audit logs from the HSMs in the Security World. The example verifier for the nShield Audit logging facility described in [Audit Log Verification](#) processes a file containing the audit log messages. It can process audit log messages from a specific HSM identified by its ESN or will use the first ESN found in the file.

It is recommended that logs from the nShield Audit Logging facility are separated from those from other applications. This can be accomplished by using the information in the audit log messages described in the section on Log Format. There are a number of entries that can be used to separate out the messages from the nShield Audit Logging facility. These include:

- Identifying elements in the CEF header:
 - Device Vendor
 - Device Product
- Identifying elements in the syslog header
 - Hostname or IP address of the machine hosting the hardserver which is distributing the audit log messages
- Using a distinct port for nShield Audit Logging see configuring syslog.

The log messages can be further split into those from specific HSMs using the ESN in the audit log messages.

As an example, the following **rsyslog** configuration will direct all messages with the string *nCipher Security* to a specific log file:

```
:msg, contains, "nCipher Security" /var/log/hsmauditlog
```

A similar strategy can be used with **syslog-ng**:

```
destination d_auditlog { file("/var/log/hsmauditlog"); };
filter f_auditlog { match("nCipher Security" value("MESSAGE")); };
log { source(s_log); filter(f_auditlog); destination(d_auditlog); }
```

Where **s_log** is the source receiving the audit logging messages.

Refer to the documentation for your syslog implementation for information on processing and distributing log messages.

20.5.1. Configuring syslog message infrastructure

It is important that the syslog infrastructure does not attempt to rewrite the log messages as this will affect the ability of the Audit Logging process to verify log messages. For example, the rsyslog default RFC-3164 parser will rewrite log messages interpreting the CEF:0 as a tag and will write **CEF: 0** to the log file. This means that an Audit Logging message persisted by the default RFC-3164 parser can not be verified as Audit Logging signs the log message starting at **CEF:**. You must configure your syslog infrastructure to preclude the signed part of the audit log message.

20.5.1.1. rsyslog

rsyslog can be configured to not reformat messages using the following approach:

1. Define a formatting template as shown below in the `/etc/rsyslog.conf` file. This should be added in the `##### MODULES #####` section of the rsyslog configuration file.

```
$template myFormat,"%rawmsg%\n"
```

2. Apply this formatting template to the processing of Audit Logging messages. For this example it is assumed that messages containing *nCipher Security* will be persisted in the file `/var/log/hsmauditlog`. You can use any other selection mechanism such as storing messages for a particular HSM as identified by its ESN in separate files.

```
:msg, contains, "nCipher Security" /var/log/hsmauditlog;myFormat
```

3. If the rsyslog server is going to be used as a relay, then the format needs to be applied to any relay statements in the rsyslog configuration file and to any receivers of the syslog message.

20.5.1.2. syslog-ng

`syslog-ng` does not appear to rewrite messages in the same way as `rsyslog`. Refer to the `syslog-ng` documentation for information on formatting.

20.6. Log format

20.6.1. CEF format

The audit log entries are emitted from the HSM in CEF format. This provides both human

readable log messages and compatibility with SIEM applications. As indicated in the previous section the audit log entries are distributed using the syslog transport mechanism.

A CEF format log message is shown below:

```
CEF:Version|Device Vendor|Device Product|DeviceVersion|Device Event Class ID|Name|Severity|[Extension]
```

Parameter	Description														
CEF:Version	This is mandatory and Version is currently 0.														
Device Vendor	This is nCipher Security														
Device Product	This identifies the family of nShield HSMs: <ul style="list-style-type: none"> • nShield Solo • nShield Solo XC • nShield Edge. 														
Device Version	This is the firmware version, for example 12.80.0.														
Device Event Class ID	<p>This is an identifier for the type of message:</p> <table> <tr> <th>Class ID</th><th>Description</th></tr> <tr> <td>1</td><td>nCore Commands</td></tr> <tr> <td>2</td><td>Internal HSM events: <ul style="list-style-type: none"> • Periodic heartbeat • Secure channel establishment </td></tr> <tr> <td>3</td><td>Audit logging control messages: <ul style="list-style-type: none"> • Signature Blocks • Certifier Blocks </td></tr> <tr> <td>4</td><td>Reserved</td></tr> <tr> <td>5</td><td>Shutdown messages</td></tr> <tr> <td>6</td><td>Reserved</td></tr> </table>	Class ID	Description	1	nCore Commands	2	Internal HSM events: <ul style="list-style-type: none"> • Periodic heartbeat • Secure channel establishment 	3	Audit logging control messages: <ul style="list-style-type: none"> • Signature Blocks • Certifier Blocks 	4	Reserved	5	Shutdown messages	6	Reserved
Class ID	Description														
1	nCore Commands														
2	Internal HSM events: <ul style="list-style-type: none"> • Periodic heartbeat • Secure channel establishment 														
3	Audit logging control messages: <ul style="list-style-type: none"> • Signature Blocks • Certifier Blocks 														
4	Reserved														
5	Shutdown messages														
6	Reserved														
Name	This is the event being logged. For Audit Logging, it is either the nCore command that is being logged, Cmd_Destroy for example, a description of the event such as heartbeat or one of ssign or ssign-cert which identifies either a Signature Block or a Certifier block.														

Parameter	Description															
Severity	This is an indication of the importance of the message.															
	<table> <tr> <th>Severity</th><th>Description</th></tr> <tr> <td>1</td><td>nCore Commands</td></tr> <tr> <td>2</td><td>Internal HSM events: <ul style="list-style-type: none"> • Reboot events • Secure channel establishment </td></tr> <tr> <td>3</td><td>nCore Commands that force a Signature Block flushing buffered message hashes.</td></tr> <tr> <td>4</td><td>Periodic Heartbeat messages</td></tr> <tr> <td>5</td><td>Audit Logging control messages: <ul style="list-style-type: none"> • Signature Blocks • Certifier Blocks </td></tr> <tr> <td>6</td><td>Shutdown messages</td></tr> <tr> <td>10</td><td>HSM Error messages</td></tr> </table>	Severity	Description	1	nCore Commands	2	Internal HSM events: <ul style="list-style-type: none"> • Reboot events • Secure channel establishment 	3	nCore Commands that force a Signature Block flushing buffered message hashes.	4	Periodic Heartbeat messages	5	Audit Logging control messages: <ul style="list-style-type: none"> • Signature Blocks • Certifier Blocks 	6	Shutdown messages	10
Severity	Description															
1	nCore Commands															
2	Internal HSM events: <ul style="list-style-type: none"> • Reboot events • Secure channel establishment 															
3	nCore Commands that force a Signature Block flushing buffered message hashes.															
4	Periodic Heartbeat messages															
5	Audit Logging control messages: <ul style="list-style-type: none"> • Signature Blocks • Certifier Blocks 															
6	Shutdown messages															
10	HSM Error messages															

20.6.2. CEF extensions

The rest of the log message is made up of CEF extensions. These are name/value pairs that are used to convey specific information for the log message. The name-value pairs can be processed by SIEM applications such as Arcsight and can be displayed in tabular reports of the messages received. They can be used for filtering and further processing within the SIEM application. The following table specifies the meaning and format of the extensions used by the Audit Logging facility.

Extension Name	Description
esn	Electronic Serial Number (ESN) of the HSM in the following format: XXXX-XXXX-XXXX
rtc	Time-stamp from the HSM's Real Time Clock (RTC) as ms since the epoch (1970 Jan 01 00:00:00 UTC).
outcome	Outcome of the operation - success or failure
hkey	Identifying nCore key hash for the main key of the command being logged as a 40 character hex string

Extension Name	Description
<code>hbase</code>	Identifying nCore key hash for the base key of a <code>Cmd_DeriveKey</code> command being logged as a hex string
<code>hwrap</code>	Identifying nCore key hash for the wrap key of a <code>Cmd_DeriveKey</code> command being logged or the logical token hash for key blobbing operations as a hex string
<code>hin3-5</code>	Identifying nCore key hashes for the remaining keys of a <code>Cmd_DeriveKey</code> command being logged as hex strings
<code>hknso</code>	Identifying nCore key hash of Security Officer's key as a hex string
<code>htok</code>	Identifying nCore Logical Token hash as a hex string
<code>shareindex</code>	Index of share being operated on by Logical Token functions as a decimal number
<code>sharesleft</code>	Number of Logical Token shares left to read or write as a decimal number
<code>tokenslot</code>	Slot number for Logical Token operations as a decimal number
<code>sharesneeded</code>	Quorum required to reconstruct a Logical Token as a decimal number
<code>sharetotal</code>	Total number of shares for a Logical Token as a decimal number
<code>timelimit</code>	How many seconds after reassembly the Logical Token is usable for
<code>shorthash</code>	Short Logical Token hash used in <code>Cmd_EraseShare</code> and <code>Cmd_ChangeSharePIN</code>
<code>hkm</code>	Identifying nCore hash of module key KM
<code>mode</code>	Mode that a channel is opened in. One of encrypt , decrypt , sign or verify
<code>source</code>	Source of command. One of host , SEE or internal
<code>flags</code>	<p>Flags supplied to <code>Cmd_SetNSOPerms</code> and <code>Cmd_InitialiseUnitEx</code>.</p> <p><code>Cmd_SetNSOPerms</code></p> <ul style="list-style-type: none"> • AlwaysUseStrongPrimes • DisablePKCS1Padding • FIPSLevel3Enforcedv2 • CommonCriteriaCMTSRestrictions <p><code>Cmd_InitialiseUnitEx</code></p> <ul style="list-style-type: none"> • AuditLogging • UseFIPSAprovedInternalMechanisms
<code>slotcount</code>	Count of Dynamic Slots to be configured
<code>slotid</code>	Dynamic Slot to create association for

Extension Name	Description
<code>prevrtc</code>	The previous value of the HSM's RTC as ms since the epoch. Used to indicate previous value of the RTC before a <code>Cmd_SetRTC</code> timestamp or an event occurring before a restart
<code>smartcardesn</code>	ESN of smartcard used for Dynamic Slot operations
<code>kmltype</code>	Type of the Module Per-Initialization Signing Key (KML) set by <code>Cmd_InitialiseUnit(Ex)</code>
<code>sos</code>	Indication of the sos code

20.6.3. Infrastructure extensions

The Audit Logging Implementation requires a number of infrastructure CEF extensions to provide data necessary for the RFC-5848 based signed syslog approach used. Please refer to RFC-5848 for further details on these infrastructure extensions. These CEF extensions replace the RFC-5424 Structured Data used in the original scheme but have the same meaning.

20.6.4. Message and reboot counters

There are two counters that are sent with all Audit Logging command log messages. The Reboot Session ID is also sent with Certifier Block and Signature Block messages.

Counter	Description
<code>seqNo</code>	<p>Log message sequence number as a decimal number. This a counter that has a range of 1 to 9999999999. When <code>seqNo</code> reaches 9999999999 it is reset to 1. It is incremented for every command log message sent. This is not part of RFC-5848 and has been added to provide a direct mechanism for detecting deleted or missing log messages.</p> <p>The sequence number is unique in the context of a reboot session. When <code>rsid</code> is incremented <code>seqNo</code> is reset to 1.</p>
<code>rsid</code>	Reboot Session ID as a decimal number. This a counter that has a range of 1 to 9999999999. When <code>rsid</code> reaches 9999999999 it is reset to 1. It is incremented every time the HSM is restarted and whenever the Global Block Counter (<code>gbc</code>) reaches its limit and is reset.

20.6.5. Certifier Block extensions

The following extensions are used in the Certifier Block where the CEF header name element is `ssign-cert`. The Certifier Block is used to distribute the log signing public key. It is

sent by the HSM when logging is enabled and every time the HSM is restarted. This provides for redundancy as any Certifier Block from an HSM that has been configured for Audit Logging will contain the same log signing public key. The Certifier Block can extend over multiple syslog messages. The extensions identified here allow the data of a Certifier Block to be rebuilt from multiple fragments. Sufficient fragments are sent in separate **ssign-cert** messages to rebuild the payload block. See [Certifier Block example](#) for the details of the data included in the certifier block.

Name	Description
tpbl	Total Payload Block Length. This is the length of all Certifier Block fragments.
findex	Index of this fragment (1 based) as a decimal number.
flen	Length of the fragment as a decimal number.
frag	Base 64 encoded Certifier Block fragment.
sign	DSA signature using KAL of the data in each fragment up to the sign extension. The DSA signature is DER encoded and then base64 encoded. It is present here to support consistency checking.

20.6.6. Signature Block extensions

The following extensions are used in the Signature Block where the CEF header name element is **ssign**. The Signature Block supports the verification of Audit Logging messages. The Signature Block is sized to fit within a syslog message which dictates the number of audit log messages it covers. This release supports a maximum of 10 audit log messages per Signature Block. The main data for the Signature Block is the SHA-256 hashes of the log messages covered by the block.

Name	Description
gbc	Global Block Counter as a decimal number. Count of signature blocks sent in this Reboot Session prior to this Signature Block. This is a decimal number that has a range of 1 to 999999999. When gbc reaches 999999999 it is reset to 0. At the same time the rsid is incremented and seqNo reset to 1. It is incremented after every Signature Block is sent.
fmn	First Message Number as a decimal number. First log message seqNo in this Signature Block.
hcnt	Count of log messages included in this Signature Block as a decimal number.
hb	The log message SHA-256 hashes base64 encoded and separated by the & character. The & character does not occur in base64 encoding and avoids SIEM issues with embedded spaces.

Name	Description
sign	DSA signature using KAL of the data in the Signature Block up to the sign extension. The DSA signature is DER encoded and then base64 encoded.

20.6.7. Example Audit Logging messages

This section shows example Certifier Block, Signature Block and Audit Logging messages and shows how the CEF extensions are used together.

20.6.7.1. Certifier Block example

This is an example Certifier Block produced after a reboot of the HSM. The log messages have been reformatted for display as each one can be up to 1024 characters long. The Reboot Session ID (rsid) is 8. There are five fragments in this example. The first four are 450 characters and the final 340 long for a total length of the payload of 2140 characters. The Event Class Id is 3 and the severity is 5 identifying these as infrastructure messages.

```
<14>Aug 10 18:22:18 nethsm CEF:0|nCipher Security|nShield SoLo XC|12.60.9|3|ssign-cert|5|esn=7109-02E0-D947
rsid=9 rtc=1628616065737 tpb1=2140 findex=1 flen=450
frag=uwAAAEQAAAArPjRM6QnKObhNt0va3OSSdLSTSEK02Ydxj2sRmsTvmDJ5tqXeNE0p7GkmFfg0/yU2iT19VLSJa70pq+BKNTw3wEAAEQAAACI
WksBX/JuSf8C9v3WRfnUWvvCAHID+OGaoYvHqqlVtW7GzRMWmg/fPe03UV32w2i+GkORU87i6dJKybxqhwVvQwAAAAKwFAAAEAAAAAAAAAAAAACA
AADwAAADcxMDktMDJFMC1E0TQ3AAANAAAA00XfBuCMHaS7g0dVUE5vnVx43guAAAABgAAAAAABEAAAA4P0MEe9CDHohvBeOvBwLkF99mopVY
1Rdam8554PDJWwhwID6+utn8AJGOAscYtfrOxabeJ1iC+btrmJZbrLrXQBAABEAAAAAd1xLlbwppzAjT8/H+off7ikbnLNQdozuNJeNaqb4+IZwCxc
NwA

<14>Aug 10 18:22:18 nethsm CEF:0|nCipher Security|nShield SoLo XC|12.60.9|3|ssign-cert|5|esn=7109-02E0-D947
rsid=9 rtc=1628616065739 tpb1=2140 findex=2 flen=450
frag=s3z1d9CeG3/LQpv6zXv9mC2HoFDes/dP/x3eUAAAC7AAADGAAAPKa01J+3cWja40ICn9Bog4a4I1RAwAAAIABAADjwRLJ1KVAir+H1VAUCW
ojKksMqGyWghwhMoqYP8ldIy7bb3UVQBp6M+fxVpSFFrz3bfDgJQNh/13YcAY1+r1JYvEner7cnGatDIjnMgNqQPN6a1qM787pMz3/eIq0L0xI8rV
y99F/foV6aFcJVCvxsjL9wIQ0d4AhjIgtFPTiAEC4UTL5Eg9YkKnjZXizpTxhReSZVMjIM8Fu2sjcvzh1Q8PQYcEuU5sZhQbLVjUvRpu2HpgOTw
heXW+X4gWpMLXsVkwV7F24j4Ax6eiyasP1HCx4savMxcyA3cxwp7dTUJnFPVr8nfpqp2H3ai3khSIefMc7d8gyajJn0L1JQMzf605Hr1Veuixd6hB
dwd

<14>Aug 10 18:22:18 nethsm CEF:0|nCipher Security|nShield SoLo XC|12.60.9|3|ssign-cert|5|esn=7109-02E0-D947
rsid=9 rtc=1628616065740 tpb1=2140 findex=3 flen=450
frag=B1Ku9rirlixkgEd+73tMVJ1FQz85aCWuRqJ104YB1YwFvZgvrXhHvzqLFeJZAuerK1LgIaZwDq1twoXzvHq88QcJdbr0i4+87VorPKkEjKtS
SGHOVkkHhBC8uNgYXnTBxqcqCqpZ14whuiEBmJQLcwgAAAAG8rgckmo3ArobecQooPxQ9AjYbCmAoKOUTRi7grTzPyAAQAA3Bvuz+tQ1uh5LvukM
LTtGDTTp1G7ks6ZkL8b+F2UW37jfn3lap27oAZq1otU4FOP4EVvoMmNSdI4uzCPi7VgcI3AcIkdjZIwbpYf9XQwvFwMxYvBPgHptc/t8Lslgs97r
MkES4Zc1NI/NwJkP0fw4kC1SBSUQUAUp6vggg2vVL9naqRHhXNRJuweaRt0060z0mBkTgCnAvscdr2ymErrWDZARHosYXJZrXghjNmXuu+rS86vT
vTc

<14>Aug 10 18:22:18 nethsm CEF:0|nCipher Security|nShield SoLo XC|12.60.9|3|ssign-cert|5|esn=7109-02E0-D947
rsid=9 rtc=1628616065742 tpb1=2140 findex=4 flen=450
frag=189gfrjMpL5aBYYAk11XqWDGHHfcltTSzCMgWalxMOeOQxaZLbzYDt12/udXIo0bn/PTga0kPYTypvzFwsQM4axpDzVYCE064YvoyjUgWB
U4kMLC4JuH5ytJAM+uA67xu36Iqx3j/mjMozTR1rGJuH+b314zgHRjvfr8AA0juixN8tdxkFFRQLzYC9Ulw4g6f0Sa06ecBIOA5Q0ylvdVjqmcX2+
+J+snC0wtXhV+yLKW8m7/DDjExTXKpHo5EqyQB24tHCogAEAAJRhmGdRJR9i6dxLTUzQFo7ZzcQpZFdbFF4TN5+8Z/TER2/toZg+onQkD2Eshd/T4
4p3WQr1XkQfJf3/syQu8p54Q0inmZzx6leFN99pXUhxPnNIIPuc8UHJWips7nQnlsa3ER6evViNqanQSBTYEmuibXRITin8NM3h0RTJekDgw1J2
wj0

<14>Aug 10 18:22:18 nethsm CEF:0|nCipher Security|nShield SoLo XC|12.60.9|3|ssign-cert|5|esn=7109-02E0-D947
rsid=9 rtc=1628616065743 tpb1=2140 findex=5 flen=340
frag=DjWEMgrKNb0X6hYqOPg/Ozid2ytohqdRx0Hr2ze16Ha17HgFfAkX1YaPxIdIgrsuNwub43HU9iVByMdSe8kj0igEW/jXzTZQMFPy/Iy1+GC
V3P8GW+liYE/DM5auWH1a9NwbftjGTWQK4A820Eqy9zSUz0Bnbn/gCLVUUMBsAGKY41Dtx7pUNU0TQpEMydBmfaoQhyx07fj070t8Zc2Ut2e5a2s/u
xbw0nSQNYJ25SDBd8UjHhntNJVHiANHU3Qu1JcUnEKUwQW00dLeLoMtD2Ldy+QaHHjsk7WYHboWCCSsWiHa4z3nVQsr3BrkrkxBUliLWtXQ1VVMaEa
```

oAAAA=

20.6.7.2. Log Messages and Signature Block

This example shows a sequence of audit log messages with a Signature Block after 10 messages. These are in the same **rsid** as the previous example. The log sequence number for this excerpt starts at 31 and the last log message before the Signature Block is sequence number 40. The name element identifies the command being executed by the HSM. Each of the example commands operates on an nCore Key and this is identified by the nCore key hash of the relevant key.

The Signature Block has name element **ssign** identifying it as a Signature Block. The **gbc** is 6 meaning this is the 7th Signature Block in this Reboot Session ID. The **fmn** is 31 and **hcnt** is 10 meaning that this Signature Block covers messages 31 to 40. As Audit Logs are generated this sequence will be repeated. Once this Signature Block has been received and with the log signing public key available the signature on this Signature Block can be verified and then the hashes of the individual log messages can be calculated and compared with the hashes recorded in the Signature Block for the corresponding log message to allow the detection of tampering.

```
<134>May 15 12:42:09 myhost CEF:0|nCipher Security|nShield Solo|12.60.2|1|Cmd_Destroy|1|esn=1111-2222-3333 rsid=8
rtc=1526388047690 seqNo=31 source=host outcome=success
hkey=c4ab637985a542e7eb3eb4838f57872d5422bbb4

<134>May 15 12:47:09 myhost CEF:0|nCipher Security|nShield Solo|12.60.2|1|Cmd_Destroy|1|esn=1111-2222-3333 rsid=8
rtc=1526388347977 seqNo=32 source=host outcome=success
hkey=c4ab637985a542e7eb3eb4838f57872d5422bbb4

<134>May 15 12:52:10 myhost CEF:0|nCipher Security|nShield Solo|12.60.2|1|Cmd_Destroy|1|esn=1111-2222-3333 rsid=8
rtc=1526388648265 seqNo=33 source=host outcome=success
hkey=c4ab637985a542e7eb3eb4838f57872d5422bbb4

<134>May 15 12:53:21 myhost CEF:0|nCipher Security|nShield Solo|12.60.20|1|Cmd_GenerateKeyPair|1|esn=1111-2222-
3333 rsid=8 rtc=1526388719548 seqNo=34 source=host outcome=success
hkey=cec7b0b1ef47d4141d65fdde9f9d23e854391dea

<134>May 15 12:53:21 myhost CEF:0|nCipher Security|nShield Solo|12.60.2|1|Cmd_Export|1|esn=1111-2222-3333 rsid=8
rtc=1526388719549 seqNo=35 source=host outcome=success
hkey=cec7b0b1ef47d4141d65fdde9f9d23e854391dea

<134>May 15 12:53:21 myhost CEF:0|nCipher Security|nShield Solo|12.60.2|1|Cmd_Export|1|esn=1111-2222-3333 rsid=8
rtc=1526388719549 seqNo=36 source=host outcome=success
hkey=cec7b0b1ef47d4141d65fdde9f9d23e854391dea

<134>May 15 12:53:21 myhost CEF:0|nCipher Security|nShield Solo|12.60.2|1|Cmd_Destroy|1|esn=1111-2222-3333 rsid=8
rtc=1526388719550 seqNo=37 source=host outcome=success
hkey=cec7b0b1ef47d4141d65fdde9f9d23e854391dea

<134>May 15 12:53:21 myhost CEF:0|nCipher Security|nShield Solo|12.60.2|1|Cmd_Import|1|esn=1111-2222-3333 rsid=8
rtc=1526388719550 seqNo=38 source=host outcome=success
hkey=cec7b0b1ef47d4141d65fdde9f9d23e854391dea

<134>May 15 12:53:21 myhost CEF:0|nCipher Security|nShield Solo|12.60.2|1|Cmd_Destroy|1|esn=1111-2222-3333 rsid=8
rtc=1526388719551 seqNo=39 source=host outcome=success
```

```

hkey=cec7b0b1ef47d4141d65fdde9f9d23e854391dea

<134>May 15 12:53:21 myhost CEF:0|nCipher Security|nShield Solo|12.60.2|1|Cmd_Import|1|esn=1111-2222-3333 rsid=8
rtc=1526388719552 seqNo=40 source=host outcome=success
hkey=cec7b0b1ef47d4141d65fdde9f9d23e854391dea

<134>May 15 12:53:21 myhost CEF:0|nCipher Security|nShield Solo|12.60.2|3|ssign|5|esn=1111-2222-3333 rsid=8
rtc=1526388719552 gbc=6 fmn=31
hcnt=10hb=8ogF/vsd9SwQ+qWEneaLDuczRE9XbE9Rf3k3dc51SXo=8Xq5dbTetg016pHQ7n6G16X+muB6C6VzN4FKbuqZqNUQ=8tOdfZk5Uvs6W9
E8mJyEU4kkNHAImwYft0v+7mHSL7VY=8ct+wqWstf+asw9ppvYVnbmkpqU3/WbXm5nHJPri9E=8r iegp83m5c3jYt2vymNf61ov+Jf8JqCeLSyh
iSDRXfA=8WgKibqrl7AEKXU3Wu2IG1VcetIIESxyXlLcAFR+qvE=8TDrCHjkuA2fS5ZBgVu4Wspta+MbdhkyrGxHhWn9Xck=8Ewf1NUbAIIvVdj
06P1EvjlljRoIXiprw56cq6GLbp9w=8Y1j1x6GE0ofqG9XdAqk91zFhA8bczBoesttvvNND3tI=8p5YnBsabKWL4F5+2WNmIJ9DmKuSeUz5v3qog0
ilztRY=sign=MEUCIDa7RJ1KmMRJ4KrDWxDYYok1t9ptQXqH1nPE5xLihegoAiEA4JWBtXsF/XrN2kJVfiVpCRicbvgTNjjjnouhS6QTM=

```

20.7. Commands Audited

The Audit Logging facility generates log entries on the module for a set of nCore commands and module operations. The commands and information logged for each command are described in the following sections.

20.7.1. Key usage logging

By default the nShield Audit Logging Facility does not log usage of keys for cryptographic operations such as sign, verify, encrypt and decrypt or their usage in channels for these purposes. Creation, Deletion and a number of other key operations are unconditionally logged by default. The Audit Logging feature provides the capability to optionally log these operations. This is determined on a per-key basis by the LogKeyUsage permission group flag on the ACL group authorizing the operation for which logging is desired. See the *nCore Developer Tutorial* for further information on ACLs.

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

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

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

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

```

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

```

In the following sections, the tables will indicate if this mechanism is required to generate a log message for a specific command or key.

20.7.2. Commands generating Audit Log messages

The following tables list the nCore commands that generate Audit Logging messages. For each command they identify command specific data that is contained in the log message and the CEF extension used to identify it.

nCore Key and Logical Token hashes are the standard nCore identifying hashes. They are used to identify a key or logical token as it is an invariant for the key or logical token. These hashes are logged as lower-case hex encoding. In some cases a short hash may be presented. This is the first 10 bytes of the hash in a lower-case hex encoding.

For each command logged the command is specified by the **name** element of the CEF header. The other elements of the CEF header are filled as detailed in the previous section. All commands being logged will also include the following CEF extensions:

Extension	Description
esn	ESN of the HSM
rsid	Reboot Session ID
rtc	Timestamp as milliseconds since the epoch derived from the HSM's Real Time Clock
seqNo	Sequence number of the Audit Log message
outcome	Success or failure



Identifying Log Messages: As Audit Logging will potentially be running for a long time, the identification of a log message from an HSM based on the **rsid** and **seqNo** will not hold if the HSM is not restarted before the **seqNo** is reset when it reaches 9999999999. In this case account can be taken of the **gbc** as this will increment at a slower rate than the **seqNo**. Therefore messages in the same **rsid** with the same **seqNo** will have significantly different values of **gbc** (the mapping between **seqNo** and **gbc** is determined by the Signature Block containing the message in question). When the **gbc** is reset the **rsid** is incremented so counting begins in a new Reboot Session. Account can also be taken of the value of the **rtc**.

20.7.3. Key commands


Commands with **Yes** in the **Requires logkeyusage ACL** column will only be logged if the Key's ACL contains the **LogKeyUsage** Flag in the permission group authorizing the operation.


Command	Command Specific Information Logged	Extension	Requires logkeyusage ACL
Cmd_Sign	nCore key hash	hkey	Yes
Cmd_Encrypt	nCore key hash	hkey	Yes
Cmd_Decrypt	nCore key hash	hkey	Yes
Cmd_Verify	nCore key hash	hkey	Yes
Cmd_ChannelOpen	nCore key hash	hkey	Yes
	channel mode	mode	-
	 Mode is one of encrypt, decrypt, sign, verify		
Cmd_Import	nCore key hash	hkey	No
Cmd_Export	nCore key hash	hkey	No
Cmd_Duplicate	nCore key hash	hkey	No
Cmd_GenerateKey	nCore key hash	hkey	No
Cmd_GenerateKeyPair	nCore key hash	hkey	No
	 nCore Key Hashes of private and public key halves are identical		
Cmd_SetAppData	nCore key hash	hkey	No
Cmd_SetACL	nCore key hash	hkey	No
Cmd_Destroy	nCore key hash	hkey	No
	Also used for Logical Tokens		
Cmd_DeriveKey	nCore Key Hash of derived key	hkey	No
	nCore Key Hash of base key	hbase	Yes
	nCore Key Hash of wrap key	hwrsp	Yes
	nCore Key Hash of third input key	hin3	Yes
	nCore Key Hash of fourth input key	hin4	Yes
	nCore Key Hash of fifth input key	hin5	Yes
	 The nCore Key Hashes for the input keys will only be included in the audit log if the Permission Group for the DeriveKey action has the LogKeyUsage flag.		

Command	Command Specific Information Logged	Extension	Requires logkeyusage ACL
Cmd_MakeBlob	nCore Key Hash	hkey	No
	nCore LT Hash	hwrap	
Cmd_LoadBlob	nCore Key Hash	hkey	No
	nCore LT Hash	hwrap	
Cmd_SetKM	nCore key hash	hkey	No
Cmd_RemoveKM	nCore key hash	hkey	No

20.7.4. Logical Token and Share Commands



These commands do not use the logkeyusage ACL mechanism and log unconditionally.



Command	Command Specific Information Logged	CEF Extension
Cmd_ChangeSharePIN	KM nCore Key Hash	hkm
	Short LT Hash	shorthash
	Share Index	shareindex
	Slot	tokenslot
Cmd_Destroy	LT Hash	hkey
	 Cmd_Destroy is used for Logical Tokens as well as Keys	
Cmd_EraseShare	Short LT Hash	shorthash
	Share Index	shareindex
	Slot	tokenslot
Cmd_GenerateLogicalToken	KM nCore Key Hash	hkm
	nCore LT Hash	htok
	Token Shares Needed	sharesneeded
	Token Total Shares	sharestotal
	Token time-limit	timelimit

Command	Command Specific Information Logged	CEF Extension
Cmd_LoadLogicalToken	KM Hash	hkm
	LT Hash	htok
	Token Shares Needed	sharesneeded
	Token Total Shares	sharestotal
	Token time-limit	timelimit
Cmd_ReadShare	LT Hash	htok
	Share Index	shareindex
	SlotId	tokenslot
	Share Left	sharesleft
	 This is the remaining number of shares required to reconstruct the Logical Token. It reduces to 0 when a quorum of the Shares have been read.	
Cmd_WriteShare	LT Hash	htok
	Share Index	shareindex
	SlotId	tokenslot

20.7.5. Administrative Commands


These commands are logged unconditionally.

Command	Command Specific Information Logged	CEF Extension
Cmd_InitialiseUnit	Type of KML key	kmltype
	 DSAP3072s256	
Cmd_InitialiseUnitEx	Type of KML key	kmltype
	InitialiseUnitEx Flags	flags
	 DSAP3072s256 Combination of AuditLogging and UseFIPSAp- provedInternalMechanisms	

Command	Command Specific Information Logged	CEF Extension
Cmd_SetNSOPerms	nCore Key Hash of Security Officers Key SetNSOPermsFlags	hkns0 flags
	 Combination of AlwaysUseStrongPrimes, DisablePKCS1Padding, FIPSLen3Enforcedv2 and CommonCriteriaCMTSRestrictions	
Cmd_CreateSeeWorld		
Cmd_SetSEEMachine		
Cmd_SetRTC	Previous RTC	prevrtc
	 New RTC value will be shown in the rtc extension	


20.7.6. Dynamic Slot Commands

These commands are logged unconditionally.

Command	Command Specific Information Logged	CEF Extension
Cmd_DynamicSlotsConfigure	Count of Dynamic Slots to be Configured	slotcount
Cmd_DynamicSlotCreateAssociation	Slot Id for Association	slotid
EstablishSecureChannel	ESN of smartcard	smartcardesn
	 This internal event is logged with name element EstablishSecureChannel, a Severity of 2 and a Device Event Class Id of 2 in the CEF header and with source=internal in the CEF extensions.	

20.7.7. Heartbeat

The heartbeat is a periodic audit log message sent every 15 minutes. This audit log message indicates that the HSM is still active. After a heartbeat event is logged a Signature Block is generated including the heartbeat log message and any outstanding audit log messages. Waiting until the heartbeat is logged before restarting the HSM will ensure outstanding log messages can be verified.

Command	Command Specific Information Logged	CEF Extension
heartbeat	nCore Key Hash of Security Officers Key	hknso
	 <p>The heartbeat is logged in the CEF header with name element heartbeat, Severity 4, and Device Event Class Id 2. In the CEF extensions it's logged with source=internal.</p>	

20.7.8. Post Reboot Logging

The nShield HSM has a number of commands and errors that cannot be logged directly when they occur. This applies primarily to errors detected during processing or self test and the reboot command **Cmd_ClearUnit**. The strategy adopted for these is to persist sufficient information and replay them as log entries after a successful reboot of the HSM. These reboot event messages occur after the Certifier Block has been emitted.

Each of these messages are emitted with **rsid** and **seqNo** relating to the current session and will have a prevrtc CEF element recording the RTC at the time of the event. The name element will identify the event. If the event is associated with a nCore SOS code this will be indicated by a sos CEF extension and an appropriate code. The Device Event Class Id is set to 5 and Severity will be set to 10 for errors or 6 for shutdown events. The source CEF extension will be internal. The following table lists the events replayed in a post reboot log. The available events depend on the type of HSM.

Event Id	Event	SOS Code
Cmd_ClearUnit	Cmd_ClearUnit	
Cmd_Fail	Cmd_Fail	D
Environment_SensorFail		HV
Temperature_OutofRange		T
RNG_PeriodicTestFail		H RTP
SOS	Starting up crypto offload	HF
SOS	cache keygen failed	HR
Voltage_Tamper		V
Battery_Tamper		B
Unknown_Tamper		TAMPER

Event Id	Event	SOS Code
SelfTestFail	POST test timed out	HC0TTO
	POST test failed: lock failure detected	HC0LC
	POST test failed: TEST_STARTED	HC0TS
	POST test failed: PROCESS_STARTED	HC0PS
	POST test failed: CPUID_CHECK	HC0CC
	POST test failed: SRAM_ALLOC	HC0SA
	POST test failed: SRAM_WRITE	HC0SW
	POST test failed: SRAM_READ	HC0SR
	POST test failed: SRAM_FREE	HC0SF
	POST test failed: CRAM_ALLOC	HC0CA
	POST test failed: CRAM_GETCACHED	HC0CG
	POST test failed: CRAM_WRITE	HC0CW
	POST test failed: CRAM_READ	HC0CR
	POST test failed: CRAM_FREE	HC0CF
	POST test failed: LOCK_CHECK	HC0LC
	POST test failed: RTC_CHECK	HC0RT
	POST test failed: KAT_DSA	HC0KS
	POST test failed: KAT_ECDSA	HC0KC
	POST test failed: KAT_DES	HC0KE
	POST test failed: KAT_DES3	HC0KF
	POST test failed: KAT_DES3CBCMAC	HC0KO
	POST test failed: KAT_AES	HC0KA
	POST test failed: KAT_AESCMAC	HC0KB
	POST test failed: KAT_AESCBCMAC	HC0KD
	POST test failed: KAT_SHA1	HC0KH
	POST test failed: KAT_SHA1HMAC	HC0KM
	POST test failed: KAT_SHA224HMAC	HC0KN
	POST test failed: KAT_SHA256HMAC	HC0KJ
	POST test failed: KAT_SHA384HMAC	HC0KP

Event Id	Event	SOS Code
	POST test failed: KAT_SHA512HMAC	HC0KI
	POST test failed: KAT_RSA	HC0KRH
	POST test failed: KAT_NISTKDF	HC0KDF
	POST test failed: KAT_HASHDRBG	HC0HD
	POST test failed: KAT_RSAAEP	HC0KZ
	POST test failed: KAT_25519	HC0KX
	POST test failed:unknown	HC0H

As an example, the following shows a post reboot log of **Cmd_ClearUnit**. In this excerpt, it can be seen after the last fragment of the Certifier Block. A Signature Block is generated after the reboot log entries.

```
....
<134>May 16 15:08:45 myhost2 CEF:0|nCipher Security|nShield Solo XC|12.60.2|3|ssign-cert|5|esn=1111-2222-4444
rsid=2 rtc=1524140117693 tpb1=2140 findex=5
flen=340frag=3/ITRJT4T/qgd2ZEJufIzCR+nR9IngOrmogj+5JM7VMFLsWGDxUqxmFlpqs52T2zWuYIeFHGQfx9WS9PUhf2eLMYf/7onn+hFU55
7/GSZ1GbCnxWybfPN27oyXjHE7pfyOrWRVK1Iw8UULHVezVsxeIsZuuNEsZa5gUQ++DkoTu5M2BoPr4A+6dVL2eDh0F1m2zKATfk2moW93GkA3A07
LNPV5xU76ujo2tT7Mttvg+vyddiF2UWe6n75U0FMFjLM9WnhpFAhNk9mJPnZ5smf4i9JuNKZat+5tq5w2b/a8Sy01EVEktJI5SSjahp5z77RseQ
8H8ytsw6oAAAA=sign=MEUCIHgrF1m7t9X5xsL/gXwLju0bPFFPjJeIeIh8TKSN7prAiEAs3lPS62zX3TE940/Dw9/1gVradNi62wrQI+WLSI4IY
U=
<134>May 16 15:08:45 myhost2 CEF:0|nCipher Security|nShield Solo XC|12.60.2|5|Cmd_ClearUnit|6|esn=1111-2222-4444
rsid=2 rtc=1524140117693 seqNo=1 source=internal prevrtc=1524140108693

<134>May 16 15:08:45 myhost2 CEF:0|nCipher Security|nShield Solo XC|12.60.2|3|ssign|5|esn=1111-2222-4444 rsid=2
rtc=1524140117693 gbc=0 fmn=1 hcnt=1
hb=ntwtggjmPYA1TR07KhdOHoyytxLb7RDvg7Wpw6FfAiC4=sign=MEYCIQDxIIJZrFKsXpMMoQ3GDEkTZ/+DTuEdNLKwHQzllfLMUQIhAPipdSPrB
SUnarrtjMslYS4k3RPCXcNo016xEhg/907z
```

20.7.9. Tracing Key Usage

With the information logged as detailed in the preceding sections it is possible to trace back from a Key Command to the loading of the Key, then to loading the Logical Token and reading the Shares that constitute the Logical Token.

The following example shows the notional traceback from a **Cmd_Encrypt** operation. This command logs the nCore Key Hash **KKKKKKKK**. Prior to this the Key was loaded onto the HSM using **Cmd_LoadBlob** which correlates the nCore Key Hash with the ncore Hash of the Logical Token that authorized loading the Key. Tracing further back we can identify the shares used to reconstruct that Logical Token. In this example two shares are required identified by share indices S1 and S2. The share index identifies a specific card in an OCS card-set.

Command	Key Hash	Logical Token Hash	Share Index
Cmd_Encrypt	KKKKKKKK		
Cmd_LoadBlob	KKKKKKKK	LLLLLLL	
Cmd_Read Share		LLLLLLL	S2
Cmd_Read Share		LLLLLLL	S1
Cmd_LoadLogicalToken		LLLLLLL	

20.8. Audit Log Verification

The audit logs produced when AuditLogging feature is active can be verified using the information contained in the audit logging metadata. Every HSM enrolled into a Security World with AuditLogging enabled generates an HSM-specific log signing private key (KAL) that is maintained in the HSM's non-volatile memory until the module is re-initialized. The public key corresponding to this private key is sent as a Certifier Block by the HSM when Audit Logging is configured either by Security World creation or by indoctrination into an existing Audit Logging Security World. Every Signature Block sent by the HSM is generated using the log signing private key. The Audit Log can be verified as follows:

- Extract the KAL public key from the Certifier block
- Verify the Signature Blocks
- Verify the log message hashes in the Signature Block against hashes of the received logs to determine if any messages have been tampered
- Identify any missing log messages.

The basics of the verification approach is shown on the [Audit Log Verification diagram](#).

To support Audit Log verification, Entrust provide an example verification program written in Python to serve as an example for developing a more comprehensive verification solution.

20.8.1. Running the example verification program

The example verification program can be found in the following location:

```
/opt/nfast/python/examples/audit-log-verifier.py
```

This program requires the use of the nShield Python interpreter. This is necessary to provide support for the nShield specific marshalling functions used to export the log signing public

key. The example verification program also requires the presence of an nShield HSM accessible to the machine on which the verification is to be performed. This is required to perform the cryptographic operations necessary to verify the log signing public key and the Signature Blocks. This HSM does not need to be the same HSM on which the logs were generated, nor does it need to be in a Security World.

The Audit Log verifier program is run with a command of the form:

```
python audit-log-verifier.py [-h] [-e ESN] SYSLOG
```

Where:

Parameter	Function
<code>-h, --help</code>	Displays the help message
<code>-e ESN, --esn ESN</code>	ESN of the logevents to be verified
<code>SYSLOG</code>	Location of the syslog file to be verified



Make sure that you use the nShield Python.

20.8.1.1. Results

Results from the Audit Log Verifier are written to several different files and saved in a sub-directory called LogResult. See example below for more detail.

20.8.1.2. Example

Running a command of the form:

```
> python audit-log-verifier.py AuditLogInputFile.txt
```

Should produce a screen output similar to the following:

```
FIRST LOG INSTANCE-1 for ESN:9204-02E0-D947 @ Line:1 rsid:8 #####

Verifying certifier block...
Verification of CERTIFICATE Success
Verifying a cert fragment...Line:1
Verifying a cert fragment...Line:2
Verifying a cert fragment...Line:3
Verifying a cert fragment...Line:4
Verifying a cert fragment...Line:5
Verifying SB...Line:7:InstanceNo:1
Verifying SB...Line:18:InstanceNo:1
Valid Hash list from SBs written to ValidHashes_fromSB_forInst1.txt
No entry in SB for event @ Line:19 : Seq No:12 --
```

```
No entry in SB for event @ Line:20 : Seq No:13 --

Verified cert blocks written to ./LogResult/CBs.txt
Sig blocks written to ./LogResult/SBs.txt
Log messages written to ./LogResult/AllEvents.txt
Anything that did not match (incl.
invalid cert blockfragments) written to ./LogResult/Inconsistent.txt
```

The screen output indicates the contents of the main results files which are stored in the *LogResult* sub-directory. The contents of the folder will vary slightly depending on the log contents and whether there were any failures, but should be similar to the following:

AllEvents.txt	# log entries relating to events
CBs.txt	# certificate blocks
Inconsistent.txt	# inconsistent log entries - should be empty [] (assuming no inconsistencies)
Instance.txt	# esn number and other info relating to the log
InvalidHashes_fromSB_forInst1 block (forInst1 refers to	# only exists if verification failed due to signature first logging instance/world found in the log file)
SBs.txt	# signature blocks
Tampered_logs.txt	# contains log messages that did not verify - e.g. due to a corrupt signature block.
This file only exists	if verification failed.
Unverified_logs.txt	# unverified log entries - e.g. any trailing entries from the end of the log file that lack an accompanying signature block
ValidHashes_fromSB_forInst1.txt	# valid hashes from the signature blocks (forInst1 refers to first logging instance/world found in the log file)
Verified_logs.txt	# verified log messages

Use a text editor to examine the files as required to check the verification. Note that *Inst1* in the filenames refers to the first **logging world** instance in the log, see [Program Architecture](#). If the log contains messages relating to more than one **logging world**, files relating to subsequent instances will be tagged with *Inst2*, *Inst3* etc.

If the verification fails, screen output should indicate the source of the failure. For example, output for a log where a log message was missing would look something like this:

```
FIRST LOG INSTANCE-1 for ESN:9204-02E0-D947 @ Line:1 rsid:8 #####

Verifying certifier block...
Verification of CERTIFICATE Success
Verifying a cert fragment...Line:1
Verifying a cert fragment...Line:2
Verifying a cert fragment...Line:3
Verifying a cert fragment...Line:4
Verifying a cert fragment...Line:5
Verifying SB....Lineno:7:InstanceNo:1
Verifying SB....Lineno:17:InstanceNo:1
Verifying SB....Lineno:28:InstanceNo:1
Valid Hash list from SBs written to ValidHashes_fromSB_forInst1.txt
No entry in SB for event @ Line:29 : Seq No:22 --
No entry in SB for event @ Line:30 : Seq No:23 --
```

```

@@@@@ Some Log events present in the SB but missing in Log file # indicates
missing log message
Verified cert blocks written to ./LogResult/CBs.txt
Sig blocks written to ./LogResult/SBs.txt
Log messages written to ./LogResult/AllEvents.txt
Anything that did not match (incl.
invalid cert blockfragments) written to ./LogResult/Inconsistent.txt

```

Output for a log where a log message had been tampered with or is otherwise corrupt might look like this:

```

FIRST LOG INSTANCE-1 for ESN:9204-02E0-D947 @ Line:1 rsid:8 #####

Verifying certifier block...
Verification of CERTIFICATE Success
Verifying a cert fragment...Line:1
Verifying a cert fragment...Line:2
Verifying a cert fragment...Line:3
Verifying a cert fragment...Line:4
Verifying a cert fragment...Line:5
Verifying SB....Lineno:7:InstanceNo:1
Verifying SB....Lineno:18:InstanceNo:1
Verifying SB....Lineno:29:InstanceNo:1
Valid Hash list from SBs written to ValidHashes_fromSB_forInst1.txt
Validating Log @ Line No:10 SeqNo:4 is Failed ---- # indicates
tampered log entry
***** Hash Mismatch No entry in SB for event @ Line:30 : Seq No:22 --
No entry in SB for event @ Line:31 : Seq No:23 --
Verified cert blocks written to ./LogResult/CBs.txt
Sig blocks written to ./LogResult/SBs.txt
Log messages written to ./LogResult/AllEvents.txt
Anything that did not match (incl.
invalid cert blockfragments) written to ./LogResult/Inconsistent.txt

```

The tampered log line(s) will be listed in output file **Tampered_logs.txt**.

Output for a log where the signature block is corrupt will look something like this:

```

FIRST LOG INSTANCE-1 for ESN:9204-02E0-D947 @ Line:1 rsid:8 #####

Verifying certifier block...
Verification of CERTIFICATE Success
Verifying a cert fragment...Line:1
Verifying a cert fragment...Line:2
Verifying a cert fragment...Line:3
Verifying a cert fragment...Line:4
Verifying a cert fragment...Line:5
Verifying SB....Lineno:7:InstanceNo:1
Verifying SB....Lineno:18:InstanceNo:1
Signature Tampered B64 decode

//k=&#s0Q61C08QB34gaTU2+rUzp/dwtAXi9Hv0IjDvDL/yg=&#Im5nW+OX0gbd1LnRFLxsZtR4meDSEXG5JXtkMmltTZU=&#LGAXS1nvGHE1vXhk8R
VT2lCK2NMtXyD90YTecV0aaBk=&#MbJAK706yU2+QyKWmtfnCV01xn/enber8aJK3cZyxLg=&#y2qx5F5Vgm/X/h6ZcZ5i0es7ZAFppM/6ND8nAXzCM/
bY=&#kWjEa6Ic1Jv494A1ZcUgGHJko7AeKvUUqVimhfExioU= Length: 577
Verifying SB....Lineno:29:InstanceNo:1
Valid Hash list from SBs written to ValidHashes_fromSB_forInst1.txt
In-Valid Hash list from SBs written to InvalidHashes_fromSB_forInst1.txt
    Log entry found in Tampered SB. Line no:8 SeqNo:2
    Log entry found in Tampered SB. Line no:9 SeqNo:3
    Log entry found in Tampered SB. Line no:10 SeqNo:4
    Log entry found in Tampered SB. Line no:11 SeqNo:5

```

```

Log entry found in Tampered SB. Line no:12 SeqNo:6
Log entry found in Tampered SB. Line no:13 SeqNo:7
Log entry found in Tampered SB. Line no:14 SeqNo:8
Log entry found in Tampered SB. Line no:15 SeqNo:9
Log entry found in Tampered SB. Line no:16 SeqNo:10
Log entry found in Tampered SB. Line no:17 SeqNo:11
No entry in SB for event @ Line:30 : Seq No:22 --
No entry in SB for event @ Line:31 : Seq No:23 --
Verified cert blocks written to./LogResult/CBs.txt
Sig blocks written to./LogResult/SBs.txt
Log messages written to./LogResult/AllEvents.txt
Anything that did not match (incl.
invalid cert blockfragments) written to :./LogResult/Inconsistent.txt

```

The failed log messages should be reported in **Tampered_logs.txt** in the **LogResult** folder.

If the certificate block is corrupt, output will be similar to that shown below. In this case, the **CBs.txt** file may be empty and the cert block fragments will be written to **Inconsistent.txt**.

```

FIRST LOG INSTANCE-1 for ESN:9204-02E0-D947 @ Line:1 rsid:8 #####

Verifying certifier block...
Verification of CERTIFICATE Success
Verifying a cert fragment...Line:1
Verifying a cert fragment...Line:2
Verifying a cert fragment...Line:3
Signature Tampered B64 decode
('Failed fragment:', 3, '<134>May  2 16:10:38 exampleCB1.myexample.com CEF:0|nCipher Security|nShield Solo
XC|12.60.2|3|ssign-cert|5|esn=9204-02E0-D947 rsid=8rtc=4294967386734000 tpbl=2140 findex=3 flen=450
frag=B1Ku9rirlixkgEd+73tMVJ1FQz85aCWuRqJl04YB1YwFvZgVRXhHvzqLFeJZAuerKLLgIaZwDq1twoXzvHq88QcJdbr0i4+87VorPKkEjKtS
SGHOVkkHhoBC8uNgYXnTBxqcqCqpZl4whuiEBmJQLcwgAAAAg8rgckmo3ArobecQooPxQ9AjYbCmAoKOUTRi7grTzPyAAQAA3Bvuz+tQ1uh5LvuKM
LTtGDTpL67ks6Zkl8b+F2UW37jFN3lap27oAZq1otU4FOP4EVVoMnNSdI4uzCPi7VgcI3AcIkdjZIwbpYf9XQwvFwMxYvdBPGhPtc/t8LsLgs97r
MkES4ZcINI/NwjKp0fw4kCiSBSUQUAUcp6vggq2vVL9naqRHhXNRJuweaRt0060z0mBkTgCnAvscdr2ymErrWDZArHosYXJZrXghjNmXvu+rS86vT
vTc sign=MEYCIQCXhbJefTv8oR8a51aU30s9w2Vs9w67mzYk584Gy+MdbgIhAOD0bAwU0Vw1xOmR2oerqWKLFeawZe5rONmDMZFbJoB')
Fragment verification unsuccessful!
Adding all fragments in this CB to Inconsistent.txt
('No Valid CB for this instance:', 1)
In-Valid Hash list from SBs written to InvalidHashes_fromSB_forInst1.txt
Log entry found in Tampered SB. Line no:6 SeqNo:1
Log entry found in Tampered SB. Line no:8 SeqNo:2
Log entry found in Tampered SB. Line no:9 SeqNo:3
Log entry found in Tampered SB. Line no:10 SeqNo:4
Log entry found in Tampered SB. Line no:11 SeqNo:5
Log entry found in Tampered SB. Line no:12 SeqNo:6
Log entry found in Tampered SB. Line no:13 SeqNo:7
Log entry found in Tampered SB. Line no:14 SeqNo:8
Log entry found in Tampered SB. Line no:15 SeqNo:9
Log entry found in Tampered SB. Line no:16 SeqNo:10
Log entry found in Tampered SB. Line no:17 SeqNo:11
Log entry found in Tampered SB. Line no:19 SeqNo:12
Log entry found in Tampered SB. Line no:20 SeqNo:13
Log entry found in Tampered SB. Line no:21 SeqNo:14
Log entry found in Tampered SB. Line no:22 SeqNo:15
Log entry found in Tampered SB. Line no:23 SeqNo:16
Log entry found in Tampered SB. Line no:24 SeqNo:17
Log entry found in Tampered SB. Line no:25 SeqNo:18
Log entry found in Tampered SB. Line no:26 SeqNo:19
Log entry found in Tampered SB. Line no:27 SeqNo:20
Log entry found in Tampered SB. Line no:28 SeqNo:21
No entry in SB for event @ Line:30 : Seq No:22 --
No entry in SB for event @ Line:31 : Seq No:23 --

```

```
Verified cert blocks written to ./LogResult/CBs.txt
Sig blocks written to ./LogResult/SBs.txt
Log messages written to ./LogResult/AllEvents.txt
Anything that did not match (incl.
invalid cert blockfragments) written to ./LogResult/Inconsistent.txt
```

20.8.2. Program Architecture

The program takes and reads the input syslog file containing the log messages. It optionally sets the ESN of module for which log events are to be validated, if this was passed in. If an ESN is not provided as input then the first ESN found in the syslog will be processed.

The verifier calls its parse function which segregates the messages based on ESN, and creates lists of Certifier fragments, Signature Blocks and Log events, based on matching with regular expressions.

Syslog may have gathered logs from multiple sources. As such, the verifier has a concept of a *logging world*, which represents a set of logs, sigblocks and certblocks that belong together, from a Security World. Based on Reboot Sequence ID, Sequence Number of the Log event, Global block counter of the Signature block and Fragment index of the Certifier block, a logging world is identified and a logging instance is created.

All records are thus given a log-instance number, such that records with the same instance number belong together.

Each event can thus be uniquely identified via a tuple. For the log messages, signature blocks and certifier blocks these are respectively (*rsid* and *sequence number*), (*rsid* and *gbc*) and (*rsid* and *findex*).

The *reconstruct_CBs* function is then called to validate the certifier fragments (using calls to an nShield HSM for crypto functionality). It then reconstructs the certifier blocks from the certifier fragments.



This does not require the HSM to be in the same Security World as the HSM that first generated the logs.

A list of valid and verified Certifier Blocks is created.

For any log instance one valid Certifier Block is enough to validate the events, so further certifier blocks are ignored after the first.

Next the *process_sbs* function is called. Signature Blocks for a supplied ESN are validated per log instance (once again via calls to the module for crypto functionality), using the KAL value taken from the Certifier block previously.

The validated Signature block hashes are maintained as a dictionary of hashes with keys as unique ids. These unique ids per instance are generated based on rsid and sequence numbers.

The `process_logs` function is finally called. This generates the hash of each of the log events and matches against hashes from corresponding signature blocks. Verified and Tampered log events are then written to different files in the *LogResult* folder.

20.8.3. Extended Verification

While the example verifier uses an HSM for cryptographic operation, it would be possible to use 3rd party cryptographic libraries to provide this functionality. This is outside the scope of this document.

Currently the log messages are verified against the hash in the signature blocks, and the signature of the signature blocks is verified against the key extracted from the certifier block. The certifier block itself is not verified. A potential extension to the verifier tool would be to verify the certifier block. The certifier block is signed by KLF2. This can be checked against the KLF2 value found within the module's warrant. This would complete the chain of trust.

Additionally, the example verifier does not cope with fields that rotate back around to zero when their max size is exceeded. (for example, `gbk`, `rsid` or `seqno`). Currently logs, SBs and CBs are uniquely identified by (`rsid` and `sequence number`), (`rsid` and `gbc`) and (`rsid` and `findex`). This means that, if any of those values rotate back around to zero, we are no longer able to uniquely identify them. As a potential extension, RTC or line number values could be used to solve this.

The example verifier does not detect missing/deleted log messages in the case where a complete group of log messages are deleted, along with their corresponding **Signature-Block**. Given that the `SeqNo` field increases for each log message, spotting missing `SeqNos` would reveal missing or deleted log messages. This is a potential extension.

The example verifier expects a static, unchanging log file to be supplied to it. This would be compatible with verifying a batch of log files at the end of each day, for example. A possible extension would be to extend the verifier to cope with a live stream of logs, continuously verifying them as they are generated.

21. Key generation options and parameters

This appendix describes the various options and parameters that you can set when running the **generatekey** utility to control the application type and other properties of a key being generated.



For information about generating keys with the **generatekey** utility, see [Generating keys with the command line](#).

21.1. Key application type (APPNAME)

The **APPNAME** parameter specifies the name of the application for which **generatekey** can generate keys. Specifying an application can restrict your choice of key type. A value for **APPNAME** must follow any **OPTIONS** and must precede any parameters specified for the key:

Parameter	Description
simple	Specifying the simple application type generates an nShield-native key. No special action is taken after the key is generated.
custom	<p>Specifying the custom application type generates a key for custom applications that require the key blob to be saved in a separate file.</p> <p>Specifying custom also causes the generation of a certificate request and self-signed certificate. However, we recommend that you specify the simple (instead of custom) application type whenever possible.</p>
pkcs11	<p>Specifying the pkcs11 application type generates keys that are formatted for use with PKCS #11 applications and are given a suitable identifier. The set of possible supported key types is currently limited to:</p> <ul style="list-style-type: none"> • DES3 • DH • DSA • ECDH • ECDSA • Ed25519 • HMACSHA1 • RSA • Rijndael (AES) • X25519 <p>Some key types are only available if the features that support them have been enabled for the module, if the Security World is not compliant with FIPS 140-2 Level 3, or if you do not set the --no-verify option.</p>

Parameter	Description
embed	<p>Specifying the embed application type generates a key for use with older applications that integrate with the OpenSSL CHIL engine and/or hwcrhk. Note, these interop libraries are no longer provided as of v12.60.</p> <p>You can use a key of the embed application type like a PEM-format RSA/DSA key file, even though it is really a specially encoded reference to a key stored in opt/nfast/kmdata/local. This allows you to use an embed key when integrating with applications that normally require software RSA keys. For example, you can supply an embed key to the patched version of OpenSSL we have provided so that it uses the module to access the key rather than using its own built-in RSA operations.</p>
kpm	Specifying the kpm application type generates a key for delivery by an nForce Ultra key server. The generatekey utility automatically creates a special ACL entry that permits a kpm to be delivered to an nForce Ultra's enrolled internal hardware security module.
seeinteg	Specifying the seeinteg application type generates an SEE integrity key. The DSA, RSA, ECDSA and KCDSA algorithms are supported. SEE integrity keys are always protected by an OCS and cannot be imported. You cannot retarget an existing key as an SEE integrity key.
seeconf	Specifying the seeconf application type generates an SEE confidentiality key. Both the Triple DES and AES algorithms are supported for this key type. SEE confidentiality keys are module-protected by default and cannot be imported. You cannot retarget an existing key as an SEE confidentiality key.

21.2. Key properties (NAME=VALUE)

The **NAME=VALUE** syntax is used to specify the properties of the key being generated.




If a parameter's argument contains spaces, you must enclose the argument within quotation marks (" ").

You can supply an appropriate **VALUE** for the following **NAME** options:

Option	Description
alias	The VALUE for alias specifies an alias to assign to the key.
assigned	The VALUE for assigned specifies if the generated key is to be Assigned as defined by <i>nShield Solo XC Common Criteria Evaluated Configuration Guide</i> . This is only relevant in common-criteria-cmts mode Security Worlds and the key must be protected with a non-recoverable softcard or token. If set to yes the ACL of the generated key will match the definition of an Assigned key in <i>nShield Solo XC Common Criteria Evaluated Configuration Guide</i> and will be verified as an Assigned key by nfkverify . The default is no .

Option	Description
<code>blobsavefile</code>	When using the <code>custom</code> application type, the <i>VALUE</i> for <code>blobsavefile</code> specifies a file name of the form <code>FILENAME_req.ext</code> to which the key blob is saved. Additionally, a text file containing information about the key is saved to a file whose name has the form <code>ROOT_inf.txt</code> ; for asymmetric key types, the public key blob is also saved to a file whose name has the form <code>ROOT_pub.EXT</code> .
<code>cardset</code>	The <i>VALUE</i> for <code>cardset</code> specifies an OCS that is to protect the key (if <code>protect</code> is set to <code>token</code>). In interactive mode, if you do not specify an OCS, you are prompted to select one at card-loading time. The default is the OCS to which the card currently inserted in the slot belongs (or the first one returned by <code>nfkminfo</code>).
<code>certreq</code>	<p>Setting <code>certreq</code> enables you to generate a certificate request when generating a PKCS #11 key (RSA keys only). The default behavior is to not generate a certificate request.</p> <p>To generate a certificate request you must set the <i>VALUE</i> for <code>certreq</code> to <code>yes</code>, which makes <code>generatekey</code> prompt you to fill in the extra fields required to generate a key with a certificate request. The resultant certificate request is saved to the current working directory with a file name of the form <code>FILENAME_req.ext</code> (where <code>FILENAME</code> is a name of your choice).</p> <p>An extra file with a name of the form <code>FILENAME.ext</code> is also generated for use as a pseudo-key-header. This file can be removed after the certificate request has been generated. You can use <code>certreq</code> with the <code>--retarget</code> option to generate a self-signed certificate for an existing key.</p>
<code>checks</code>	For RSA key generation only, this specifies the number of checks to be performed. Normally, you should leave <i>VALUE</i> empty to let the module pick an appropriate default.
<code>curve</code>	For ECDH and ECDSA key generation only, the <i>VALUE</i> for <code>curve</code> specifies which curves from the supported range to use. Supported curves are: ANSI-B163v1, ANSIB191v1, BrainpoolP160r1, BrainpoolP160t1, BrainpoolP192r1, BrainpoolP192t1, BrainpoolP224r1, BrainpoolP224t1, BrainpoolP256r1, BrainpoolP256t1, BrainpoolP320r1, BrainpoolP320t1, BrainpoolP384r1, BrainpoolP384t1, BrainpoolP512r1, BrainpoolP512t1, NISTP192, NISTP224, NISTP256, NISTP384, NISTP521, NISTB163, NISTB233, NISTB283, NISTB409, NISTB571, NISTK163, NISTK233, NISTK283, NISTK409, NISTK571, SECP160r1 and SECP256k1
<code>embedconvfile</code>	The <i>VALUE</i> for <code>embedconvfile</code> specifies the name of the PEM file that contains the RSA key to be converted.

Option	Description
<code>embedsavefile</code>	<p>When using the <code>embed</code> application type, the <code>VALUE</code> for <code>embedsavefile</code> specifies the name for the file where the fake RSA private key is to be saved. The file has the same syntax as an RSA private key file, but actually contains the key identifier rather than the key itself, which remains protected.</p> <p>A certificate request and a self-signed certificate are also written. If the file-name is <code>ROOT.EXT</code> then the request is saved to <code>ROOT_req.EXT</code> and the self-signed certificate is saved to <code>ROOT_selfcert.EXT</code>.</p>
<code>from-application</code>	When retargeting a key, the <code>VALUE</code> for <code>from-application</code> specifies the application name of the key to be retargeted. Only applications for which at least one key exists are acceptable.
<code>from-ident</code>	When retargeting a key, the <code>VALUE</code> for <code>from-ident</code> specifies the identifier of the key to be retargeted (as displayed by the <code>nfkminfo</code> command-line utility).
<code>hexdata</code>	The <code>VALUE</code> for <code>hexdata</code> specifies the hex value of DES or Triple DES key to import. The hex digits are echoed to the screen and can appear in process listings if this parameter is specified in the command line.
<code>ident</code>	The <code>VALUE</code> for <code>ident</code> specifies a unique identifier for the key in the Security World. For applications of types <code>simple</code> , this is the key identifier to use. For other application types, keys are assigned an automatically generated identifier and accessed by means of some application-specific name.
<code>keystore</code>	The <code>VALUE</code> for <code>keystore</code> specifies the file name of the key store to use. This must be an nShield key store.
<code>keystorepass</code>	The <code>VALUE</code> for <code>keystorepass</code> specifies the password to the key store to use.
<code>logkeyusage</code>	The <code>VALUE</code> for <code>logkeyusage</code> specifies if usage of the generated key in cryptographic operations is subject to audit logging. If set to yes the ACL of the generated key will predicate audit-logging entries to be made for cryptographic usages of the key. The default is no .
<code>module</code>	<p>The <code>VALUE</code> for <code>module</code> specifies a module to use when generating the key. If there is more than one usable module, you are prompted to supply a value for one of them. The default is the first usable module (one in the current Security World and in the operational state).</p> <div>  <p>You can also specify a module by setting the <code>--module</code> option.</p> </div>
<code>paramsreadfile</code>	The <code>VALUE</code> for <code>paramsreadfile</code> specifies the name of the group parameters file that contains the discrete log group parameters for Diffie-Hellman keys only. This should be a PEM-formatted PKCS#3 file. If a <code>VALUE</code> for <code>paramsreadfile</code> is not specified, the module uses a default file.

Option	Description
<code>pemreadfile</code>	The <i>VALUE</i> for <code>pemreadfile</code> specifies the name of the PEM file that contains the key to be imported. When importing an RSA key, this is the name of the PEM-encoded PKCS #1 file to read it from. Password-protected PEM files are not supported.
<code>plainname</code>	The <i>VALUE</i> for <code>plainname</code> specifies the key name within the Security World. For some applications, the key identifier is derived from the name, but for others the name is just recorded in <i>kmdata</i> and not used otherwise.
<code>protect</code>	The <i>VALUE</i> for <code>protect</code> specifies the protection method, which can be <code>module</code> for security-world protection, <code>softcard</code> for softcard protection or <code>token</code> for Operator Card Set protection. The default is <code>token</code> , except for <code>seeconf</code> keys, where the default is <code>module</code> . <code>seeinteg</code> keys are always token-protected. The <code>softcard</code> option is only available when your system has at least one softcard present.
<code>pubexp</code>	For RSA key generation only, the <i>VALUE</i> for <code>pubexp</code> specifies (in hexadecimal format) the public exponent to use when generating RSA keys. We recommend leaving this parameter blank unless advised to supply a particular value by Support.
<code>recovery</code>	The <i>VALUE</i> for <code>recovery</code> enables recovery for this key and is only available for card-set protected keys in a recovery-enabled world. If set to <code>yes</code> , the key is recoverable. If set to <code>no</code> , key is not recoverable. The default is <code>yes</code> . Non-recoverable module-protected keys are not supported.
<code>seeintegname</code>	If present, the <i>VALUE</i> for <code>seeintegname</code> identifies a <code>seeinteg</code> key. The ACL of the newly generated private key is modified to require a certificate from the <code>seeinteg</code> key for its main operational permissions, such <code>Decrypt</code> and <code>Sign</code> (<code>DuplicateHandle</code> , <code>ReduceACL</code> , and <code>GetACL</code> are still permitted without certification.)
<code>selfcert</code>	The <i>VALUE</i> for <code>selfcert</code> enables you to generate a self-signed certificate when generating a PKCS #11 key (RSA keys only). To generate a self-signed certificate request you must set <code>selfcert</code> to <code>yes</code> , which makes <code>generatekey</code> prompt you to fill in the extra fields required to generate a key with a self-signed certificate. The resultant certificate is saved to the current working directory with a file name of the form <i>FILENAME.ext</i> . You can use this parameter with the <code>--retarget</code> option to generated a self-signed certificate for an existing key.
<code>size</code>	For key types with variable-sized keys, the <i>VALUE</i> for <code>size</code> specifies the key size in bits. The range of allowable sizes depends on the key type and whether the <code>--no-verify</code> option is used. The default depends on the key type; for information on available key types and sizes, see Cryptographic algorithms . This parameter does not exist for fixed-size keys, nor for ECDH and ECDSA keys which are specified using <code>curve</code> .

Option	Description
<code>strict</code>	For DSA key generation only, setting the <i>VALUE</i> for <code>strict</code> to <code>yes</code> enables strict verification, which also limits the size to 2048 or 3072 bits. The default is <code>no</code> .
<code>type</code>	The <i>VALUE</i> for <code>type</code> specifies the type of key. You must usually specify the key type for generation and import (though some applications only support one key type, in which case you are not asked to choose). Sometimes the type must also be specified for retargeting; for information on available key types and sizes, see Cryptographic algorithms . The <code>--verify</code> option limits the available key types.
<code>x509country</code>	The <i>VALUE</i> for <code>x509country</code> specifies a country code, which must be a valid 2-letter code, for the certificate request.
<code>x509dnscommon</code>	The <i>VALUE</i> for <code>x509dnscommon</code> specifies a site domain name, which can be any valid domain name, for the certificate request.
<code>x509email</code>	The <i>VALUE</i> for <code>x509email</code> specifies an email address for the certificate request.
<code>x509locality</code>	The <i>VALUE</i> for <code>x509locality</code> specifies a city or locality for the certificate request.
<code>x509org</code>	The <i>VALUE</i> for <code>x509org</code> specifies an organization for the certificate request.
<code>x509orgunit</code>	The <i>VALUE</i> for <code>x509orgunit</code> specifies an organizational unit for the certificate request.
<code>x509province</code>	The <i>VALUE</i> for <code>x509province</code> specifies a province for the certificate request.
<code>xsize</code>	The <i>VALUE</i> for <code>xsize</code> specifies the private key size in bits when generating Diffie-Hellman keys. The defaults are 256 bits for a key size of 1500 bits or more or 160 bits for other key sizes.

21.3. Available key properties by action/application

The following table shows which actions (generate, import, and retarget) are applicable to the different *NAME* options:

Property	generate	import	retarget
<code>alias</code>	X	X	X
<code>blobsavefile</code>	X	X	X
<code>cardset</code>	X	X	
<code>certreq</code>			
<code>checks</code>	X		

Property	generate	import	retarget
curve	X		
embedconvfile		X	
embedsavefile	X	X	X
from-application			X
from-ident			X
hexdata		X	
ident	X	X	
keystore	X	X	X
keystorepass	X	X	X
module	X	X	
nvrnm	X	X	
paramsreadfile	X		
pemreadfile		X	
plainname	X	X	X
protect	X	X	
pubexp	X		
qsize	X		
recovery	X	X	
seeintegname			
selfcert			
size	X		
strict	X		
type	X		
x509country	X	X	X
x509dnscommon	X	X	X
x509email	X	X	X
x509locality	X	X	X
x509org	X	X	X
x509orgunit	X	X	X

Property	generate	import	retarget
x509province	X	X	X
xsize	X		

The following table shows which applications are applicable to the different *NAME* options:

Property	custom	embed	hwcrhk	pkcs 11	seeconf	seeinteg	seessl	simple	kpm
alias									
blobsavefile	X								
cardset	X	X	X	X				X	X
certreq				X					
checks	X	X	X	X				X	X
curve	X	X	X	X	X	X		X	
embedconvfile		X							
embedsavefile		X		X					
from-application	X	X	X	X				X	X
from-ident	X	X	X	X				X	X
hexdata	X	X	X	X				X	
ident			X					X	X
keystore									
keystorepass									
module	X	X	X	X			X	X	X
nvram	X	X	X	X				X	
paramsreadfile	X	X	X	X	X	X		X	
pemreadfile	X		X					X	X
plainname	X	X		X	X	X	X	X	X
protect	X	X	X	X	X	X	X	X	X
pubexp	X	X	X	X				X	X
qsize	X	X	X	X				X	X
recovery	X	X	X	X	X	X		X	X
seeintegname	X							X	

Property	custom	embed	hwcrhk	pkcs 11	seeconf	seeinteg	seessl	simple	kpm
selfcert				X					
size	X	X	X	X	X	X	X	X	X
strict	X	X	X	X				X	
type	X	X	X	X	X	X	X	X	X
x509country		X							X
x509dnscommon		X							X
x509email		X							X
x509locality		X							X
x509org		X							X
x509orgunit		X							X
x509province		X							X
xsize	X	X	X	X				X	

22. Checking and changing the mode on an nShield Connect

This appendix tells you how to check and change the mode on an nShield Connect. You must change the mode to perform certain configuration tasks.

22.1. nShield Connect front panel controls

See [The nShield Connect user interface](#) for a description of the nShield Connect user interface, including the front panel controls.



We recommend that you use a keyboard to manage the front panel menu options and enter text. See [Using a keyboard to control the unit](#) for more information.

22.2. Available modes

The following modes are available:

- **Operational**
 - The default setting for day-to-day use
- **Initialization**
 - Sets the nShield Connect to start in pre-initialization mode
 - Allows you to use the nShield Connect to create a Security World or add the module to an existing one



You cannot select **Maintenance** mode. It is managed by the nShield Connect and cannot be set by a user.

22.3. Identifying the current mode

You can check the current mode of an nShield Connect:

- At the nShield Connect itself
- By using the **enquiry** command-line utility from a client computer
- By using KeySafe from a client computer

22.3.1. Checking the mode at the nShield Connect

22.3.1.1. The status LED

The nShield Connect Status LED indicates the operational status of the module.

Status LED	Description
On, occasionally blinks off.	Status: Operational mode The module is in Operational mode and accepting commands. The more frequently the Status LED blinks off, the greater the load on the module.
Flashes two short pulses, followed by a short pause.	Status: Initialization mode Existing Security World data on the module has been erased. The module is automatically placed in Initialization mode after a Security World is created. For more information, see the <i>nShield Connect User Guide</i> .
Flashes two long pulses followed by a pause.	Status: Maintenance mode Used for reprogramming the module with new firmware. The module only goes into Maintenance mode during a software upgrade.

22.3.1.2. The front panel display screen

The nShield Connect screen shows a color-coded footer at the bottom of the display when it is not in Operational mode.

Footer color	Text in footer	Meaning
Yellow	Initialization	The system is rebooting or waiting for an Administrator Card to be inserted.
Blue	Maintenance	An administrative task is being performed. This mode is only entered during firmware upgrades.
Red	HSM Failed	The internal module has failed.

22.3.2. Checking the mode using enquiry

You can use the **enquiry** command-line utility to display information about the hardware and the status of the nShield Connect. The **enquiry** utility is in the **bin** subdirectory of the **nCipher** directory. This is usually **/opt/nfast**.

To check the mode using **enquiry**:

1. Log in on the client computer as a user, and open a command window.

2. Run the command:

```
opt/nfast/bin/enquiry
```

The following is an example of the **enquiry** command output:

```
Server:
enquiry reply flags      none
enquiry reply level     Six
serial number           #####-####-####-####
mode                    operational
version                 #.#.#
speed index             ###
rec. queue              ##.##
...
version serial          #
remote port (IPv4)      #####

Module #1:
enquiry reply flags      none
enquiry reply level     Six
serial number           #####-####-####-####
mode                    operational
version                 #.#.#
speed index             ###
rec. queue              ##.##
...
rec. LongJobs queue     ##
SEE machine type        PowerPCSXF
```

In this example, the **mode** line shows that the nShield Connect is in **operational** mode.

22.3.3. Checking the mode by using KeySafe

You can use the **Module Status tree** of the KeySafe GUI to identify the current mode of the nShield Connect.

To check the mode using KeySafe:

1. Start KeySafe on a client computer.
2. Locate the **Module Status tree** (part of the **Security World status** panel) positioned to the bottom left of the KeySafe window.
3. Expand the **Security World** and/or **Outside Security World** nodes as required.
4. Locate the appropriate nShield Connect (**Module**).

The current mode of the module is displayed in the **State** field.

See [Using KeySafe](#) for more about using KeySafe. See [Module information](#) for more about checking the mode.

22.4. Changing the mode

You can change the mode of an nShield Connect using:

- The front panel controls of the nShield Connect
- The `nopclearfail` command-line utility from a client computer

22.4.1. Changing the mode using the front panel controls of an nShield Connect

To change the mode of an nShield Connect, use the front panel menu screens and dialogs to do the following:

1. Navigate to **HSM > Set HSM mode**.
2. Select **Initialisation** or **Operational** as required.

22.4.2. Changing the mode using remote mode and nopclearfail

You can enable or disable the ability to make remote mode changes, see [Enabling and disabling remote mode changes](#)

Once you have enabled remote mode changes, you can change the mode of an nShield Connect from a computer using the `nopclearfail` command, without accessing the unit itself.

22.4.2.1. Available commands

You can use the following commands to change the mode of a module:

Command	Resulting mode
<code>nopclearfail --operational -O</code>	Operational
<code>nopclearfail --initialization -I</code>	Pre-initialization

To change the mode, do the following:

1. Run either:
 - a. The `nopclearfail --operational | -O` command.
or:
 - b. The `nopclearfail --initialization | -I` command.
When finished, the system responds with **OK**.



The system responds with **OK**, regardless of whether the mode of the nShield Connect has changed or not. To confirm that state of the module, do the following:

2. Run the **enquiry** command.

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

23. Upgrading the nShield Connect image file and associated firmware

The nShield image file package for the nShield Connect is a single software bundle that consists of the following components:

- The internal module firmware
- The unit operating system image and host side software.

The image file package is installed on the remote file system when you install the client software. The following sections describe how to load this firmware package onto your nShield module.

23.1. Version Security Number (VSN)

Each nShield image file has a Version Security Number (VSN). In addition, the module firmware has its own individual VSN. This number is increased whenever we improve the security of the image file and/or firmware.

We supply several versions of the module firmware. You can always upgrade to firmware with an equal or higher VSN than that currently installed on your module.



You can never load an image file with a lower VSN than the currently installed version.



You can never load module firmware with a lower VSN than the currently installed firmware.

Ensuring you use an image file package with the highest available VSN allows you to benefit from security improvements and enhanced functionality. It also prevents future downgrades of the image file and/or that could potentially weaken security. However, you may choose to install an image file or associated firmware that does not have the highest available VSN. For example, if you have a regulatory requirement to use FIPS-approved firmware, you should install the latest available FIPS-validated firmware package, which may not have the highest VSN. Similarly, if you want to install a version with enhanced features without committing yourself to the upgrade, you can do so providing you upgrade only to firmware with a VSN equal to that currently installed on your module.

23.2. Key data

Security Worlds and key data are preserved on the RFS host computer only when you upgrade the unit image and/or firmware. You must restore the unit to the Security World if you wish to continue using the key data.



Adding or restoring a module will require authorisation from a quorum of Administrator cards.



When upgrading the Connect image file, client licenses and features activations on the nShield Connect will persist unless you deliberately factory state the unit (from the front panel menu).

For more information, see [Adding or restoring an HSM to the Security World](#).

23.3. Upgrading the nShield Connect image file and firmware using the front panel

Before you upgrade the image file or firmware, ensure that you have installed the latest Security World Software on the client computer and that the remote file system has an up-to-date set of Security World files.

The nShield Connect image file contains both the Connect image and the firmware.

To upgrade the nShield Connect image file and firmware:

1. Ensure that the nShield Connect image file that you require has been copied to the following directory:
 - Windows: `%NFAST_HOME%\nethsm-firmware\<version>`.
 - Linux: `/opt/nfast/nethsm-firmware/<version>`.

Where `<version>` is a subfolder containing a firmware image for the respective version. There can be more than one `<version>` subfolder. The string `<version>` must match the name of the version folder in which the image is located on the version's firmware ISO.

2. From the main menu on the unit, select **System > Upgrade system**.
3. Confirm that you want to upgrade the nShield Connect image file.
4. Select the directory that contains the image file or firmware that you require. You are informed that the files are being transferred.
5. Verify the image version, HSM (firmware) version, and image VSN that are displayed, and confirm the upgrade when prompted.

23.4. Remotely enabling dynamic feature certificates including nShield Connect client licenses

Feature certificates contained on the remote file system (RFS) can be applied to the nShield Connect. The main use case for applying feature certificates is for enabling the client licenses dynamic feature which have been purchased after the initial nShield Connect purchase, although both static and other dynamic feature certificates can be applied.

To apply a dynamic feature certificate, e.g. nShield Connect client license, do the following:



Feature certificates must be present on the RFS in the folder `$NFAST_KM DATA/hsm-ESN/features`.

1. Use the `nethsmadmin` utility to list the nShield Connect feature files on the RFS. Run the command:

```
nethsmadmin --module=<MODULE> --rfs=<RFS_IP> --list-features
```

In this command:

- `<MODULE>` specifies the HSM to use, by its ModuleID (default = 1).
 - `<RFS_IP>` specifies the IP address of the RFS.
2. Use the `nethsmadmin` utility to make the nShield Connect use a specific feature file from the RFS. Run the command:

```
nethsmadmin --module=<MODULE> --rfs=<RFS_IP> --apply-feature=<feature_file>
```

In this command:

- `<feature_file>` must be the path to the feature file that is displayed when you run the `nethsmadmin` command with the `--list-features` option. Errors are reported if you use either just the feature name, or the full path. The file must be alphanumeric, and no longer than 150 characters.

The following is an example of the output expected when applying a dynamic feature:

```
Applying feature <DYNAMIC_FEATURE> to module <MODULE_NO> ...
Feature <DYNAMIC_FEATURE> application process started on module <MODULE_NO>
*DYNAMIC_FEATURE DETECTED*
Please restart you clientside hardserver and check the enquiry output to ensure the dynamic feature
has been applied correctly!
For the client licences feature check the 'max exported modules' section in enquiry to see if the new
client number has been applied correctly.
```

The following is an example of the output expected when applying a static feature:

```
Applying feature <STATIC_FEATURE> to module <MODULE_NO> ...
Feature <STATIC_FEATURE> application process started on module <MODULE_NO>
*STATIC_FEATURE DETECTED*
To be able to use the static feature please clear module MODULE_NO.
Use the fet utility to verify the feature was applied correctly.
```

In the output examples:

- **<DYNAMIC_FEATURE>** specifies the name of the dynamic feature file applied.
- **<STATIC_FEATURE>** specifies the name of the static feature file applied.
- **<MODULE_NO>** specifies the HSM that the feature was applied to.

23.5. Upgrading the nShield Connect image file and firmware from a privileged client

Before you upgrade the nShield Connect firmware, ensure you have installed the latest Security World software on the RFS and Client. However, an actual Security World is not required for this operation.

The following description assumes the RFS and Client are separate machines which an nShield Connect has already been configured to use. If you are using a combined RFS/Client, then apply the following instructions to the same machine. The Client must have privileged access to the nShield Connect.

The image upgrade file may be supplied as a separate item that must be copied into the sub folder for its respective version. The file must always be named **nCx3N.nff** irrespective of its version.

1. Ensure that the new nShield Connect image file is in the following folder on the RFS:

- Windows: **%NFAST_HOME%\nethsm-firmware\<version>**
- Linux: **/opt/nfast/nethsm-firmware/<version>**

Where **<version>** is a subfolder containing the nShield Connect image for the respective version. There can be more than one **<version>** subfolder. The string **<version>** must match the name of the version folder in which the image is located on the version's firmware ISO.

If the **<version>** subfolder does not already exist on the RFS, it must be created by a user with the necessary privileges.

2. List the image file(s) available on the RFS, run the following command from the Client:

```
Client:>>nethsmadmin -m<n> -s <RFS_IP> -l
```

Where:

- **<n>** is the module number for the target nShield Connect
- **<RFS_IP>** is the IP address of the RFS.

For example, when the image file is located in the appropriately named **<version>** folder:

```
client:>>nethsmadmin -m2 -s 194.28.158.146 -l
Initiating RFS nethsm image check on 194.28.158.146...

Checking the nethsm-firmware directory on the RFS.
nethsm-firmware/VersionName/nCx3N.nff
nethsm-firmware/AnotherVersionName/nCx3N.nff

Images were successfully found on the RFS (194.28.158.146).
```

For example, if the version folder does not exist or its name is not correct, the **nethsmadmin** command cannot find the image:

```
client:>>nethsmadmin -m2 -s 194.28.158.146 -l
Initiating RFS nethsm image check on 194.28.158.146...

Checking the nethsm-firmware directory on the RFS.
No images found on the RFS (194.28.158.146).
```

3. In order to load (or upgrade) the firmware image onto the nShield Connect, run the following command from the Client:

```
Client:>>nethsmadmin -m<n> --upgrade-image=nethsm-firmware/<selected-image-version>/nCx3N.nff
```

Where:

- **<n>** is the module number for the target nShield Connect
- **<selected-image-version>** specifies the version subfolder on the RFS containing the firmware image you wish to load (upgrade) onto the nShield Connect.



Copy the path to the required image file as provided by the available image list above. (Linux style path separators are used irrespective of whether the Client or RFS are Windows or Linux based).

e.g.

```
client:>nethsmadmin -m2 --upgrade-image=nethsm-firmware/VersionName/nCx3N.nff
Initiating appliance image upgrade using file nethsm-firmware/VersionName/nCx3N.nff...
Upgrade operation state changed to: Image Transfer Initiated
Upgrade operation state changed to: Image Transferred
Upgrade operation state changed to: Image Verified
Not able to contact appliance because of reason(23): CrossModule,#1-ExplicitRequest,#2-Mode
Upgrade operation final state: Image Verified
Image upgrade completed.
Please wait for appliance to reboot.
Please wait for approximately half an hour for the appliance to internally upgrade.
```

The following line is expected and requires no action:

```
Not able to contact appliance because of reason(23): CrossModule,#1-ExplicitRequest,#2-Mode
```



If the nShield Connect suffers a loss of power while you are upgrading the image file or internal module firmware, exit the **nethsmadmin** utility, wait until power is restored to the HSM, then try to restart the process as shown above.

4. After the image upgrade has completed, run the **enquiry** utility to check the image version of the target nShield Connect is as expected.

23.5.1. Enabling and disabling remote upgrade

You can enable or disable the ability to remotely upgrade an nShield Connect, see [Enabling and disabling remote mode changes](#).

Once you have enabled remote upgrade, you can upgrade an nShield Connect from a computer using the **nethsmadmin** command, without accessing the unit itself.

23.6. After firmware installation

After you have installed new firmware and initialized the HSM, you can create a new Security World with the HSM or reinitialize the HSM into an existing Security World.

If you are initializing the HSM into a new Security World, see [Creating a Security World](#).

If you are re-initializing the HSM into an existing Security World, see [Adding or restoring an HSM to the Security World](#).

24. SNMP monitoring agent

This appendix describes the Simple Network Management Protocol (SNMP) monitoring agent. The SNMP monitoring agent provides you with components that you can add to your (third-party) SNMP manager application.

SNMP was developed in 1988 and revised in 1996. It is currently regarded as the standard method of network management. It is widely supported and offers greater interoperability than traditional network management tools (for example, `rsh` or `netstat`). This makes it ideal for use for the large array of platforms that we support and also avoids the overhead of remote login and execution, helping to reduce network congestion and improve performance.

SNMP defines a collection of network management functions allowing management stations to gather information from, and transmit commands to, remote machines on the network. Agents running on the remote machines can take information gathered from the system and relay this information to the manager application. Such information is either requested from the underlying operating system or gained by interrogating the hardware.



Every SNMP manager adds monitor components differently. Consult the documentation supplied with your SNMP Manager application for details on how to add the MIB files.

SNMP defines the following SNMP messages:

Message	Description
<code>get</code>	This message is sent by a manager to retrieve the value of an object at the agent.
<code>set</code>	This message is sent by a manager to set the value of an object at the agent.
<code>trap</code>	This message is sent by an agent to notify a management station of significant events.

The SNMP monitoring agent is based on the open-source Net-SNMP project, version 5.7.3. More information on SNMP in general, and the data structures used to support SNMP installations, is available from the NET-SNMP project Web site: <https://net-snmp.sourceforge.io/>.

This site includes some support information and offers access to discussion e-mail lists. You can use the discussion lists to monitor subjects that might affect the operation or security of the SNMP agent or command-line utilities.



Discuss any enquiries arising from information on the NET-SNMP Web site with Support before posting potentially sensitive information to the

24.1. Installing the SNMP agent

The SNMP agent is installed with the installation of the Security World Software and starts automatically.

24.1.1. Default installation settings

When installing Security World Software, you may be prompted to select Security World Software components from a list. If you select **all** components, then the SNMP agent is installed as part of a full Security World Software installation. The default installation directory for the nShield Management Information Base (MIB) and the SNMP configuration files (**snmp.conf** and **snmpd.conf**) is **/opt/nfast/etc/snmp/**.

24.1.2. Do you already have an SNMP agent running?

If you already have another SNMP agent running, you must configure the ports used by the agents in order to avoid conflicts before enabling the SNMP agent. A port is assigned by editing the **agentaddress** entry in the **snmpd.conf** file or by editing the **defaultPort** entry in **snmpd.conf** file. If both files have been edited, the **agentaddress** entry in **snmpd.conf** file takes priority for **snmpd**, and the **defaultPort** entry in **snmpd.conf** is ignored.

If no existing SNMP agent is found, the SNMP agent runs on the default port 161. If an existing SNMP agent is detected, and no SNMP agent configuration files are found (implying a fresh installation), the installer automatically configures the SNMP agent to use the first unused port above 161 by creating a new **snmpd.conf** configuration file with the appropriate directive. It then displays a message indicating the number of the port that is has selected.

If an existing SNMP agent is found and an existing SNMP agent installation exists, the installer checks the existing configuration files for an appropriate directive and warns you if one does not exist. If you need to edit these configuration files yourself, a port is assigned by editing the **agentaddress** entry in **snmpd.conf** file or editing the **defaultPort** entry in **snmpd.conf** file. If both files have been edited, the **agentaddress** entry in **snmpd.conf** file takes priority for **snmpd**, and the **defaultPort** entry in **snmpd.conf** is ignored.

24.1.3. Starting the SNMP agent

The SNMP agent is started automatically however it can be stopped and started manually.

To stop, start, or restart (stop and immediately start again) the SNMP daemon:

```
/etc/init.d/nc_ncsnmpd stop|start|restart
```

See [The SNMP configuration file: snmp.conf](#) for more information on additional parameters accepted by snmpd.

24.2. Basic configuration

24.2.1. Protecting the SNMP installation

The SNMP agent allows other computers on the network to connect to it and make requests for information. The SNMP agent is based on the NET-SNMP code base, which has been tested but not fully reviewed by Entrust. We strongly recommend that you deploy the SNMP agent only on a private network or a network protected from the global Internet by appropriate network protection systems (e.g. a firewall, a network Intrusion Detection/Prevention System, etc.).

The default nShield SNMP installation allows read-only access to the Management Information Base (MIB). There is no default write access to any part of the MIB.

Every effort has been taken to ensure the confidentiality of cryptographic keys even when the SNMP agent is enabled. In particular, the nShield module is designed to prevent the theft of keys even if the security of the host system is compromised, provided that the Administrator Cards are used only with trusted hosts. Care must be used when changing the configuration of the SNMP agent.



We strongly advise that you use the SNMP User-based Security Model (USM) with Authentication and Privacy protocols selected, to ensure only authorised users can obtain information from the SNMP agent and the confidentiality and data integrity of the transferred information is protected.

Care has also been taken to ensure that malicious attackers are unable to inundate your module with requests by flooding your SNMP agent. Command results from administration or statistics commands are cached, and thus the maximum rate at which the SNMP agent sends commands to the module is throttled. For more information on setting the cache time-outs. see [The SNMP configuration file: snmp.conf](#).

24.2.2. Configuring the SNMP agent

The Security World Software package uses various configuration files to configure its applications. This section describes the overall nature of the configuration files for the SNMP agent.

If you are installing the SNMP agent to a host that has an existing SNMP agent installation, you may need to edit the SNMP configuration files (`snmpd.conf` and `snmp.conf`) associated with the SNMP agent to change the port on which the agent listens for SNMP requests. For more information, see [Do you already have an SNMP agent running?](#).



Make sure you protect access to the configuration files, since these contain information that defines the security parameters of the SNMP system.

By default, the SNMP configuration files are located in the `/opt/nfast/etc/snmp/` directory.

24.2.2.1. Re-reading SNMP configuration files

The SNMP agent reads its configuration files on startup, and any changes made after this point will have no effect. If new directives are added and need to be applied, the SNMP agent can be forced to re-read its configuration files with:

- An `snmp set` of integer(1) to `enterprises.ncipher.reloadConfig.0(.1.3.6.1.4.1.7682.999.0)`
- kill `-HUP` signal sent to the `snmpd` agent process
- stop then restart the SNMP agent.

24.2.2.2. The SNMP configuration file: `snmp.conf`

The `snmp.conf` configuration file contains directives that apply to all SNMP applications. These directives can be configured to apply to specific applications. The `snmp.conf` configuration file is not required for the agent to operate and report MIB entries.

24.2.2.3. The SNMP agent configuration file: `snmpd.conf`

The `snmpd.conf` configuration file defines how the SNMP agent operates. It is required only if an agent is running.

The `snmpd.conf` file can contain any of the directives available for use in the `snmp.conf` file and may also contain the following Security World Software-specific directives:

Directive	Description
<code>statsttimeout</code>	This directive specifies the length of time for which statistics commands are cached. The default is 5 seconds.
<code>admintimeout</code>	This directive specifies the length of time for which administrative commands are cached. The default is 60 seconds.
<code>keytable</code>	This directive sets the initial state of the key table to <code>none</code> , <code>all</code> , or <code>query</code> . See listKeys in Administration sub-tree overview .
<code>enable_trap_zero_suffix</code>	This directive appends the '0' suffix to object identifiers (OIDs) for backward compatibility. The default is <code>0</code> (disabled): the directive can be set to <code>1</code> to restore the suffix. Valid values are 0 and 1.
<code>memoryUsageOkThreshold</code>	This directive specifies the threshold (as a percentage) below which HSM memory usage is considered to be ok. The default is 0. See Memory usage monitoring for more details.
<code>memoryUsageHighThreshold</code>	This directive specifies the threshold (as a percentage) at which HSM memory usage is considered to be too high. The default is 0. See Memory usage monitoring for more details.



There may be a tolerance gap between the `memoryUsageOkThreshold` and the `memoryUsageHighThreshold` values.



The timeouts should be set to values that achieve a balance between receiving up to date information whilst preventing excessive load.

24.2.3. The SNMP agent persistent configuration file

On running the SNMP agent for the first time, the `persist` directory will be created. This contains configuration files that are maintained by the SNMP agent. This directory will be created in the following location:

```
/opt/nfast/etc/snmp/persist
```

Modifications should only be made to the persist folder's `snmp.conf` file in order to create users. The files within this directory should otherwise only be managed by the SNMP agent itself.

User creation can be performed with the `createUser` directive. See [USM users](#). On initialization of the agent the information is read from the file and the lines are removed (eliminating the storage of the master password for that user) and replaced with the key that is derived from it. This key is a localised key, so that unlike the password, if it is stolen it can not be used to access other agents.



Do not modify the persistent `snmpd.conf` file while the agent is running. The file is only read on initialization of the agent and it is overwritten when the SNMP agent terminates. Any changes made to this file while the SNMP agent directives is running will be lost. The SNMP agent should be stopped prior to adding `createUser` directories to the configuration file.

24.2.4. Agent Behaviour

There are a small number of directives that control the behaviour of the SNMP Agent when considering it as a daemon providing a network service.

24.2.5. agentaddress directive

The listening address(es) that the SNMP Agent will use are defined by the `agentaddress` directive. It takes a comma separated list of address specifiers where an address specifier consists of one or more of:

- a transport specifier `udp:` or `tcp`
- a hostname or IPv4 address
- a port number (e.g. `:161` or `:1161`).

The default behaviour is to listen on UDP port 161 on all IPv4 interfaces (i.e. equivalent to `udp:161`).

```
agentaddress localhost : 161,tcp:1161
```

`agentaddress` will listen on UDP port 161, but only on the loopback interface (the port specification `:161` is not strictly necessary as this is the default port). It will also listen on TCP port 1161 on all IPv4 interfaces.

24.2.6. agentgroup and agentuser directives

The user and group that the SNMP Agent changes to after opening the listening port(s) are defined using the `agentgroup` and `agentuser` directives. The following must be used:

```
agentgroup ncsnmpd
agentuser ncsnmpd
```


24.2.7. System information

Most of the scalar objects in the .iso.org.dod.internet.mgmt.mib-2.system sub-tree can be configured.

```
sysLocation STRING
sysContact STRING
sysName STRING
```

The three directives above set the system location, contact or name for the SNMP Agent respectively. Ordinarily these objects are writable via a suitably authorised SNMP SET request, however, specifying one of these directives in the configuration file makes the corresponding object read-only.

```
sysServices INTEGER
```

Sets the value of the sysService.0 object. RFC1213 defines how the integer value is calculated.

```
sysDescr STRING
sysObjectID OID
```

The two directives above set the system description and object ID for the agent. These objects are not SNMP-writable, but these directives can be used by a network administrator to configure suitable values for them.

24.3. USM users

The SNMPv3 protocol supports a User based Security Model as defined in RFC-3414. USM provides authentication and privacy (encryption) functions and operates at the message level allowing for the following security level to be used with SNMPv3:

- Communication without authentication and privacy (**noauth**)
- Communication with authentication and without privacy (**auth**)
- Communication with authentication and privacy (**priv**).

Within this document the three possible security levels are referred to as **noauth**, **auth** and **priv**. However, other forms are sometimes used within the NET-SNMP and the equivalents are:

Security level	Equivalents
noauth	noauthnopriv

Security level	Equivalents
auth	authnopriv
priv	authpriv

Users can be added to the SNMP configuration with the `createUser` directive, defining the security mechanisms to be used.

```
createUser [-e ENGINEID] username [SHA authpass phrase] [AES privpass phrase]
```

It would not normally be necessary to specify the engine ID, but if it is specified, `ENGINEID` is defined as a hexadecimal string of octets starting with the 0x prefix. The encoding of the engine ID is defined in the description of `SnmpEngineID` from RFC3411. The following recommendations should be followed when defining the security parameters for SNMPv3:

- Select a 'Security Level' of Priv, (`authpriv`) or auth (`authNoPriv`).
 - `Priv` is the preferred 'Security Level', since this will provide both data source authentication and confidentially protection for the SNMP messages.
 - `auth` is the minimum 'Security Level' that should be selected, since this will ensure that SNMP data sent/received has not been tampered with and has been sent from an authorised entity.
- Define separate `authpass phrase` and `privpass phrase`.
 - It is good security practice to have key separation.
- Use randomly generated pass phrases which contain upper and lower case characters, numbers and symbols (e.g. ASCII characters 0x20 - 0x7E).
 - This should give an entropy per character of 6.57bits,
- Use either 15 char for 96 bits of security strength keys and 20 char for 128 bits security strength keys.
 - The minimum length of both `auth` and `priv` pass phrases is eight characters.
 - If a random pass phrase is not used, consult NIST SP800-63-2 - Appendix A to determine the security strength of the password and the resultant keys. See <https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-63-2.pdf>.



MD5 and DES are not supported or enabled in the nShield distribution of SNMP. Only SHA may be used for authentication, and only AES may be used for privacy (encryption).

It is strongly recommended that `createUser` directives be added to the `persist/snmpd.conf` file, so that the pass phrases are not available after the SNMP agent is installed. See [USM users](#). The user can then be referenced in access control directives(s) after which it can be

used.

24.4. Traditional access control

Most simple access control requirements can be specified using the directives **rouser** /**rwuser** (for SNMPv3) or **rocommunity**/**rwcommunity** (for SNMPv1 or SNMPv2c).

```
rouser [-s usm] USERNAME [noauth | auth | priv [OID | -V VIEW [CONTEXT]]]
rwuser [-s usm] USERNAME [noauth | auth | priv [OID | -V VIEW [CONTEXT]]]
```

These directives specify that an SNMPv3 user (USERNAME) will be allowed read-only or read-write access respectively. The default (unspecified) security level is **auth**, which is the recommended minimum security level (see above). It is not recommended to use the usm security level **noauth**, where all SNMP messages are unauthenticated and any tampering of the message cannot be detected. Using **noauth** will reduce the security of the SNMP messages to the level of SNMPv1 or SNMPv2c.

OID restricts access for that user to the subtree rooted at the given OID.

VIEW restricts access for that user to the specified View-based Access Control Model (VACM) view name. An optional context can also be specified, or **context** to denote a context prefix. If no context field is specified (or the token ***** is used), the directive will match all possible contexts. (Contexts are a mechanism within SNMPv3 whereby an agent can support parallel versions of the same MIB objects, referring to different underlying data sets.)

A security model can be specified with **-s SECMODEL** however the default security model **usm** is the only security model which is supported in the nShield distribution of SNMP.

Example:

- Read-only user with access to the full OID tree requiring authentication as a minimum:

```
rouser user1
```

- Or

```
rouser -s usm user1 auth .1
```

- Read-only user with access to the nShield MIB allowing unauthenticated requests:

```
rouser user2 noauth .1.3.6.1.4.1.7682
```

- Read-write user with access to the full OID tree requiring authentication as a minimum:

```
rwuser user3
```

Or

```
rwuser user3 auth .iso
```

- Read-write user with access to the snmpVacmMIB subtree requiring authentication and encryption:

```
rwuser user4 priv snmpVacmMIB
```

Or

```
rwuser user4 priv .1.3.6.1.6.3.16
```

```
rocommunity COMMUNITY [SOURCE [ OID | -V VIEW [CONTEXT]]]
rwcommunity COMMUNITY [SOURCE [ OID | -V VIEW [CONTEXT]]]
```

Specifies an SNMPv1 or SNMPv2c community that will be allowed read-only (**GET** and **GET-NEXT**) or read-write (**GET**, **GETNEXT** and **SET**) access respectively. By default, this will provide access to the full OID tree for such requests, regardless of where they were sent from. **SOURCE** allows access either from a particular range of source addresses, or globally (**default**). A restricted source can either be a specific hostname or address (e.g. **localhost** or 127.0.0.1), or a subnet - represented as IP/MASK (e.g. 10.10.10.0/255.255.255.0), or IP/BITS (e.g. 10.10.10.0/24).

OID **VIEW** and **CONTEXT** are as defined for **rouser** and **rwuser**.

Example:

- Setting up a read-only community named **public** that can be accessed by any user with the community name:

```
rocommunity public
```

- Setting up a read/write community named **private** that can only be accessed from the machine on which the agent is running:

```
rocommunity private localhost
```

In each case, only one directive should be specified for a given SNMPv3 user, or community string. It is not appropriate to specify both `rouser` and `rwuser` directives referring to the same SNMPv3 user (or equivalent community settings). The `rwuser` directive provides all the access of `rouser` (as well as allowing `SET` support). The same applies to `rwcommunity` and `rocommunity`.

More complex access requirements (such as access to two or more distinct OID subtrees, or different views for `GET` and `SET` requests) should use VACM configuration directives.

24.5. VACM configuration

The full flexibility of the VACM, for example allowing access to two or more distinct OID subtrees, or different access requirements for reading and writing, is available using four configuration directives - `com2sec`, `group`, `view` and `access`. The directives essentially define who has access and what they have access to using four directives. The first two directives (`com2sec` and `group`) define the who, while the last two (`view` and `access`) define the what.

```
Com2sec [-Cn CONTEXT] SECNAME SOURCE COMMUNITY
```

Maps an SNMPv1 or SNMPv2c community string to a security name. As it defines the community and maps it to a security name, `rocommunity`/`rwcommunity` directives are not required when using the directive.

`SECNAME` is the security name to be defined.

`SOURCE` is as defined for the `rocommunity`/`rwcommunity` directives above.

`COMMUNITY` defines the community name to be mapped to the security name. The same community string can be specified in several separate directives with different source tokens, and the first source/community combination that matches the incoming request will be selected. Various source/community combinations can also map to the same security name.

`CONTEXT` if defined (using `-Cn`), means that the community string will be mapped to a security name in the named SNMPv3 context. Otherwise the default context ("") will be used.

Example:

Creating three SNMPv1/v2c community names (`private`, `public` and `ltd`), where `private` and `ltd` only allow requests from the machine on which the SNMP Agent is running (note lines beginning with a `#` in `snmpd.conf` are treated as comments):

```
# [-Cn CONTEXT] SECNAME SOURCE COMMUNITY
```

```
com2sec "" sec_private localhost private
com2sec sec_public default public
com2sec sec_limited localhost ltd
```

```
group GROUP v1 | v2c | usm SECNAME
```

Maps a security name (in the specified security model) into a named group. Several group directives can specify the same group name, allowing a single access setting to apply to several users and /or community strings. Note that groups must be set up for the two community-based models separately - a single **com2sec** directive will typically be accompanied by two **group** directives.

GROUP is the group name being defined/added to **v1**, **v2c** or **usm** defines the security model to which the definition relates **SECNAME** is the security (USM user name or security name defined by **com2sec** to be added to the group.

Example:

Creating three groups (**grp_private**, **grp_public**, **grp_limited**) for three USM users (**user1**, **user2** and **user3**) and the three communities shown in the **com2sec** example above:

```
# GROUP v1|v2c|usm SECNAME
Ggroup grp_private v1 sec_private
group grp_private v2c sec_private
group grp_private usm user1

group grp_public v1 sec_public
group grp_public v2c sec_public
group grp_public usm user2

group grp_limited v1 sec_limited
group grp_limited v2c sec_limited
group grp_limited usm user3
```

```
view VNAME included | excluded OID [MASK]
```

Defines a named **view** - a subset of the overall OID tree. This is most commonly a single subtree, but several **view** directives can be given with the same view name (**VNAME**), to build up a more complex collection of OIDs. An optional mask can also be specified, providing a means of indicating which parts of the OID must be matched.

VNAME is the view being modified.

included | **excluded** allows you to define whether the view includes or excludes the subtree, allowing the definition of a more complex view (e.g. by excluding certain sensitive objects from an otherwise accessible subtree).

MASK is an optional list of hex octets (separated by '.' or ':') whose bits indicate which OID sub-identifiers to match against. So for example if we assume we have an OID with 11 sub-identifiers (`.1.3.6.1.x.y.z.table.entry.column.1`) where the last four relate to a table, an entry, a column and index 1, specifying a **MASK** value of " `FF.A0` " (i.e. 111111110100000) maps to this OID as follows:

```
.1.3.6.1.x.y.z.table.entry.column.1
1 1 1 1 1 1 1 1 1 0 1
```

i.e. this mask means all parts of the OID except the column must match, therefore defining a view to any column of the first row of the table.

By including and excluding various aspects of the full OID tree, it is possible to define fine grained visibility within a view's definition.

Example:

Creating five views where `vw_sysContact` only has access to the `system.sysContact.0` OID, `vw_nCipher` only has access to the MIB, `vw_global` has access to the full OID tree, `vw_nCipher_stats` only has access to `nCipher.nC-series.statistics` and `vw_nCipher_admin` only has access to `nCipher.nC-series.administration`:

#	VNAME	included excluded	OID	[MASK]
view	vw_sysContact	excluded	.1	
view	vw_sysContact	included	system.sysContact.0	FF.80
view	vw_nCipher	excluded	.iso	
view	vw_nCipher	included	.1.3.6.1.4.1.7682	
view	vw_global	included	.1	
view	vw_nCipher_stats	excluded	.1	
view	vw_nCipher_stats	included	enterprises.nCipher.nC-series.statistics	
view	vw_nCipher_admin	excluded	.1	
view	vw_nCipher_admin	included	enterprises.nCipher.nC-series.administration	

```
access GROUP CONTEXT any | v1 | v2c | usm noauth | auth | priv exact | prefix READ WRITE NOTIFY
```

Maps from a group of users/communities (with a particular security model and minimum security level, and specific context) to one of three views, depending on the request being processed.

GROUP is a group name defined by the group directive and specifies the group that access is being defined for.

CONTEXT specifies the context for the access (the default context is the empty string ""). The context of incoming requests must match against the context either exactly or by prefix, as

specified by the choice of **exact** | **prefix** made in this directive.

any, **v1**, **v2c**, or **usm** define the security model to which this definition relates.

noauth | **auth** | **priv** define the security level to which this definition relates. For **v1** or **v2c** access, this will need to be **noauth** as these protocols do not support authentication.

exact | **prefix** specify how **CONTEXT** should be matched against the context of the incoming request, either an exact match to **CONTEXT**, or prefixed by **CONTEXT**.

READ, **WRITE** and **NOTIFY** specifies the view to be used for **GET***, **SET** and **TRAP/INFORM** requests (although the **NOTIFY** view is not currently used). The keyword **none** is used if there is to be no access for that type of request.

Example:

Specifying that:

- SNMPv1 requests using the public community only have read access to the enterprises.nCipher.nC-series.statistics subtree,
- SNMPv2c requests using the public community only have read access to the enterprises.nCipher.nC-series.administration.subtree,
- SNMPv3 requests using the user2 USM user, must as a minimum be authenticated, and have read, notify access to the nCipher MIB (i.e. enterprises nCipher)
- SNMPv3 requests using the user1 USM user, must as a minimum be authenticated and encrypted, and have read, write and notify access to the full OID tree. Note that since requests must be authenticated and encrypted as a minimum, SNMPv1 and v2c requests using the private community cannot be made even though the community is included in grp_private.
- SNMPv1 and SNMPv2 requests using the ltd community and SNMPv3 requests using the user3 USM user, do not require to be authenticated or encrypted, and have read, write access to the system.sysContact.0 OID.

#	GROUP	CONTEXT	SECMODEL	LEVEL	PREFIX	READ	WRITE	NOTIFY
access	grp_public	""	v1	noauth	exact	vw_nCipher_stats	none	none
access	grp_public	""	v2c	noauth	exact	vw_nCipher_admin	none	none
access	grp_public	""	usm	auth	exact	vw_nCipher	none	vw_nCipher
access	grp_private	""	any	priv	exact	vw_global	vw_global	
vw_global								
access	grp_limited	""	any	noauth	exact	vw_sysContact	vw_sysContact	
none								

24.6. Trap Configuration

The distribution of SNMP supports SNMPv1, SNMPv2 and SNMPv3 traps. Control over

these traps is defined with a number of directives:

24.6.1. SNMPv1 and SNMPv2 traps

```
trapcommunity COMMUNITY
```

Defines the default community to be used when sending SNMPv1 or SNMPv2 traps. Note that this directive must be used prior to a `trapsink` or `trap2sink` directive that wishes to use this community.

COMMUNITY the community name to be used.

Example:

```
trapcommunity traps
```

```
trapsink HOST [COMMUNITY [PORT]]
trap2sink HOST [COMMUNITY [PORT]]
```

Defines a destination for SNMPv1 or SNMPv2 traps generated by the agent.

HOST is an address specifier defining the network target that traps will be sent to. It consists of an optional transport specifier (`udp` (default if not specified) or `tcp`) followed by a host-name or IPv4 address followed by an optional port number, delimited by colons ":". (e.g. `localhost` or `tcp:192.168.137.2:163`).

COMMUNITY if specified will be the community name used for the traps. If it is not specified, the most recently specified `trapcommunity` will be used.

PORT allows for port-number to be defined if it is not present as part of the **HOST** specification. If no port is defined, the default port number of 162 will be used.

When a TCP transport specifier is used the SNMP agent establishes the TCP connection with the trap manager at start-up. Therefore the trap manager must be started before the SNMP agent otherwise an error is reported for the line in the `snmpd.conf` file which defines the trap manager.

Likewise when the TCP connection between the SNMP agent and the trap manager is dropped, traps are lost. Therefore it is inadvisable to use TCP instead of UDP for the transport specifier of trap managers.

If TCP is used for the connection between the SNMP agent and the trap manager and the connection is lost, to re-establish the connection the SNMP agent must be restarted, with

the trap manager running and able to accept a TCP connection from the SNMP agent.

For issues with the trap manager accepting TCP connections from a SNMP agent refer to trap manager documentation.

Example:

```
trap2sink udp:192.168.137.220:162 traps
```

24.6.2. SNMPv3 traps

```
trapsess [SNMPCMD_ARGS] HOST
```

Defines the configuration for a trap. This is the only way to define SNMPv3 traps and it is an alternative method for defining SNMPv1 and SNMPv2 traps.

SNMPCMD_ARGS are arguments that would be used for an equivalent **snmptrap** command. So for example to send an SNMPv3 trap as USM user **user1** with authentication and encryption, the value **-v3 -u user1 -1 priv** would be used.

HOST see host definition for **trap2sink** above.

Example:

```
trapsess -v3 -u user1 -1 priv udp:192.168.137.220:162
trapsess -v2c -c public 192.168.137.221:162
```

24.7. Using the SNMP agent with a manager application



The nShield SNMP monitoring agent provides MIB files that can be added to your (third-party) SNMP manager application.

24.7.1. Manager configuration

The manager application is the interface through which the user is able to perform network management functions. A manager communicates with agents using SNMP primitives (**get**, **set**, **trap**) and is unaware of how data is retrieved from, and sent to, managed devices. This form of encapsulation creates the following:

- The manager is hidden from all platform specific details
- The manager can communicate with agents running on any IP-addressable machine.

As a consequence, manager applications are generic and can be bought off the shelf. You may already be running SNMP managers, and if so, you can use them to query the SNMP agent.



The manager is initially unaware of the MIB tree structure at a particular node. Managed objects can be retrieved or modified, but only if their location in the tree is known.

It is more useful if the manager can see the MIB tree present at each managed node. The MIB module descriptions for a particular node must be parsed by a manager-specific MIB compiler and converted to configuration files. These files are read by the manager application at run time.

The SNMP agent is designed to monitor all current nShield modules, working with all supported versions of nShield firmware (contact Support for details of supported firmware).

24.7.2. MIB module overview

A large proportion of the SNMP system is fully specified by the structure of the MIB; the behavior of the agent depends on relaying information according to the layout of the MIB.

The MIB module resides at a registered location in the MIB tree determined by the Internet Assigned Numbers Authority (IANA). The private enterprise number of 7682 designated by the IANA corresponds to the root of the branch, and by convention this (internal) node is the company name.

The MIB module groups logically related data together, organizing itself into a classification tree, with managed objects present at leaf nodes. The nC-series node (`enterprises.ncipher.nc-series`) is placed as a sub-tree of the root (`enterprises.ncipher`); this allows future product lines to be added as additional sub-trees. The structure of the tree underneath the registered location is vendor-defined, and this specification defines the structure chosen to represent Security World Software-specific data.

The MIB file can be found in the following location:

```
/opt/nfast/etc/snmp/mibs/ncipher-mib.txt
```

24.7.3. MIB functionality

The MIB module separates module information into the following categories:

- Retrieval of status and information about installed nC-series modules

- Retrieval of live statistics of performance of installed nC-series modules

These categories form the top-level nodes of the sub-tree; the functionality of the first category is in the administration sub-tree, and the second category is in the statistics sub-tree. The top-level tree also contains three items that it would be useful to check at-a-glance:

Node name	R/W	Type	Remarks
<code>hardserverFailed</code>	R	TruthValue	True if the remote hardserver is not running. If the hardserver is not running, then most of the rest of the information is unreliable or missing.
<code>modulesFailed</code>	R	TruthValue	True if any modules have failed.
<code>load</code>	R	Unsigned32	Percentage of total available capacity currently utilized.

24.7.3.1. Traps

The traps sub-tree (`enterprises.nCipher.nC-series.nC-traps`) contains traps that the SNMP agent sends when certain events occur. For details on configuring traps, see [USM users](#).

The following table gives details of the individual traps:

Node name	Description
<code>hardserverAlert</code>	This trap is sent when the hardserver fails or is shut down.
<code>hardserverUnAlert</code>	This trap is sent when the hardserver restarts.
<code>moduleAlert</code>	This trap is sent when a module fails.
<code>moduleUnAlert</code>	This trap is sent when a module is restarted after a failure.
<code>psuAlert</code>	This trap is sent when a PSU fails.
<code>psuUnAlert</code>	This trap is sent when a previously-failed PSU is working again.
<code>fanfailureAlert</code>	This trap is sent when a fan fails.
<code>fanfailureUnAlert</code>	This trap is sent when a previously-failed fan is working again.
<code>memoryUsageHighAlert</code>	This trap is sent when the HSM memory usage high threshold has been reached or exceeded by an HSM. See section on Memory usage monitoring below for more details.
<code>memoryUsageOkAlert</code>	This trap is sent when the memory usage for an HSM falls below the HSM memory usage ok threshold. See section on Memory usage monitoring below for more details.



Some traps can take up to five minutes to be received.



Other generic Net-SNMP traps may also be received. These include the two below, see Net-SNMP project website for more details.

Net-SNMP trap name	Description
<code>SNMPv2-MIB::coldStart</code>	This trap is sent when the SNMP agent is started
<code>NET-SNMP-AGENT-MIB::nsNotifyShutdown</code>	This trap is sent when the SNMP agent is stopped

24.7.4. Memory usage monitoring

The HSM memory usage thresholds and memory usage traps provide a mechanism to monitor HSM memory usage for HSMs in which the SNMP agent's client computer are enrolled.

With memory usage monitoring enabled, there will be a `memoryUsageHighAlert` trap sent each time the currently in-use `memoryUsageHighThreshold` is reached or exceeded by an HSM.

With memory usage monitoring enabled, a `memoryUsageHighAlert` trap is also sent:

- If the SNMP agent starts up and recognises that there are HSMs in a high memory usage state or,
- If HSMs in a high memory usage state are enrolled or,
- If the SNMP agent loses and then re-gains contact with the local hardserver which is connected to HSMs in a high memory usage state or,
- If failed HSMs in a high memory usage state then recover.

For each of the four scenarios above, one `memoryUsageHighAlert` trap will be sent for each HSM in a high memory usage state.

With memory usage monitoring enabled, there will be a `memoryUsageOkAlert` trap sent each time the memory usage for an HSM falls below the currently in-use `memoryUsageOkThreshold`.

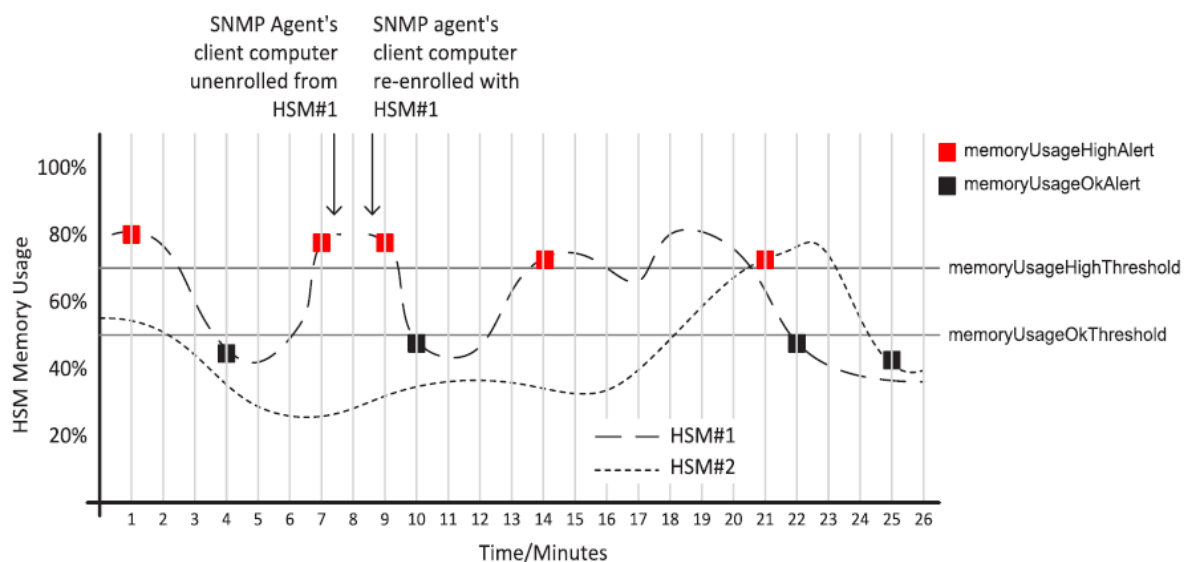
The value for `memoryUsageOkThreshold` is read from the `snmpd.conf` file on starting the SNMP agent and is used provided it contains an integer value in the range 0 to 100 (inclusive); otherwise, the default value of 0 is used. The value for `memoryUsageHighThreshold` is processed in the same way.

Memory usage monitoring is enabled unless the in-use values for `memoryUsageOkThreshold` and `memoryUsageHighThreshold` are both 0 or the in-use values are such that `memoryUsageOk`

`Threshold > memoryUsageHighThreshold`.

For example, in `snmpd.conf`, if `memoryUsageOkThreshold` is assigned an invalid value and `memoryUsageHighThreshold` is assigned a valid value of say 75%, then memory usage monitoring will be enabled and the values 0% and 75% will be used respectively.

An example of memory usage monitoring by an SNMP agent on a client computer enrolled with 2 HSMs is given below:



24.7.5. Administration sub-tree overview

The administration sub-tree (`enterprises.nCipher.nC-series.administration`) contains information about the permanent state of the hardware and the connected modules. It is likely that most of the information in this branch rarely changes over time, unlike the `statistics` branch. The information given in the administration sub-tree is mostly acquired by the `NewEnquiry` command and is supplied both per-module and (where appropriate) aggregated over all modules.

The following table gives details of the individual nodes in the administration sub-tree:

Node name	R/W	Type	Remarks
<code>hardserverRunning</code>	R	Enum 1: Running 2: NotRunning	This variable reflects the current state of the hardware (<code>Running</code> or <code>NotRunning</code>).
<code>noOfModules</code>	R	Gauge32	Number of nC-series modules.
<code>hsVersion</code>	R	DisplayString	Hardware version string.

Node name	R/W	Type	Remarks
<code>globalSpeedIndex</code>	R	Gauge32	Number of 1024-bit signatures each second.
<code>globalminQ</code>	R	Gauge32	Minimum recommended queue.
<code>globalmaxQ</code>	R	Gauge32	Maximum recommended queue.
<code>SecurityWorld</code>	R	TruthValue	True if a Security World is installed and operational.
<code>swState</code>	R	DisplayString	Security World display flags, as reported by <code>nfkminfo</code> .
<code>listKeys</code>	R/W	Integer 1: none 2: all 3: query 4: resetquery	Controls the behavior of the key table (switch off, display all keys, enable individual attribute queries, clear the query fields). Displaying all keys can result in a very long list.
<code>serverFlags</code>	R	DisplayString	Supported hardserver facilities (the <code>NewEnquiry</code> level 4 flags).
<code>remoteServerPort</code>	R	Gauge32	TCP port on which the hardserver is listening.
<code>swGenTime</code>	R	DisplayString	Security World's generation time.
<code>swGeneratingESN</code>	R	DisplayString	ESN of the module that generated the Security World.

`listKeys` can be preset using the `keytable` config directive in `snmpd.conf` file (see [The SNMP configuration file: snmpd.conf](#)).

24.7.5.1. Security World hash sub-tree

The following table gives details of the nodes in the Security World hash sub-tree (`enterprises.nCipher.nC-series.administration.swHashes`):

Node name	R/W	Type	Remarks
<code>hashKNSO</code>	R	MHash	Hash of the Security Officer's key.
<code>hashKM</code>	R	MHash	Hash of the Security World key.
<code>hashKRA</code>	R	MHash	Hash of the recovery authorization key.

Node name	R/W	Type	Remarks
hashKRE	R	MHash	Hash of the recovery key pair.
hashKFIPS	R	MHash	Hash of the FIPS authorization key.
hashKMC	R	MHash	Hash of the module certification key.
hashKP	R	MHash	Hash of the pass phrase replacement key.
hashKNV	R	MHash	Hash of the nonvolatile memory (NVRAM) authorization key.
hashKRTC	R	MHash	Hash of the Real Time Clock authorization key.
hashKDSEE	R	MHash	Hash of the SEE Debugging authorization key.
hashKFTO	R	MHash	Hash of the Foreign Token Open authorization key.

24.7.5.2. Security World quorums sub-tree

The following table gives details of the nodes in the Security World quorums sub-tree (`enterprises.nCipher.nC-series.administration.swQuorums`):

Node name	R/W	Type	Remarks
adminQuorumK	R	Gauge32	The default quorum of Administrator cards.
adminQuorumN	R	Gauge32	The total number of cards in the ACS.
adminQuorumM	R	Gauge32	The quorum required for module reprogramming.
adminQuorumR	R	Gauge32	The quorum required to transfer keys for OCS replacement.
adminQuorumP	R	Gauge32	The quorum required to recover the pass phrase for an Operator card.
adminQuorumNV	R	Gauge32	The quorum required to access non volatile memory (NVRAM).
adminQuorumRTC	R	Gauge32	The quorum required to update the Real Time Clock.

Node name	R/W	Type	Remarks
adminQuorumDSEE	R	Gauge32	The quorum required to view full SEE debug information.
adminQuorumFTO	R	Gauge32	The quorum required to use a Foreign Token Open Delegate Key.

24.7.5.3. Module administration table

The following table gives details of the nodes in the module administration table (`enterprises.nCipher.nC-series.administration.moduleAdminTable`):

Node name	R/W	Type	Remarks
moduleAdminIndex	R	Gauge32	Module number of this row in the table.
mode	R	Integer 1: Operational 2: Pre-init 3: Init 4: Pre-maint 5: Maint 6: AccelOnly 7: Failed 8: Unknown	Current module state.
fwVersion	R	DisplayString	Firmware version string.
speedIndex	R	Gauge32	Speed index (approximate number of 1024-bit modulo exponentiation operations possible per second) of module
minQ	R	Gauge32	Module minimum recommended queue length
maxQ	R	Gauge32	Module maximum recommended queue length
serialNumber	R	DisplayString	Module Electronic Serial Number (ESN).
productName	R	DisplayString	

Node name	R/W	Type	Remarks
hwPosInfo	R	DisplayString	Hardware bus/slot info (such as PCI slot number).
moduleSecurityWorld	R	TruthValue	Indicates whether or not the module is in the current SW.
smartcardState	R	DisplayString	Description of smart card in slot (empty, unknown card, admin/operator card from current SW, failed). N/A for acceleration only modules.
moduleSWState	R	Integer 1: Unknown 2: Usable 3: MaintMode 4: Uninitialized 5: Factory 6: Foreign 7: AccelOnly 8: Failed 9: Unchecked 10: InitMode 11: PreInitMode 12: Unverified 13: UnusedTableEntry	Current module and Security World state.
moduleSWFlags	R	DisplayString	Security World flags for this module.
hashKML	R	MHash	Hash of the module's secret key.
moduleFeatures	R	DisplayString	Features enabled on this module.
moduleFlags	R	DisplayString	Like <code>serverFlags</code> , but for each module.
versionSerial	R	Gauge32	Firmware Version Security Number (VSN). See Version Security Number(VSN) .
hashKETI	R	MHash	<code>K_{NETI}</code> hash, if present.

Node name	R/W	Type	Remarks
longQ	R	Gauge32	Max. rec. long queue.
connectionStatus	R	DisplayString	Connection status (for imported modules).
connectionInfo	R	DisplayString	Connection information (for imported modules).
machineTypeSEE	R	DisplayString	SEE machine type.

24.7.5.4. Slot administration table

The following table gives details of the nodes in the slot administration table (`enterprises.nCipher.nC-series.administration.slotAdminTable`):

Node name	R/W	Type	Remarks
slotAdminModuleIndex	R	Integer32	Module number of the module containing the slot.
slotAdminSlotIndex	R	Integer32	Slot number (1-based, unlike nCore which is 0-based).
slotType	R	Integer 1: Datakey 2: Smart card 3: Emulated 4: Soft token 5: Unconnected 6: Out of range 7: Unknown	Slot type.
slotFlags	R	DisplayString	Flags referring to the contents of the slot (from <code>slotInfo</code>).

Node name	R/W	Type	Remarks
slotState	R	Integer 1: Unused 2: Empty 3: Blank 4: Administrator 5: Operator 6: Unidentified 7: Read error 8: Partial 9: Out of range	Partial refers to cards in a partially-created card set.
slotListFlags	R	DisplayString	Flags referring to attributes of the slot (from getslotlist).
slotShareNumber	R	Gauge32	Share number of card currently in slot, if present.
slotSharesPresent	R	DisplayString	Names of shares present in card currently in slot.

24.7.5.5. Card set administration table

The following table gives details of the nodes in the card set administration table (**enterprises.nCipher.nC-series.administration.cardsetAdminTable**):

Node name	R/W	Type	Remarks
hashKLTU	R	MHash	Hash of the token protected by the card set.
cardsetName	R	DisplayString	
cardsetK	R	Gauge32	Required number of cards in the card set.
cardsetN	R	Gauge32	Total number of cards in the card set.
cardsetFlags	R	DisplayString	Other attributes of the card set.
cardsetNames	R	DisplayString	Names of individual cards, if set.

Node name	R/W	Type	Remarks
<code>cardsetTimeout</code>	R	Gauge32	Token time-out period, in seconds, or 0 if none.
<code>cardsetGenTime</code>	R	DisplayString	Generation time of card set.

24.7.5.6. Key administration table

The key administration table is visible as long as the `listKeys` node in the administration sub-tree is set to a value other than `none`.

The following table gives details of the nodes in the key administration table (`enterprises.nCipher.nC-series.administration.keyAdminTable`):

Node name	R/W	Type	Remarks
<code>keyAppname</code>	R	DisplayString	Application name.
<code>keyIdent</code>	R	DisplayString	Name of key, as generated by the application.
<code>keyHash</code>	R	MHash	
<code>keyRecovery</code>	R	Integer 1: Enabled 2: Disabled 3: No key 4: Unknown 5: Invalid 6: Unset	The value <code>unset</code> is never returned by the key table. If you set the value <code>unset</code> , the keys are not filtered based on any of the attributes.
<code>keyProtection</code>	R	Integer 1: Module 2: Cardset 3: No key 4: Unknown 5: Invalid 6: Unset	The value <code>unset</code> is never returned by the key table. If you set the value <code>unset</code> , the keys are not filtered based on any of the attributes.

Node name	R/W	Type	Remarks
keyCardsetHash	R	MHash	Hash of the card set protecting the key, if applicable.
keyFlags	R	DisplayString	Certificate and public key flags.
keyExtraEntries	R	Gauge32	Number of extra key attributes.
keySEEEInteg	R	DisplayString	SEE integrity key, if present.
keyGeneratingESN	R	DisplayString	ESN of the module that generated the key, if present.
keyTimeLimit	R	Gauge32	Time limit for the key, if set.
keyPerAuthUseLimit	R	Gauge32	Per-authentication use limit for the key.

24.7.5.7. Key query sub-tree

The key query sub-tree is used if the `ListKeys` node in the administration sub-tree is set to `query`.

If these values are set, they are taken as required attributes for filtering the list of available keys; if multiple attributes are set, the filters are combined (AND rather than OR).

The following table gives details of the nodes in the key query sub-tree (`enterprises.nCipher.nC-series.administration.keyQuery`):

Node name	R/W	Type	Remarks
keyQueryAppname	R/W	DisplayString	Application name.
keyQueryIdent	R/W	DisplayString	Name of key, as generated by the application.
keyQueryHash	R/W	DisplayString	
keyQueryRecovery	R/W	Integer 1: Enabled 2: Disabled 3: No key 4: Unknown 5: Invalid 6: Unset	The value <code>unset</code> is never returned by the key table. If you set the value <code>unset</code> , the keys are not filtered based on any of the attributes.

Node name	R/W	Type	Remarks
keyQueryProtection	R/W	Integer 1: Module 2: Cardset 3: No key 4: Unknown 5: Invalid 6: Unset	The value unset is never returned by the key table. If you set the value unset , the keys are not filtered based on any of the attributes.
keyQueryCardsetHash	R/W	DisplayString	Hash of the card set protecting the key, if applicable.
keyQueryFlags	R/W	DisplayString	Certificate and public key flags.
keyQueryExtraEntries	R/W	Gauge32	Number of extra key attributes.
keyQuerySEInteg	R/W	DisplayString	SEE integrity key, if present.
keyQueryGeneratingESN	R/W	DisplayString	ESN of the module that generated the key, if present.
keyQueryTimeLimit	R/W	Gauge32	Time limit for the key, if set (0 for no limit).
keyQueryPerAuthUseLimit	R/W	Gauge32	Per-authentication use limit for the key (0 for no limit).

24.7.6. Statistics sub-tree overview

The statistics sub-tree (**enterprises.nCipher.nC-series.statistics**) contains rapidly changing information about such topics as the state of the nShield modules, the work they are doing, and the commands being submitted.



Do not rely on information returned from the agent to change instantaneously on re-reading the value. To avoid loading the nShield module with multiple time-consuming statistics commands, the agent can choose to cache the values over a specified period. You can configure this period in the agent configuration file see [The SNMP configuration file: snmp.conf](#).

24.7.6.1. Statistics sub-tree

The following table gives details of the nodes in the statistics sub-tree, and the module statistics table (`enterprises.nCipher.nC-series.statistics.moduleStatsTable`):

Node name	R/W	Type	Remarks
<code>moduleStatsIndex</code>	R	Integer	Module number of this row (for <code>moduleStatsTable</code>).
<code>hsuptime</code>	R	Counter32	Uptime of the hardserver.
<code>cmdCountAll</code>	R	Counter32	Returned aggregated for all modules and all commands.
<code>cmdBytesAll</code>	R	Counter32	
<code>cmdErrorsAll</code>	R	Counter32	Returned as for <code>cmdCount</code> , returned value is combined module errors added to hardserver marshalling/unmarshalling errors.
<code>replyCountAll</code>	R	Counter32	
<code>replyBytesAll</code>	R	Counter32	
<code>replyErrorsAll</code>	R	Counter32	See notes above for <code>cmdErrors</code> .
<code>clientCount</code>	R	Gauge32	
<code>maxClients</code>	R	Counter32	
<code>deviceFails</code>	R	Counter32	
<code>deviceRestarts</code>	R	Counter32	
<code>outstandingCmds</code>	R	Counter32	Total number of outstanding commands over all modules.
<code>load[All]</code>	R	Counter32	

24.7.6.2. Module statistics table

The following table gives details of the nodes in the module statistics table (`enterprises.nCipher.nC-series.statistics.moduleStatsTable`):

Node name	R/W	Type	Remarks
<code>moduleStatsIndex</code>	R	Integer	Module number of this row (for <code>moduleStatsTable</code>).
<code>uptime</code>	R	Counter32	Uptime of the module.
<code>cmdCount</code>	R	Counter32	Returned aggregated for all commands.

Node name	R/W	Type	Remarks
cmdBytes	R	Counter32	
cmdErrors	R	Counter32	Returned as for cmdCount all the different error states aggregated into one counter.
replyCount	R	Counter32	
replyBytes	R	Counter32	
replyErrors	R	Counter32	See notes above for cmdErrors .
loadModule	R	Counter32	
loadModule	R	Counter32	
objectCount	R	Gauge32	
clock	R	DisplayString	Depending on the module settings, this can require K_{NSO} permissions to read (and therefore depend on the installation parameters of the agent).
currentTemp	R	DisplayString	Character representation of the current temperature value (SNMP does not provide for a floating-point type). Only available on non-XC variants.
maxTemp	R	DisplayString	Maximum temperature the module has ever reached. Only available on non-XC variants.
nvRAMInUse	R	Gauge32	
volatileRAMInUse	R	Gauge32	
tempSP	R	DisplayString	Only available on XC variants.
currentCPUTemp1	R	DisplayString	Only available on XC variants.
currentCPUTemp2	R	DisplayString	Only available on XC variants.
currentFanSpeed	R	DisplayString	Only available on XC variants.
currentFanDuty	R	DisplayString	Only available on XC variants.
CPUVoltage1	R	DisplayString	Only available on XC variants.
CPUVoltage2	R	DisplayString	Only available on XC variants.
CPUVoltage3	R	DisplayString	Only available on XC variants.

Node name	R/W	Type	Remarks
CPUVoltage4	R	DisplayString	Only available on XC variants.
CPUVoltage5	R	DisplayString	Only available on XC variants.
CPUVoltage6	R	DisplayString	Only available on XC variants.
CPUVoltage7	R	DisplayString	Only available on XC variants.
CPUVoltage8	R	DisplayString	
CPUVoltage9	R	DisplayString	
CPUVoltage10	R	DisplayString	
CPUVoltage11	R	DisplayString	

24.7.6.3. nShield Connect statistics table

The following table gives details of the nodes in the nShield Connect statistics table (`enterprises.nCipher.nC-series.statistics.nethSMStatsTable`):

Node name	R/W	Type	Remarks
nethSMStatsIndex	R	Integer	Table index (not module number).
nethSMUptime	R	Counter32	Host system uptime.
nethSMCPUUsage	R	Gauge32	CPU usage of unit host processor.
nethSMUserMem	R	Gauge32	Total user memory of unit.
nethSMKernelMem	R	Gauge32	Total kernel memory of unit.
nethSMCurrentTemp	R	DisplayString	Internal unit temperature (sensor 1).
nethSMMaxTemp	R	DisplayString	Maximum recorded temperature (sensor 1).
nethSMCurrentTemp2	R	DisplayString	Internal unit temperature (sensor 2).
nethSMMaxTemp2	R	DisplayString	Maximum recorded temperature (sensor 2).
nethSMVoltage5V	R	DisplayString	unit 5V power reading.
nethSMVoltage12V	R	DisplayString	unit 12V power reading.
nethSMFan1Speed	R	Gauge32	Fan 1 speed (RPM).
nethSMFan2Speed	R	Gauge32	Fan 2 speed (RPM).
nethSMFan3Speed	R	Gauge32	Fan 3 speed (RPM).

Node name	R/W	Type	Remarks
nethSMIPAddress	R	IpAddress	IP address of unit.
nethSMDescription	R	DisplayString	Textual description of module (for example, <i>Local module 3</i>).
nethSMFan4Speed	R	Gauge32	Fan 4 speed (RPM).
nethSMVoltage3p3V	R	DisplayString	3.3V Supply Rail Voltage
nethSMCurrent3p3V	R	DisplayString	3.3V Supply Rail Current
nethSMCurrent5V	R	DisplayString	5V Supply Rail Current
nethSMCurrent12V	R	DisplayString	12V Supply Rail Current
nethSMVoltage5VSB	R	DisplayString	5V Supply Rail Voltage (Standby)
nethSMCurrent5VSB	R	DisplayString	5V Supply Rail Current (Standby)
nethSMTamperBattery1	R	DisplayString	Voltage of Tamper Battery 1
nethSMTamperBattery2	R	DisplayString	Voltage of Tamper Battery 2
nethSMPSUfailure	R	TruthValue	Power Supply failure status

24.7.6.4. Per connection statistics table

The following table gives details of the nodes in the per connection statistics table (`enterprises.nCipher.nC-series.statistics.connStatsTable`):

Node name	R/W	Type	Remarks
connStatsIndex	R	Integer32	Index of this entry.
connNumber	R	Integer32	Hardserver connection number.
connUptime	R	Counter32	Uptime of the connection.
connCmdCount	R	Counter32	Number of commands submitted through this connection.
connCmdBytes	R	Counter32	Number of bytes submitted through this connection.
connCmdErrors	R	Counter32	Number of marshalling errors on commands through this connection.
connReplyCount	R	Counter32	Number of replies received by this connection.
connReplyBytes	R	Counter32	Number of bytes received by this connection.

Node name	R/W	Type	Remarks
<code>connReplyErrors</code>	R	Counter32	Number of marshalling errors on replies through this connection.
<code>connDevOutstanding</code>	R	Gauge32	Number of commands outstanding on this connection.
<code>connQOutstanding</code>	R	Gauge32	Number of commands outstanding in the hardserver queue.
<code>connLongOutstanding</code>	R	Gauge32	Number of long jobs outstanding for this connection.
<code>connRemoteIPAddress</code>	R	IpAddress	IP Address of connection client.
<code>connProcessID</code>	R	Integer32	Process identifier reported by connection client.
<code>connProcessName</code>	R	DisplayString	Process name reported by connection client.
<code>connObjectTotal</code>	R	Gauge32	The total object count for a connection

24.7.6.5. Module/connection statistics table

The following table gives details of the nodes in the per module, per connection statistics table (`enterprises.nCipher.nC-series.statistics.connModuleStatsTable`).

Node name	R/W	Type	Remarks
<code>connModuleStatsConnId</code>	R	Integer	Identity of this connection
<code>connModuleStatsModuleIndex</code>	R	Integer	Index of the module entry
<code>connModuleStatsObjectCount</code>	R	Gauge32	The object count on this module for this connection

24.7.6.6. Fan table

The fan table provides the speeds of each fan on the remote module (HSM). The following table gives details of the nodes in the fan table (`enterprises.nCipher.softwareVersions.nethSMFanTable`):

Node name	R/W	Type	Remarks
<code>nethSMModuleIndex</code>	R	Integer32	Module number
<code>nethSMFanIndex</code>	R	Integer32	Fan number

Node name	R/W	Type	Remarks
nethSMFanSpeed	R	Gauge32	Fan speed (RPM)

24.7.6.7. Software versions table

The following table gives details of the nodes in the software versions table (`enterprises.nCipher.softwareVersions.softwareVersionsTable`):

Node name	R/W	Type	Remarks
compIndex	R	Integer	Table index.
compName	R	DisplayString	Component name.
compOutput	R	Component output name	Component name.
compMajorVersion	R	Gauge	
compMinorVersion	R	Gauge	
compPatchVersion	R	Gauge	
compRepository	R	DisplayString	Repository name.
compBuildNumber	R	Gauge	

24.8. SNMP agent command-line

24.8.1. SNMP agent (snmpd) switches

The SNMP agent that binds to a port and awaits requests from SNMP management software is `snmpd`. Upon receiving a request, `snmpd` processes the request, collects the requested information and/or performs the requested operation(s) and returns the information to the sender.

The SNMP agent supports a limited subset of command line switches that can be specified when starting the agent.

Usage

```
snmpd [-h] [-v] [-f] [-a] [-d] [-V] [-P PIDFILE]:] [-q] [-D] [-p NUM] [-L] [-l LOGFILE] [-r]
```

This command can take the following options:

Option	Description
-h	This option displays a usage message.
-H	This option displays the configuration file directives that the agent understands.
-v	This option displays version information.
-f	This option specifies not forking from the calling shell.
-a	This option specifies logging addresses.
-A	This option specifies that warnings and messages should be appended to the log file rather than truncating it.
-d	This option specifies the dumping of sent and received UDP SNMP packets.
-V	This option specifies verbose display.
-P	PIDFILE This option specifies the use of a file (PIDFILE) to store the process ID.
-q	This option specifies that information be printed in a more easily parsed format (quick print).
-D	This option turns on debugging output.
-p	NUM This option specifies running on port NUM instead of the default: 161.
-c	CONFFILE This option specifies reading CONFFILE as a configuration file.
-C	This option specifies that the default configuration files not be read.
-L	This option prints warnings and messages to stdout and err.
-s	This option logs warnings/messages to syslog.
-r	This option specifies not exiting if root-only accessible files cannot be opened.
-I	[-]INITLIST This option specifies a list of MIB modules to initialize (or not). Run snmpd with the -Dmib_init option for a list.
-l	LOGFILE This option prints warnings/messages to a file LOGFILE (by default, LOG-FILE=log/snmpd.log).

24.8.2. Using the SNMP command-line utilities

As an alternative to using an SNMP manager application, we supply several command-line utilities to test your SNMP installation and enable you to obtain information about your nShield module from the SNMP agent. These utilities support the **-h** (display a usage message) as described in the table above.

Utility	Description
snmpctest	This utility monitors and manages SNMP information.

Utility	Description
<code>snmpget</code>	This utility runs a single GET request to query for SNMP information on a network entity.
<code>snmpset</code>	This utility runs a single SET request to set SNMP information on a network entity.
<code>snmpgetnext</code>	This utility runs a single GET NEXT request to query for SNMP information on a network entity.
<code>snmptable</code>	This utility obtains and prints an SNMP table.
<code>snmptranslate</code>	This utility translates SNMP object specifications into human-readable descriptions.
<code>snmpwalk</code>	This utility communicates with a network entity using repeated GET NEXT requests.
<code>snmpbulkwalk</code>	This utility communicates with a network entity using BULK requests.



These tools are general purpose SNMP utilities and are configurable for use with other SNMP agents. For more information on configuring and using these tools, refer to the NET-SNMP project Web site: <http://net-snmp.sourceforge.net/>.

25. Morse code error messages

If a Hardware Security Module (HSM) encounters an unrecoverable error, it enters the error state. In the error state, the module does not respond to commands and does not write data to the bus.

The blue Status LED flashes the Morse distress code (SOS: three short pulses, followed by three long pulses, followed by three short pulses). The Morse distress code is followed by one of the error codes listed in the tables shown in this guide.

For internal security modules running firmware 2.61.2 and above, the error code listed in this chapter is also reported by the **enquiry** utility in the **hardware status field** of the **Module** and under **hardware errors** in the hardserver log.

Errors are a rare occurrence. If any module goes into the error state, except as a result of you issuing the **Fail** command, contact Support, and give full details of your set up and the error code.

Contact Support even if you successfully recover from the error by taking the recommended action. For troubleshooting information, see the relevant *Installation Guide* for your module type.

25.1. Reading Morse code

The following guidelines are useful when reading Morse code messages from the module:

- The duration of a dash (-) is 3 times the duration of a dot (.).
- The gap between components of a letter has the same duration as a dot.
- The gap between letters has the same duration as a dash.
- The duration of the gap between repeated series of letters (a Morse code word gap) is 7 times the duration of a dot.

The following table shows the error codes corresponding to numerals.

Numeral	Morse
1	. - - - -
2	.. - - -
3	... - -
4 -
5

Numeral	Morse
6	-
7	- - . . .
8	- - - . .
9	- - - - .
0	- - - - -

25.2. Runtime library errors

Memory failures can occur if the module is exposed to excessive heat. If you experience these errors, check the ventilation around the module. The module generates considerable heat and, if not well ventilated, may be operating at too high a temperature, even if the rest of your server room is at an appropriate temperature.

The runtime library error codes could be caused by firmware bugs or by faulty hardware. If any of these errors is indicated, reset the module.

Code				Meaning
OLC	- - -	. - . .	- . . .	SIGABRT: assertion failure and/or <code>abort()</code> called.
OLD	- - -	. - . .	- . .	Interrupt occurred when disabled.
OLE	- - -	. - . .	.	SIGSEGV: access violation.
OLI	- - -	. -	SIGSTAK: out of stack space.
OLJ	- - -	. - . .	. - - -	SIGFPE: unsupported arithmetic exception (such as division by 0).
OLK	- - -	. - . .	- . .	SIGOSERROR: runtime library internal error.
OLN	- - -	. - . .	- .	SIGFATALPANIC: error in error handling code.

Codes **OLD**, and **OLE** are more likely to indicate a hardware problem than a firmware problem.

To reset a unit that is in an error state, turn off the unit and then turn it on again.

25.3. Hardware driver errors

In general, the hardware driver error codes described in the following table indicate that some form of automatic hardware detection has failed. As well as indicating simple hardware failure, one of these error codes could indicate that there is a bug in the firmware or

that the wrong firmware has been loaded.



In the following table, the symbol “#” stands for a given numeral’s Morse code representation.

If any of these errors is indicated, contact support.

Code						Meaning
HL-..				M48T37 NVRAM (or battery) failed
H B	-...				Debug serial port initialization failed.
H C	-. .				Processing thread initialization failed.
HCP	-..	.-..			Card poll thread initialization failed.
H D	-..				Failure reading unique serial number.
H E				EEPROM failed on initialization.
HF-				Starting up crypto offload.
H I				Interrupt controller initialization failed.
H M	--				System hardware initialization failed.
H O	---				Token interface initialization failed.
H R-.				Random number generator failed. <div> <p>This code may also be generated if an attempt is made to downgrade firmware on an nShield Solo+ to version 2.50.x or older.</p> </div>
HRS-.	...			RNG startup failed.
H RTP-.	-	.-..		Periodic (scheduled daily) RNG selftest failed.
HRM-.	--			RNG data matched.
HS				Unexpected error from SCSI controller or host interface initialization failed.
HV-				Environment sensors failed (for example, temperature sensor)

Code						Meaning
HCV	-.-.	...-			CPLD wrong version for PCI policing firmware.
HPP--.	.--.			PCI Interface Policing failure.
HST	-			Speed test failed.
HRH			RTC hardware detection failed or random number generator detection failed.
HRH			RNG hardware failed during operation
KR	-.-	..				RSA selftest failed.
HM n	--	#			DSP n failed self-test at start up.
HC n CA	-.-.	#	-.-.	..	CPU n failed self-test; no memory for cached RAM test.
HC n C C	-.-.	#	-.-.	-.-.	CPU n failed self-test; CPU ID check failed.
HC n CF	-.-.	#	-.-.	...-	CPU n failed self-test; freeing memory for cached RAM test.
HC n C G	-.-.	#	-.-.	--.	CPU n failed self-test; setting up cached RAM test.
HC n CR	-.-.	#	-.-.	..	CPU n failed self-test; read error during cached RAM test.
HC n CV	-.-.	#	-.-.	...-	CPLD version number incorrect (nShield Solo).
HC n C W	-.-.	#	-.-.	..-	CPU n failed self-test; write error during cached RAM test.
HC n HD	-.-.	#	DRBG n failed self-test.
HC n KA	-.-.	#	-.-	..	CPU n failed selftest - AES known-answer test.
HC n KB	-.-.	#	-.-	-...	CPU n failed selftest - AES CMAC known-answer test.
HC n KC	-.-.	#	-.-	-.-.	CPU n failed selftest - ECDSA known-answer test
HC n KE	-.-.	#	-.-	.	CPU n failed self-test; DES known-answer test.
HC n KF	-.-.	#	-.-	...-	CPU n failed self-test; Triple-DES known-answer test.

Code						Meaning
H C n K H	- . - .	#	- . -	CPU <i>n</i> failed self-test; SHA-1 known-answer test.
H C n K I	- . - .	#	- . -	..	CPU <i>n</i> failed selftest - HMAC-SHA512 known-answer test.
H C n K J	- . - .	#	- . -	. - - -	CPU <i>n</i> failed selftest - HMAC-SHA256 known-answer test.
H C n K M	- . - .	#	- . -	- -	CPU <i>n</i> failed self-test; HMAC-SHA1 known-answer test.
H C n K N	- . - .	#	- . -	- .	CPU <i>n</i> failed selftest - HMAC-SHA224 known-answer test.
H C n K P	- . - .	#	- . -	. - - .	CPU <i>n</i> failed selftest - HMAC-SHA384 known-answer test.
H C n K R	- . - .	#	- . -	. - .	CPU <i>n</i> failed selftest - RSA known-answer test
H C n K S	- . - .	#	- . -	...	CPU <i>n</i> failed self-test; DSA known-answer test.
H C n L C	- . - .	#	. - .	- . - .	CPU <i>n</i> failed self-test; locking check.
H C n P S	- . - .	#	. - -	CPU <i>n</i> failed self-test; test terminated at start.
H C n R T	- . - .	#	. - .	-	CPU <i>n</i> failed selftest - RTC check.
H C n S A	- . - .	# - - .	CPU <i>n</i> failed self-test; no memory for uncached RAM test.
H C n S F	- . - .	# - .	CPU <i>n</i> failed self-test; freeing memory for uncached RAM test.
H C n S R	- . - .	# - .	CPU <i>n</i> failed self-test; read error during uncached RAM test.
H C n S W	- . - .	# - -	CPU <i>n</i> failed self-test; write error during uncached RAM test.
H C n T S	- . - .	#	-	...	CPU <i>n</i> failed self-test; could not start test.

25.4. Maintenance mode errors

The following error codes indicate faults encountered when a module is in the maintenance mode.

Code				Meaning	Action
I D	..	-..		Copies of metadata do not match when trying to run image.	Contact Support.
I H		Bad metadata: hash mismatch.	Repeat firmware upgrade.
I I		Execution image does not match metadata.	Contact Support.
I L	..	.-..		Bad metadata: either bad length or bad metadata when running loadboot application.	Repeat firmware upgrade.
I M	..	--		Bad metadata: malformed ImageMetaData.	Repeat firmware upgrade.
I P	..	.--.		Bad metadata: bad padding.	Repeat firmware upgrade.
I R	..	.-.		Bad metadata: extra bytes at end.	Repeat firmware upgrade.
I S		Image entry point not found.	Contact Support.
I U-		Bad metadata: ROM blank.	Repeat firmware upgrade.
I X	..	-...-		Bad metadata: malformed header.	Repeat firmware upgrade.
J H	.----		Both copies of metadata invalid.	Contact Support.
H Z E	--..	.	Monitor checksum failed.	Contact Support.
K F E	-.-	..-	.	Flash sector erase failed.	Repeat firmware upgrade.
K F P	-.-	..-	.--.	Flash sector program failed.	Repeat firmware upgrade.
M M D	--	--	-..	No memory for download buffer.	Contact Support.



For instructions on upgrading module firmware, see the appendix in the User Guide for your module type.

25.5. Operational mode errors

The following runtime library error codes could be caused by either bugs in the firmware or faulty hardware.

Code				Meaning	Action
D	-..			Fail command received.	Reset module by turning it off and then on again.

Code				Meaning	Action
T	-			Temperature of the module has exceeded the maximum allowable.	Restart your host computer, and improve module cooling.
G G G	--.	--.	--.	Failure when performing ClearUnit or Fail command.	Contact Support.
I J A	..	.---	.-	Audit logging: failed to send audit log message.	Contact Support.
I J B	..	.---	-...	Audit logging: no module memory (therefore failed to send audit log message).	Contact Support.
I J C	..	.---	-..	Audit logging: key problem or FIPS incompatibility (therefore failed to sign audit log message).	Contact Support.
I J D	..	.---	-..	Audit logging: NVRAM problem (therefore failed to configure or send audit log message).	Contact Support.



SOS IJA can occur for any type of log message (i.e. a log message, signature block or certifier block).



To improve the cooling of your PCIe module, increase the distance between PCIe cards, and increase the airflow through your host computer.

26. Application Performance Tuning

26.1. Job Count

To achieve the best throughput of cryptographic jobs (such as Sign or Decrypt) in your application, arrange for multiple jobs to be on the go at the same time, rather than doing them one at a time. This is true even when using only a single HSM in your system.

When using a Solo, Solo+, Connect or Connect+, around 40 outstanding jobs per HSM is a good target when using an application that is coded directly against the nCore API. When using higher-level APIs such as PKCS#11 or CNG, your application may benefit from increasing the job count further, e.g. to 60 or more outstanding jobs per HSM.

The `ncperfctest` utility supports performance measurements of a range of cryptographic operations with different job counts and client thread counts. You may find this useful to inform tuning of your application. Run `ncperfctest --help` to see the available options.

26.2. Client Configuration

If your application is coded directly against nCore, you have a choice of sending multiple jobs asynchronously from a single client connection to the hardserver, or having multiple threads each with their own client connection to the hardserver with a single job sent synchronously in each. You can use the `--threads` parameter to the `ncperfctest` utility to experiment with the performance impact of having more threads/connections with fewer jobs outstanding in each, or having fewer or just one thread/connection with more jobs outstanding in that connection.

When using higher-level APIs such as PKCS#11 or CNG, all cryptographic operations are synchronous, so larger numbers of threads must be used to increase the job count and make full use of HSM resources. These APIs automatically create a hardserver connection for each thread. If many HSMs are being used, a great many threads may be required to achieve best throughput. You can adjust the thread counts in the performance test tools for these APIs (e.g. `cksigtest` for PKCS#11 and `cngsoak` for CNG) to gauge how much concurrency is required for best throughput in your application.

26.3. Highly Multi-threaded Client Applications

If your application is highly multi-threaded, operating system defaults may not be optimal for best performance:

You may benefit from using a scalable memory allocator that is designed to be efficient in multi-threaded applications, examples include `tc_malloc`.

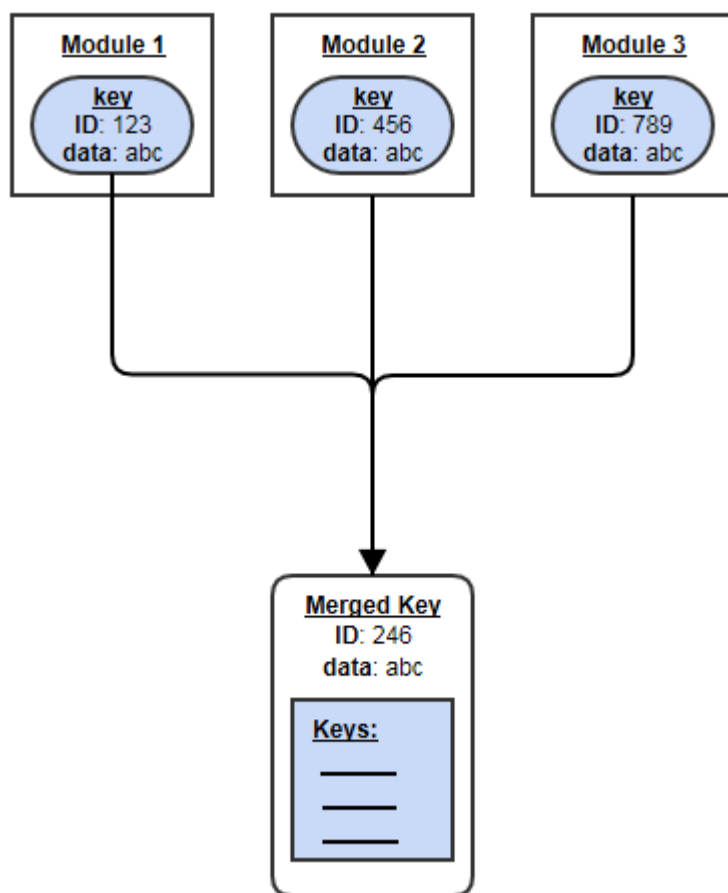
On some systems the default operating system scheduling algorithm is also not optimized for highly multi-threaded applications. A real-time scheduling algorithm such as the POSIX round-robin scheduler may yield noticeable performance improvements for your application.

26.4. File Descriptor Limits

On Linux systems, large numbers of threads each with their own hardserver connection will require your application to make use of large numbers of file descriptors. It may be necessary to increase the file descriptor limit for your application. This can be done using `ulimit -n NewLimit` on most systems, but you may need to increase system-wide hard limits first.

27. Merged Keys Concept

A merged key is a level of abstraction higher than normal keys. It holds an internal list of normal key IDs, each associated with its corresponding module. When a command to the hardserver specifies a MergedKey ID instead of a normal (single) key ID, the hardserver chooses an HSM and corresponding single key ID from the list in the Merged Key. See diagram below. Which module is chosen may depend on multiple factors, including load sharing settings in the hardserver config.



Benefits of MergedKeys:

- A client need hold only a single M_KeyID reference instead of one for each HSM.
- That ID remains usable even while the key's actual IDs on HSMs can fluctuate.
- The hardserver can use heuristics to choose the most appropriate HSM (e.g. the least heavily loaded one).
- If some HSMs become unavailable, the hardserver uses the remaining ones automatically.
 - A MergedKey can be updated, removing its entry for a defunct HSM and corre-

sponding single-key ID.

- New HSMs can be added: if a new HSM is made operational and added to the relevant security world, then
 - the key can be loaded onto that HSM, thus creating a new single-key ID for that key on that HSM, and then
 - the new (Key ID, HSM) pair can be added to the existing Merged Key.

28. Product returns

If you need to return your nShield Connect product, contact Support for instructions:

<https://nshieldsupport.entrust.com>.