nShield PQC Option Pack

nShield PQC Option Pack v1.2.1 User Guide

25 March 2025

© Entrust Corporation. All rights reserved.

Table of Contents

1. Introduction . . . 1
2. Software Prerequisites . . . 2
    2.1. Security World software . . . 2
    2.2. CodeSafe . . . 3
    2.3. HSM Firmware . . . 3
    2.4. Third party software . . . 4
    2.5. Remove existing versions of the nShield PQC Option Pack . . . 4
3. Installation . . . 5
    3.1. nShield XC installation . . . 5
    3.2. nShield 5 installation . . . 6
        3.2.1. nShield 5 HSM installation . . . 6
        3.2.2. nShield 5 client installation . . . 8
        3.2.3. nShield 5 client SSH configuration . . . 8
4. Uninstallation . . . 10
5. Supported Post-Quantum Algorithms . . . 11
    5.1. Digital Signature Algorithms . . . 11
    5.2. Key Encapsulation Mechanisms . . . 12
6. Application Programming Interface . . . 13
    6.1. Integration . . . 13
    6.2. HostCommands . . . 13
        6.2.1. Constants . . . 13
        6.2.2. Methods . . . 16
7. Examples . . . 28
    7.1. Installation with no pre-built examples . . . 28
    7.2. Installations with pre-built examples . . . 28

1. Introduction

The nShield Post-Quantum Cryptography (PQC) Option Pack provides users of Security World Software with the ability to generate and use keys with public-key cryptographic algorithms selected by NIST as part of the Post-Quantum Cryptography standardisation process.

The nShield PQC Option Pack is installed on top of your Security World Software, allowing you to use your existing keys and algorithms alongside post-quantum algorithms.

> The nShield PQC Option Pack provides the user with the opportunity to experiment with the use of PQC digital signature algorithms and signatures but at this moment the underlying Security World protection mechanisms still use classical (non post-quantum) crypto. It should not be used in an environment where a full post-quantum resistance security solution is required.

2. Software Prerequisites

Before you install the nShield PQC Option Pack:

- Refer to the latest Release Notes at https://nshieldsupport.entrust.com/hc/en-us/sections/360001115837-Release-Notes for hardware and software compatibility, and known and fixed issues.
- Check you have the Security World software installed, and a working Security World configured. See Security World software.
- Check you have the CodeSafe software installed. See CodeSafe.
- Check that you have a suitable firmware version on your nShield HSM. See HSM Firmware.
- Remove any previous installations of the nShield PQC Option Pack. See Remove existing versions of the nShield PQC Option Pack.
- Check that you have a usable OCS, and that it is present.
- Check that you have the required third party software installed. See Third party software.

This release of nShield PQC Option Pack is compatible with Linux only. See the *nShield PQC Option Pack Release Notes* for the list of compatible operating systems and versions.

2.1. Security World software

The nShield PQC Option Pack requires nShield Security World software to be installed, and a working Security World to be configured. To confirm that there is a usable Security World, use the `nfkminfo` command:

```
nfkminfo
```

If the Security World is usable then the state line in the `nfkminfo` output shows `Usable`.

```
World
    generation 2
    state 0x37270008 Initialised Usable Recovery !PINRecovery !ExistingClient RTC NVRAM FTO AlwaysUseStrongPrimes
!DisablePKCS1Padding !PpStrengthCheck !AuditLogging SEEDebug
...
```

Additionally, your hardserver should have appropriately configured `priv_port` and `nonpriv_port` settings.

```
[server_startup]
nonpriv_port=9000
```

```
priv_port=9001
```

For further information on installing Security World software and creating a Security World, see the *User Guide* shipped with your nShield Security World software.

See the *nShield PQC Option Pack Release Notes* for supported Security World Software versions.

2.2. CodeSafe

The requirements for nShield PQC Option Pack remote and local HSM installations are different.

- nShield PQC Option Pack remote installations require only Security World software.
- Local HSM installations require nShield CodeSafe software to be installed for building examples with the nShield PQC Option Pack. To confirm that CodeSafe is installed, use the `elftool` command:

```
elftool --version
```

If CodeSafe is installed, a message similar to the following will be printed in your terminal:

```
elftool, nshield (12.70.4-265-4efba9d)
```

See the *nShield PQC Option Pack Release Notes* for supported CodeSafe versions.

2.3. HSM Firmware

The nShield PQC Option Pack requires a supported version of the nShield HSM firmware to be installed. To confirm the installed version, use the `enquiry` command.

```
Module #1:
...
 version              13.4.3
 speed index          20000
 rec. queue           120..250
 level one flags      Hardware HasTokens SupportsCommandState SupportsHotReset
 version string       13.4.3-338-6c66aa0d
 checked in           00000000649dc17c Thu Jun 29 13:38:04 2023
 level two flags      none
...
```

In this example, the installed firmware version is `13.4.3`. See the *nShield PQC Option Pack Release Notes* for supported firmware versions.

In addition, for nShield XC, you must enable the Unrestricted SEE feature. This is indicated by the presence of `SEE Activation (EU+10)` in the enabled features seen in FET.

For nShield 5, you must enable the SEE Activation feature. This is indicated by the presence of `SEE Activation, Codesafe 5` or `SEE Activation (Codesafe 5)` in the enabled features seen in FET.

2.4. Third party software

The nShield PQC Option Pack requires following third party software to be installed:

- CMake 3.13.0 (or higher).
- GCC version 4.8.5 (or higher).
- GNU Make version 3.82 (or higher).
- Java Development Kit (Java 8) version 1.8.0_362 (or higher).
- Apache Ant version 1.9.4 (or higher).

2.5. Remove existing versions of the nShield PQC Option Pack

Please see Uninstallation for uninstallation instructions.

3. Installation

> The nShield PQC Option Pack requires that an OCS is present and usable. Please refer to the appropriate user guide for your product for further information.

To install the nShield PQC Option Pack:

1. Sign in as a user with root privileges.
2. Open a terminal window and create a temporary directory to mount the nShield PQC Option Pack ISO to:

   ```
   sudo mkdir /mnt/pqsdk
   ```

3. Mount the nShield PQC Option Pack ISO to the temporary directory created above:

   ```
   sudo mount -o loop pqsdk-1.1.0.iso /mnt/pqsdk
   ```

4. Change to the root directory and extract the appropriate `.tar.gz` file in the mounted nShield PQC Option Pack ISO.

The following three installation files are provided in the nShield PQC Option Pack ISO:

- pqsdk-xc-1.1.0.tar.gz : nShield XC local HSM installation with SDK sources and builder, no pre-built binaries.
- pqsdk-n5-hsm-1.1.0.tar.gz : nShield 5 local HSM installation with SDK sources, builder and pre-built examples.
- pqsdk-n5-client-1.1.0.tar.gz : nShield 5 remote client installation (without a local HSM installed) with pre-built examples and sources.

An installation file can be extracted as follows:

This installs all files required by the nShield PQC Option Pack to `/opt/nfast`.

```
cd /
sudo tar -xzf /mnt/pqsdk/pqsdk-<VERSION>.tar.gz -C /
```

> Only install one of the nShield PQC Option Pack installation files at a time.

3.1. nShield XC installation

In the case of a Solo XC or a Connect XC installation, the nShield PQC Option Pack can now be installed using the following commands:

```
/opt/nfast/pqsdk/sbin/install_pqsdk_java_pkg
/opt/nfast/pqsdk/sbin/install_pqsdk_see_xc
```

The installation process can be customised for your needs. Use `--help` for more information.

> The nShield PQC Option Pack is built from source during the installation. This may take a short while.

> The nShield PQC Option Pack is provided as a source-code release so that you may experiment with it. Changes to the code require re-installa tion of the nShield PQC Option Pack. Repeat the last step to rebuild and reinstall the product.

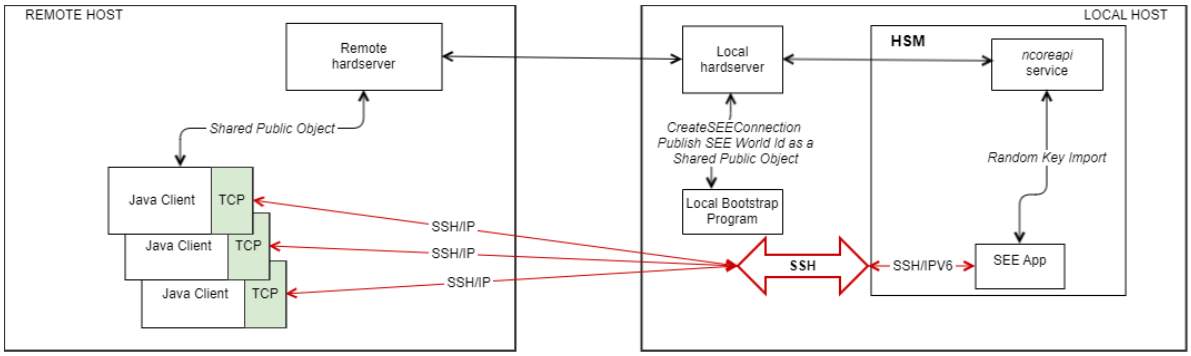
3.2. nShield 5 installation

There are two possible nShield 5 installations, the nShield 5 HSM installation and the nShield 5 remote installation. Each is described in detail in the following sections.

3.2.1. nShield 5 HSM installation

In this deployment, administration of the HSM is performed by the customer (for example, initializing the Security World on the HSM). The CodeSafe container is started from a local host and a local bootstrap program sends the `CreateSEEConnection` command and publishes the obtained worldid as an HSM object. After this, the client applications retrieves the published worldid and connects to the SEE machine using the IP address.

> For more information about the `CreateSEEConnection` command, see the Codesafe Developer Guide.



Loading and running the SEE Machine uses HSM Keys with Combined Certs. In the case of the nShield 5 HSM installation, the nShield PQC Option Pack can now sign the SEE Machine image, install and run it using the following command:

```
sudo /opt/nfast/pqsdk/sbin/install_pqsdk_see_n5 --install local --askeyname codesafe-ask --devkeyname
testdeveloperkey --devintcert devinter_certs.pem --bin /opt/nfast/pqsdk/bin/see/pqsdk-1.1.0.bin --sign true
```

The signing process will always look for the required certificates under: `/opt/nfast/pqsdk/n5_signing_certs/*`

A signed nShield PQC Option Pack cs5 image can only be run on the same machine it was generated and signed on.

As seen in the example above, in `/opt/nfast/pqsdk/n5_signing_certs/*` the intermediate and development certificates can be combined as a sin gle file. For example, `devinter_certs.pem`. Use the `--devintcert` option to specify the combined cert. `/opt/nfast/pqsdk/n5_signing_certs/dev inter_certs.pem`

The `--cs5` flag allows the re-running of a signed cs5 image again:

```
sudo /opt/nfast/pqsdk/sbin/install_pqsdk_see_n5 --install local --devintcert devinter_certs.pem --cs5
/opt/nfast/pqsdk/bin/see/pqsdk-1.1.0-signed.cs5
```

The HSM Keys created and used for signing can be specified using `--askeyname` and `--devkeyname`

```
/opt/nfast/pqsdk/n5_signing_certs$ nfkminfo -k
Key list - 2 keys
 AppName simple              Ident codesafe-ask
 AppName simple              Ident testdeveloperkey
```

If the default names for certificates and key files are different, then they can be specified with the following option flags when executing install_pqsdk_see_n5:

- `-askeyname`, `--askeyname [string]` : The ask key name for signing the image.
- `-askeyfile`, `--askeyfile [string]` : The ask key file for signing the image.
- `-devkeyfile`, `--devkeyfile [string]` : The developer signing key File used for signing.
- `-devcert`, `--devcert [string]` : The signed developer certificate PEM file, used for load ing or signing.
- `-intcert`, `--intcert [string]` : The signed intermediate certificate PEM file, used for loading image.

> Bootstrap is automatically started by the script (ie. `/opt/nfast/python3/bin/python3 ./src/pqsdk/source/python/n5/bootstrap.py --cs5file /opt/nfast/pqsdk/bin/see/<pqsdk.cs5> --uuid <uuid>`)

3.2.2. nShield 5 client installation

In this scenario, administration of the HSM and the CodeSafe container is performed by Entrust. The remote client application will need to have a Hardserver that is remotely connected to the Hardserver at the HSM location. The client installation will retrieve the published worldid via a shared public object and connect to the SEE Machine using the remote IP address to the SEE machine container.

For reference, the java examples could be run using the following command:

```
java -cp .:/opt/nfast/pqsdk/bin/java/pqsdk.jar com.entrust.pqsdk.examples.<example> <IP_ADDRESS> <PORT>
```

3.2.3. nShield 5 client SSH configuration

For security, an SSH tunnel is to be configured between the Java Client and the CodeSafe container running the SEE Machine. The container SSHD will only start when the state for that container has been set to enabled. By default, it is disabled. It can be enabled with the csadmin SSHD state enable subcommand. The enable subcommand will also return the port and address on which the SSHD is listening on. Example output is shown below:

```
csadmin sshd state enable -u 81c4f72f-586d-48c8-8020-7a6975159b37
F973-CCA4-370B        SUCCESS
SSHD PORT: 3006
LISTENING ADDRESS: fe80::53:89ff:fe92:3260
```

Once the container SSHD is enabled, started, and its keys are set, to forward connections to the container SSHD, the user will establish port forwarding on the host machine. Use of a secure tunnel over an open TCP connection to communicate with the container is recommended and so we use local port forwarding to enable SSH tunneling. This is done with the SSH local port forward command (can be IPv4 or IPv6 depending on the required network configuration). The basic usage is as follows:

```
ssh -i [PATH_TO_PRIV_KEY] -L [LOCAL_IP]:[LOCAL_PORT]:[CONTAINER_IP_ADDR%lxcbr0]:[CONTAINER_PORT] -f -N -p
[SSHD_PORT] launcher@[LISTENING_ADDRESS]%nshield0
```

Where:

- PRIV_KEY is the private key to the public client key that was set with the `setclient` command.
- LOCAL_PORT is the port on the local client where traffic to be forwarded will be sent.
- CONTAINER_IP_ADDR is the ip address of the container. This is the address returned when the container is started.
- CONTAINER_PORT is the port on the container that is listening for forwarded traffic which is set in the network-conf.json file as "ssh_tunnel" (note valid range 1024-65535).
- SSHD_PORT is the port the SSH daemon is listening on. This port is returned when SSHD is enabled and can also be found with the `csadmin sshd state get` command.
- LISTENING_ADDRESS is the address the host clients use to communicate with the HSM (on the hsm side). This address is returned when SSHD is enabled and can also be found with the `csadmin sshd state get` command.

As an example, in the local host, using an IPv4 LOCAL_IP and LOCAL_PORT:

```
ssh -i /home/.../test_key -L [10.194.147.71]:6000:[fe80::216:3eff:fe41:995%lxcbr0]:8888 -f -N -p 3000
launcher@fe80::53:89ff:fe92:3260%nshield0
```

And in the remote host:

```
java -cp /opt/nfast/pqsdk/bin/java/pqsdk-examples.jar com.entrust.pqsdk.examples.KEM 10.194.147.71 6000
```

Further information on enabling SSH port forwarding is available in the CodeSafe 5 Developer Guide.

4. Uninstallation

Before uninstalling the nShield PQC Option Pack, Entrust advises that you create a full back up of `/opt/nfast/pqsdk`. Removing the nShield PQC Option Pack will not affect your existing Security World, or your Security World Software installation. Once the nShield PQC Option Pack is removed, you will no longer be able to use keys created with nShield PQC Option Pack.

To uninstall the nShield PQC Option Pack, run the following command:

```
/opt/nfast/pqsdk/sbin/uninstall
```

The following options are available for the uninstall command (the full list is available with `--help`):

- `-p`, `--product [string]` : Use to specify "xc" or "n5" (default "xc")
- `-i`, `--install_type [string]` : The type of install to uninstall for n5 only. "local" or "remote" (default "remote").

> Uninstall by default runs for the nShield Solo XC. Use `n5` with the `-p` option as well as the appropriate `-i` option when you are uninstalling for that platform.

5. Supported Post-Quantum Algorithms

5.1. Digital Signature Algorithms

This version of the nShield PQC Option Pack supports the following signing algorithms:

Algorithm	Variants
Falcon	• Falcon-512 • Falcon-1204
CRYSTALS-Dilithium	• Dilithium2 • Dilithium2-AES • Dilithium3 • Dilithium3-AES • Dilithium5 • Dilithium5-AES
SPHINCS+-Haraka	• SPHINCS+-Haraka-128f-robust • SPHINCS+-Haraka-128f-simple • SPHINCS+-Haraka-128s-robust • SPHINCS+-Haraka-128s-simple • SPHINCS+-Haraka-192f-robust • SPHINCS+-Haraka-192f-simple • SPHINCS+-Haraka-192s-robust • SPHINCS+-Haraka-192s-simple • SPHINCS+-Haraka-256f-robust • SPHINCS+-Haraka-256f-simple • SPHINCS+-Haraka-256s-robust • SPHINCS+-Haraka-256s-simple

Algorithm	Variants
SPHINCS+-SHA256	• SPHINCS+-SHA256-128f-robust • SPHINCS+-SHA256-128f-simple • SPHINCS+-SHA256-128s-robust • SPHINCS+-SHA256-128s-simple • SPHINCS+-SHA256-192f-robust • SPHINCS+-SHA256-192f-simple • SPHINCS+-SHA256-192s-robust • SPHINCS+-SHA256-192s-simple • SPHINCS+-SHA256-256f-robust • SPHINCS+-SHA256-256f-simple • SPHINCS+-SHA256-256s-robust • SPHINCS+-SHA256-256s-simple
SPHINCS+-SHAKE256	• SPHINCS+-SHAKE256-128f-robust • SPHINCS+-SHAKE256-128f-simple • SPHINCS+-SHAKE256-128s-robust • SPHINCS+-SHAKE256-128s-simple • SPHINCS+-SHAKE256-192f-robust • SPHINCS+-SHAKE256-192f-simple • SPHINCS+-SHAKE256-192s-robust • SPHINCS+-SHAKE256-192s-simple • SPHINCS+-SHAKE256-256f-robust • SPHINCS+-SHAKE256-256f-simple • SPHINCS+-SHAKE256-256s-robust • SPHINCS+-SHAKE256-256s-simple

5.2. Key Encapsulation Mechanisms

Algorithm	Variants
CRYSTALS-Kyber	• Kyber512 • Kyber512-90s • Kyber768 • Kyber768-90s • Kyber1024 • Kyber1024-90s

6. Application Programming Interface

The nShield PQC Option Pack can build and load a CodeSafe SEE machines onto nShield HSMs:

- Legacy CodeSafe SEE machines for nShield Solo XC HSMs
- CodeSafe 5 SEE machines for nShield 5s HSMs.

During this process, a Java binary is produced that enables you to interface with the SEE machine.

See Examples for example programs that show how to use the nShield PQC Option Pack API.

6.1. Integration

To integrate the nShield PQC Option Pack within your own Java code, please include `/opt/nfast/pqsdk/bin/java/pqsdk.jar` in your classpath. `pqsdk.jar` exposes a Java API for you to integrate with. The source files describing the API exist at `/opt/nfast/pqsdk/src/pqsdk/source/java`.

You should also ensure that the `nCipherKM.jar` is installed.

6.2. HostCommands

The `HostCommands` class provides you with a Java interface into the SEE machine. You may then use the exposed API to perform key generation, signing, and verification activities.

```
HostCommands hc = new HostCommands();
```

6.2.1. Constants

Modifier and Type	Name	Value
public static final int	algorithm_dilithium_2	1
public static final int	algorithm_dilithium_3	2
public static final int	algorithm_dilithium_4	3
public static final int	algorithm_falcon_512	4
public static final int	algorithm_falcon_1024	5

Modifier and Type	Name	Value
public static final int	algorithm_sphincs_haraka_128f_robust	6
public static final int	algorithm_sphincs_haraka_128f_simple	7
public static final int	algorithm_sphincs_haraka_128s_robust	8
public static final int	algorithm_sphincs_haraka_128s_simple	9
public static final int	algorithm_sphincs_haraka_192f_robust	10
public static final int	algorithm_sphincs_haraka_192f_robust	11
public static final int	algorithm_sphincs_haraka_192s_robust	12
public static final int	algorithm_sphincs_haraka_192s_simple	13
public static final int	algorithm_sphincs_haraka_256f_robust	14
public static final int	algorithm_sphincs_haraka_256f_simple	15
public static final int	algorithm_sphincs_haraka_256s_robust	16
public static final int	algorithm_sphincs_haraka_256s_simple	17
public static final int	algorithm_sphincs_sha256_128f_robust	18
public static final int	algorithm_sphincs_sha256_128f_simple	19
public static final int	algorithm_sphincs_sha256_128s_robust	20
public static final int	algorithm_sphincs_sha256_128s_simple	21
public static final int	algorithm_sphincs_sha256_192f_robust	22
public static final int	algorithm_sphincs_sha256_192f_simple	23

Modifier and Type	Name	Value
public static final int	algorithm_sphincs_sha256_192s_robust	24
public static final int	algorithm_sphincs_sha256_192s_simple	25
public static final int	algorithm_sphincs_sha256_256f_robust	26
public static final int	algorithm_sphincs_sha256_256f_simple	27
public static final int	algorithm_sphincs_sha256_256s_robust	28
public static final int	algorithm_sphincs_sha256_256s_simple	29
public static final int	algorithm_sphincs_shake256_128f_robust	30
public static final int	algorithm_sphincs_shake256_128f_simple	31
public static final int	algorithm_sphincs_shake256_128s_robust	32
public static final int	algorithm_sphincs_shake256_128s_simple	33
public static final int	algorithm_sphincs_shake256_192f_robust	34
public static final int	algorithm_sphincs_shake256_192f_simple	35
public static final int	algorithm_sphincs_shake256_192s_robust	36
public static final int	algorithm_sphincs_shake256_192s_simple	37
public static final int	algorithm_sphincs_shake256_256f_robust	38
public static final int	algorithm_sphincs_shake256_256f_simple	39
public static final int	algorithm_sphincs_shake256_256s_robust	40
public static final int	algorithm_sphincs_shake256_256s_simple	41

Modifier and Type	Name	Value
public static final int	algorithm_dilithium_5	42
public static final int	algorithm_dilithium_2_aes	43
public static final int	algorithm_dilithium_3_aes	44
public static final int	algorithm_dilithium_5_aes	45
public static final int	algorithm_kyber_512	46
public static final int	algorithm_kyber_768	47
public static final int	algorithm_kyber_1024	48
public static final int	algorithm_kyber_512_90s	49
public static final int	algorithm_kyber_768_90s	50
public static final int	algorithm_kyber_1024_90s	51

6.2.2. Methods

6.2.2.1. init(String)

```
protected void init (String published)
```

Initialises the HostCommands object.

Throws

`RuntimeException` - if one of the following conditions is met:

- The nShield PQC Option Pack SEE machine is not installed, published, or running.
- The Security World is not usable.

6.2.2.2. algorithm_toString(long)

```
public static String algorithm_toString(long value)
```

Gets the name of an algorithm, given a constant value.

Parameters

`value` - an algorithm id constant.

Returns

A `String` value describing the algorithm, or `UNKNOWN`.

6.2.2.3. getVersion()

```
public getVersionResponse getVersion()
```

Gets the version of the SEE machine interface.

Returns

A `getVersionResponse` object encapsulating the SEE machine version.

Throws

`NFException` - if one of the following conditions is met:

- The nShield PQC Option Pack SEE machine is not installed, published, or running.
- The Security World is not usable.

6.2.2.4. list()

```
public listResponse list()
```

List all PQC keys

Returns

A `listResponse` object encapsulating a list of key hashes.

Throws

`NFException` - if one of the following conditions is met:

- The nShield PQC Option Pack SEE machine is not installed, published, or running.
- The Security World is not usable.
- The nShield HSM ran out of memory.

6.2.2.5. getPublic(String)

```
public getPublicResponse getPublic(String alias)
```

Get the public half of a PQC key-pair

Parameters

`alias` - the name of the key

Returns

A `getPublicResponse` object encapsulating the key

Throws

`NFException` - if one of the following conditions is met:

- The nShield PQC Option Pack SEE machine is not installed, published, or running.
- The Security World is not usable.
- The nShield HSM ran out of memory.
- The key did not exist.

6.2.2.6. generate(String, int, boolean)

```
public generateResponse generate(String alias, int algorithm, boolean overwrite)
```

Generate a key pair

Parameters

`alias` - the alias under which to store the key pair.
`algorithm` - the PQC algorithm (one of the algorithms in `HostCommands.algorithm_*`).
`overwrite` - if true, and `alias` exists, overwrites the existing key pair.

Returns

A `generateResponse` object encapsulating the alias and hash of the generated key pair.

Throws

`NFException` - if one of the following conditions is met:

- The nShield PQC Option Pack SEE machine is not installed, published, or running.
- The Security World is not usable.
- The nShield HSM ran out of memory.

6.2.2.7. generate(String, int)

```
public generateResponse generate(String alias, int algorithm)
```

Generate a key pair

Parameters

`alias` - the alias under which to store the key pair.
`algorithm` - the PQC algorithm (one of the algorithms in `HostCommands.algorithm_*`).

Returns

A `generateResponse` object encapsulating the alias and hash of the generated key pair.

Throws

`NFException` - if one of the following conditions is met:

- The nShield PQC Option Pack SEE machine is not installed, published, or running.
- The Security World is not usable.
- The nShield HSM ran out of memory.

6.2.2.8. importKeypair(String, int, byte[], byte[], boolean)

```
public importKeypairResponse importKeypair(String alias, int algorithm, byte[] publicKey, byte[] privateKey,
boolean overwrite)
```

Import a key pair

Parameters

`alias` - the alias under which to store the key pair.
`algorithm` - the PQC algorithm (one of the algorithms in `HostCommands.algorithm_*`).
`publicKey` - the public key represented as a byte array.
`privateKey` - the private key represented as a byte array.
`overwrite` - if true, and `alias` exists, overwrites the existing key pair.

Returns

A `importKeypairResponse` object encapsulating the alias and hash of the imported key pair.

Throws

`NFException` - if one of the following conditions is met:

- The nShield PQC Option Pack SEE machine is not installed, published, or running.
- The Security World is not usable.
- The nShield HSM ran out of memory.

6.2.2.9. importKeypair(String, int, byte[], byte[])

```
public importKeypairResponse importKeypair(String alias, int algorithm, byte[] publicKey, byte[] privateKey)
```

Import a key pair

Parameters

`alias` - the alias under which to store the key pair.
`algorithm` - the PQC algorithm (one of the algorithms in `HostCommands.algorithm_*`).
`publicKey` - the public key represented as a byte array.
`privateKey` - the private key represented as a byte array.

Returns

A `importKeypairResponse` object encapsulating the alias and hash of the imported key pair.

Throws

`NFException` - if one of the following conditions is met:

- The nShield PQC Option Pack SEE machine is not installed, published, or running.
- The Security World is not usable.
- The nShield HSM ran out of memory.

6.2.2.10. sign(String, byte[], byte[])

```
public signResponse sign(String alias, byte[] keyhash, byte[] message)
```

Sign a message

Parameters

`alias` - (optional) the name of the key with which to sign.
`keyhash` - the hash of the key with which to sign.
`message` - the message to sign.

Returns

A `signResponse` object encapsulating the message signature.

Throws

`IOException` - if the key could not be read.
`NFException` - if one of the following conditions is met:

- The nShield PQC Option Pack SEE machine is not installed, published, or running.
- The Security World is not usable.

6.2.2.11. sign(String, byte[])

```
public signResponse sign(String alias, byte[] message)
```

Sign a message

Parameters

`alias` - the name of the key with which to sign.
`message` - the message to sign.

Returns

A `signResponse` object encapsulating the message signature.

Throws

`IOException` - if the key could not be read.
`NFException` - if one of the following conditions is met:

- The nShield PQC Option Pack SEE machine is not installed, published, or running.
- The Security World is not usable.

6.2.2.12. sign(byte[], byte[])

```
public signResponse sign(byte[] keyhash, byte[] message)
```

Sign a message

Parameters

`keyhash` - the hash of the key with which to sign.
`message` - the message to sign.

Returns

A `signResponse` object encapsulating the message signature.

Throws

`IOException` - if the key could not be read.
`NFException` - if one of the following conditions is met:

- The nShield PQC Option Pack SEE machine is not installed, published, or running.
- The Security World is not usable.

6.2.2.13. verify(String, byte[], byte[], byte[])

```
public verifyResponse verify(String alias, byte[] keyhash, byte[] message, byte[] signature)
```

Verify the signature of message

Parameters

`alias` - (optional) the name of the key to verify with.
`keyhash` - the hash of the key to verify with.
`message` - the message to verify.
`signature` - the message signature.

Returns

A `verifyResponse` object encapsulating a boolean value indicating the operation status.

Throws

`IOException` - if the key could not be read.
`NFException` - if one of the following conditions is met:

- The nShield PQC Option Pack SEE machine is not installed, published, or running.
- The Security World is not usable.

6.2.2.14. verify(String, byte[], byte[])

```
public verifyResponse verify(String keyhash, byte[] message, byte[] signature)
```

Verify the signature of message

Parameters

`alias` - the name of the key to verify with.
`message` - the message to verify.
`signature` - the message signature.

Returns

A `verifyResponse` object encapsulating a boolean value indicating the operation status.

Throws

`IOException` - if the key could not be read.
`NFException` - if one of the following conditions is met:

- The nShield PQC Option Pack SEE machine is not installed, published, or running.
- The Security World is not usable.

6.2.2.15. verify(byte[], byte[], byte[])

```
public verifyResponse verify(byte[] keyhash, byte[] message, byte[] signature)
```

Verify the signature of message

Parameters

`keyhash` - the hash of the key to verify with.
`message` - the message to verify.
`signature` - the message signature.

Returns

A `verifyResponse` object encapsulating a boolean value indicating the operation status.

Throws

`IOException` - if the key could not be read.
`NFException` - if one of the following conditions is met:

- The nShield PQC Option Pack SEE machine is not installed, published, or running.
- The Security World is not usable.

6.2.2.16. verifyImmediate(byte[], int, byte[], byte[])

```
public verifyResponse verifyImmediate(byte[] publicKey, int algorithm, byte[] message, byte[] signature)
```

Verify the signature of message using immediate values

Parameters

`publicKey` - the public key to verify with.
`algorithm` - the signature algorithm.
`message` - the message to verify.
`signature` - the message signature.

Returns

A `verifyResponse` object encapsulating a boolean value indicating the operation status.

Throws

`NFException` - if one of the following conditions is met:

- The nShield PQC Option Pack SEE machine is not installed, published, or running.
- The Security World is not usable.

6.2.2.17. encapsulate(String, byte[])

```
public encapsulateResponse encapsulate(String alias, byte[] keyhash)
```

Encapsulate a shared secret using the public half of a key pair

Parameters

`alias` - the name of the public key to encapsulate with.
`keyhash` - the hash of the public key to encapsulate with. Only used if the `alias` is null.

Returns

An `encapsulateResponse` object encapsulating a shared secret in plaintext and a ciphertext (encapsulation) of this shared secret.

Throws

`RuntimeException` - if the alias passed in does not belong to an existing key.
`NFException` - if one of the following conditions is met:

- The nShield PQC Option Pack SEE machine is not installed, published, or running.
- The Security World is not usable.

6.2.2.18. encapsulate(String)

```
public encapsulateResponse encapsulate(String alias)
```

Encapsulate a shared secret using the public half of a key pair

Parameters

`alias` - the name of the public key to encapsulate with.

Returns

An `encapsulateResponse` object encapsulating a shared secret in plaintext and a ciphertext (encapsulation) of this shared secret.

Throws

`RuntimeException` - if the alias passed in does not belong to an existing key.
`NFException` - if one of the following conditions is met:

- The nShield PQC Option Pack SEE machine is not installed, published, or running.
- The Security World is not usable.

6.2.2.19. encapsulate(byte[])

```
public encapsulateResponse encapsulate(byte[] keyhash)
```

Encapsulate a shared secret using the public half of a key pair

Parameters

`keyhash` - the hash of the public key to encapsulate with.

Returns

An `encapsulateResponse` object encapsulating a shared secret in plaintext and a ciphertext (encapsulation) of this shared secret.

Throws

`RuntimeException` - if the alias passed in does not belong to an existing key.
`NFException` - if one of the following conditions is met:

- The nShield PQC Option Pack SEE machine is not installed, published, or running.
- The Security World is not usable.

6.2.2.20. encapsulateImmediate(byte[], int)

```
public encapsulateImmediateResponse encapsulateImmediate(byte[] publicKey, int algorithm)
```

Encapsulate a shared secret using a public key

Parameters

`publicKey` - the public key to encapsulate with.
`algorithm` - the encapsulation algorithm.

Returns

An `encapsulateResponse` object encapsulating a shared secret in plaintext and a ciphertext (encapsulation) of this shared secret.

Throws

`NFException` - if one of the following conditions is met:

- The nShield PQC Option Pack SEE machine is not installed, published, or running.
- The Security World is not usable.

6.2.2.21. decapsulate(String, byte[], byte[])

```
public decapsulateResponse decapsulate(String alias, byte[] keyhash, byte[] ciphertext)
```

Decapsulate a shared secret using the private half of a key pair

Parameters

`alias` - the name of the private key to decapsulate with.
`keyhash` - the hash of the private key to decapsulate with. Only used if the alias is null.
`ciphertext` - the encapsulated secret to decapsulate.

Returns

A `decapsulateResponse` object encapsulating the shared secret.

Throws

`RuntimeException` - if the alias passed in does not belong to an existing key.
`NFException` - if one of the following conditions is met:

- The nShield PQC Option Pack SEE machine is not installed, published, or running.
- The Security World is not usable.

6.2.2.22. decapsulate(String, byte[])

```
public decapsulateResponse decapsulate(String alias, byte[] ciphertext)
```

Decapsulate a shared secret using the private half of a key pair

Parameters

`alias` - the name of the private key to decapsulate with.
`ciphertext` - the encapsulated secret to decapsulate.

Returns

A `decapsulateResponse` object encapsulating the shared secret.

Throws

`RuntimeException` - if the alias passed in does not belong to an existing key.
`NFException` - if one of the following conditions is met:

- The nShield PQC Option Pack SEE machine is not installed, published, or running.
- The Security World is not usable.

6.2.2.23. decapsulate(byte[], byte[])

```
public decapsulateResponse decapsulate(byte[] keyhash, byte[] ciphertext)
```

Decapsulate a shared secret using the private half of a key pair

Parameters

`keyhash` - the hash of the private key to decapsulate with.
`ciphertext` - the encapsulated secret to decapsulate.

Returns

A `decapsulateResponse` object encapsulating the shared secret.

Throws

`RuntimeException` - if the alias passed in does not belong to an existing key.
`NFException` - if one of the following conditions is met:

- The nShield PQC Option Pack SEE machine is not installed, published, or running.
- The Security World is not usable.

7. Examples

There are various example programs included in the `examples` directory. These examples use the `pqsdk.jar` file to interface with the SEE machine and demonstrate how to use the nShield PQC Option Pack API.

7.1. Installation with no pre-built examples

In an installation with no pre-built examples, the example programs need to be compiled before they can be run. For example, to execute the `SignVerify` example for a Solo XC run the following commands:

```
cd /opt/nfast/pqsdk/examples/xc
javac -d . -cp /opt/nfast/java/classes/nCipherKM.jar:/opt/nfast/pqsdk/bin/java/pqsdk.jar SignVerify.java
jar -cf examples.jar com/entrust/pqsdk/examples/SignVerify.class
java -cp examples:/opt/nfast/pqsdk/bin/java/pqsdk.jar com.entrust.pqsdk.examples.SignVerify
```

You can use the source code as the basis for your own programs, or modify it to your needs.

7.2. Installations with pre-built examples

In an installation with pre-built examples, the example programs are compiled into pqsdk-examples.jar. For example, to execute the `ImportKeypair` example under n5 run the following commands:

```
cd /opt/nfast/pqsdk/examples/n5
java -cp /opt/nfast/pqsdk/bin/java/pqsdk-examples.jar com.entrust.pqsdk.examples.ImportKeypair
fe80::216:3eff:fe6c:1bb%nshield0 8888
```