Cloud Integration Option Pack

CIOP v2.2.1 User Guide

09 April 2024

Table of Contents

1. Introduction

The Entrust nShield Cloud Integration Option Pack (CIOP) provides users of cloud services with the ability to generate keys in their own environment and export them for use in the cloud while having confidence that:

- Their key has been generated securely using a strong entropy source.
- The long term storage of their key is protected by a FIPS-certified Hardware Security Module (HSM).

The following cloud services are supported:

- Amazon Web Services (AWS)
- Google Compute Engine
- Google Cloud Key Management (Google KMS)
- Microsoft Azure
- Salesforce

The generated encryption key is passed to a cloud service provider. Therefore, we recommend that both the URL and the server certificate of the provider are verified as current and valid.

2. Software Installation

Before you install CIOP:

- See 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.
- Remove any previous installations of CIOP.
- Check you have the Security World software (v12.60+) installed, and a working Security World configured. For more information on creating a Security World, see the *User Guide* for your nShield HSM.

Please note that CIOP has different Security World version requirements depending on which cloud service provider you are attempting to export keys for. Please refer to the *Release Notes* for further details.

2.1. Install on Windows

To install the Cloud Integration Option Pack on Windows:

1. Download the `CIOP-<version>.zip` file.

   For example, `CIOP-2.2.1.zip`.

2. Extract the `CIOP-<version>.zip`.
3. Start a command prompt with `Run as administrator`.
4. Change to the directory from step 2.
5. In the command prompt, install using the provided batch script `install.bat`. This script uses the nShield Python `pip` tool.

   ```
   install.bat
   ```

2.2. Install on Linux

To install the Cloud Integration Option Pack on Linux:

1. Download the `CIOP-<version>.zip` file.

   For example, `CIOP-2.2.1.zip`.

2. Extract the `CIOP-<version>.zip`.

3. Change to the directory from step 2.
4. In the command prompt, install using the provided shell script `install.sh`. This script uses the nShield Python `pip` tool.

```
sudo ./install.sh
```

2.3. Uninstall CIOP

To uninstall CIOP on Windows, run:

```
"%NFAST_HOME%\python\python.exe" -m pip uninstall nshield-cssadmin
```

To uninstall CIOP on Linux, run:

```
"%NFAST_HOME%\python\bin\python" -m pip uninstall nshield-cssadmin
```

3. Checking the Installation

Confirm that the Security World and module are usable with `nfkminfo`. Specifically, ensure that the World state shows `Usable` and the desired `Module #n` state (where `n` is the number of the module) shows `0x2 Usable`.

You are now able to generate key material in the HSM and send it securely to a cloud service provider

4. cssadmin

`cssadmin` creates a key in a Security World. This key is first wrapped by wrapping key, and only then exported for use outside of the Security World.

`cssadmin` requires at least three parameters to be specified:

provider The name of the provider.

keyName The name of the Security World key (an existing key can be specified, however, it must have first been created by `cssadmin`).

wrapKey The name of the cloud service provider's wrapping key.

For AWS, Google Compute Engine and Salesforce, the following parameter must also be specified:

wrapAlg The wrapping algorithm.

For Microsoft Azure, the following parameter must also be specified:

—azure-kek The key identifier of the Microsoft Key Exchange Key.

If multiple HSMs exist in the Security World you can, optionally, specify a specific HSM to be used.

For AWS, Google Compute Engine and Salesforce, the generated key is a 256-bit AES key. For Microsoft Azure and Google Cloud Key Management (Google KMS), you can choose from a list of available key types using a command line argument.

4.1. Run cssadmin

To run cssadmin on Windows:

```
"%NFAST_HOME%\python\python.exe" -m cssadmin
```

To run cssadmin on Linux:

```
/opt/nfast/python/bin/python -m cssadmin
```

4.2. Usage and options

```
usage: cssadmin.py [-h] [--version] [-m MODULE] [-O CARDSET | -S SOFTCARD | -M] [-a AZUREKEK] [-k KEYTYPE]
provider keyName wrapKey [wrapAlg]
```

4.3. Help

Export a key from a Security World to be used outside, by a provider.

If the key does not exist, a new one is generated. By default, the key is module protected.

```
positional arguments:

provider    The provider for which to export the key.
            One of ['aws', 'google-compute-engine',
            'google-cloud-key-management', 'microsoft-azure', 'salesforce']
keyName     The name of the key to export/generate
wrapKey     The filename of the key to use for wrapping (DER
            encoded)
wrapAlg     The wrapping algorithm (required for aws, google-compute-engine or salesforce).
            One of ['RSAES_OAEP_SHA_256', 'RSAES_OAEP_SHA_1', 'RSAES_PKCS1_V1_5']
```

```
optional arguments:

-h, --help                  show this help message and exit
--version                   show program's version number and exit
-m MODULE, --module         MODULE
                            Specify a module to use
-O CARDSET, --ocs           CARDSET
                            Select OCS protection
-S SOFTCARD, --softcard     SOFTCARD
                            Select softcard protection
 -M, --module-protection    Select module protection (default)
```

```
Microsoft Azure:

-a AZUREKEK, --azure-kek AZUREKEK   The full URL key identifier of the Microsoft Azure wrapping key.
                                    Typically a URI i.e. https://myvault.vault.azure.net/keys/mykey/version
```

```
Microsoft Azure and Google KMS:

-k KEYTYPE, --key-type KEYTYPE      The type of the target key.
                                    One of ['AES-256', 'EC-NISTP256', 'EC-NISTP384', 'EC-NISTP521', EC-
SECP256K1', 'RSA-2048', 'RSA-3072', 'RSA-4096'] Will default to RSA-2048 if not supplied
```

The key type option represents the possible key types for both Microsoft Azure and Google KMS as there are common types. An error message will be printed if the type is not supported by the specified provider.

> Microsoft Azure supports: 'RSA-2048', 'RSA-3072', 'RSA-4096', 'EC-NISTP256', EC-NISTP384', 'EC-NISTP521', 'EC-SECP256K1'
>
> Google Cloud Key Management supports: 'RSA-2048', 'RSA-3072', 'RSA-4096', 'EC-NISTP256', 'EC-NISTP384', 'AES-256'

5. Integrate with Amazon Web Services

5.1. Access the Key Management System in the AWS Management Console

Connect to the Key Management Service (KMS) using the AWS Management Console, and select your AWS region.

The steps below can be carried out either on the AWS Management Console or using the AWS KMS API. See the AWS KMS documentation for details of how to use these tools. We provide instructions here for using the console. Instructions for using the API can be found in the AWS documentation.

5.2. Create a customer master key in the AWS Management Console

1. In the navigation pane in KMS, select **Customer Managed Keys**.
2. Select **Create key**.
3. Select **Symmetric**.
4. Expand **Advanced Options**.
5. Specify **External** for **Key Material Origin**.
6. Confirm that you understand the implications of using an imported key.
7. Select **Next**.
8. Create an Alias, Description, and Tags:
    a. Specify an alias for the key and, optionally, a description and tags.
    b. Select **Next**.
9. Define Key Administrative Permissions:
    a. Specify the Identity and Access Management (IAM) users who can administer the key, and decide whether key administrators are able to delete the key.
    b. Select **Next**.
10. In the **This Account** section, define Key Usage Permissions:
    a. Specify the IAM users who can use the key.
    b. Select **Next**.
11. Review Key Policy:
    a. A preview of your key policy should be displayed.

b. Select **Finish**.

If the operation succeeds, you have created a CMK with no key material. The following mes sage should be displayed:

Your customer master key (CMK) was created with alias *<key name>* and key ID *<key id>*. To use this CMK, you must import key material.

5.3. Download a wrapping key and import token in the AWS Management Console

If you have just completed the instructions in Create a customer master key in the AWS Management Console, you should be on the Download wrapping key and import token screen.

Otherwise complete these steps:

1. Connect to the AWS Management Console, and select your AWS Region.
2. In the navigation pane in KMS, select **Customer Managed Keys**.
3. Select the alias or key ID of the CMK that is pending import.
4. Expand the **Cryptographic configuration** section, and view its values.
5. Expand the **Key Material** section, and then select **Download wrapping key and import token**.

You can continue to download the wrapping key:

1. Select the wrapping algorithm to use.

   We recommend selecting the strongest hashing algorithm supported.

2. Select **Download wrapping key and import token**, then save the file (`ImportParameters.zip`).
3. Unzip `ImportParameters`, and you should find the following files:
   - A README text file.
   - The wrapping key file.
   - The import token file.

5.4. Use cssadmin to generate, wrap, and export a key

Using the downloaded wrapping key file, call `cssadmin` as follows:

```
cssadmin aws <key name> <wrapping key> <wrapping algorithm>
```

Running this command creates a file called <key name>-wrapped.

5.5. Upload the wrapped key material

If you have just completed the instructions in Download a wrapping key and import token in the AWS Management Console, you should be on the Upload your wrapped key material screen.

Otherwise complete these steps:

1. Connect to the AWS Management Console and select your AWS Region.
2. In the navigation pane in KMS, select **Customer Managed Keys**.
3. Select the key ID or alias of the CMK for which you downloaded the public key and import token.
4. Expand the **Cryptographic configuration** section, and view its values.
5. Expand the **Key Material** section, and then select **Upload key material**.

Now you can continue to upload the wrapped key.

1. In the **Encrypted key material and import token** section, under **Wrapped key material**, select **Choose file**.

   Upload the file that contains your wrapped (encrypted) key material. This is the file that was produced in Use cssadmin to generate, wrap, and export a key.

2. In the **Encrypted key material and import token** section, under **Import token**, select **Choose file**.

   Upload the file that contains the import token that you downloaded.

3. In the **Expiration option** section, you can determine whether the key material expires.

   To set an expiration date and time, choose **Key material expires**, and use the calendar to select a date and time.

4. Select **Upload key material**.

The following message should be displayed:

```
Your key material was imported into the customer master key (CMK) with key ID <key id>. You can now use this CMK.
```

The AWS key is now ready for use.

At this point users have now extended trust to AWS in terms of use of the key and its destruction.

6. Integrate with Google Compute Engine

6.1. Access Google Compute Engine in the Google Cloud Platform Console

From **Products & services**, select **Compute Engine**.

6.2. Download the Google Compute Engine public key certificate

To protect the delivery of the customer-supplied encryption key, Google provide a public key in a certificate to wrap the generated key. The public key certificate is available at https://cloud-certs.storage.googleapis.com/google-cloud-csek-ingress.pem.

We recommend confirming that this certificate is valid before use. See https://cloud.google.com/compute/docs/disks/customer-supplied-encryption for additional information.

6.3. Use cssadmin to generate, wrap and export a key

Using the downloaded public key certificate, call `cssadmin` as follows:

```
cssadmin google-compute-engine <key name> <wrapping key> <wrapping algorithm>
```

At the time of writing, only RSAES_OAEP_SHA_1 is supported.

6.4. Encrypt a disk with the wrapped key material

1. From within Google Compute Engine, select **Disks**.
    a. Select **CREATE DISK**.
    b. Specify the details of the disk to be created.
2. Under **Encryption**:
    a. Select **Customer supplied**.
    b. Check **Wrapped key**.
    c. Paste the contents of `<key name>-wrapped`.
    d. Select **Create**.

When the disk is created, it should be shown as **ready for use**, and **Encryption** as **Customer** supplied. The Google Compute Engine key is now ready for use.

At this point users have now extended trust to Google in terms of use of the key and its destruction.

7. Integrate with Google Cloud Key Management

7.1. Access Google Cloud Key Management

Google Cloud Key Management can be accessed through the Google Cloud Platform or by using the Google Cloud SDK Shell Command-Line Interface (CLI). Follow the Google Cloud Key Management documentation to ensure that you use the appropriate authentication, and are communicating with a genuine Google Cloud Key Management server.

This operation requires the use of a Google KMS project with billing enabled. We recommend that you use the Google guides guide when carrying out the Google KMS operations because they might have been updated after the publication of this User Guide for the Cloud Integration Option Pack. See https://cloud.google.com/kms/docs/key-import and https://cloud.google.com/kms/docs/importing-a-key. Refer to the instructions for **Importing a manually-wrapped key** but note that the wrapping will be carried out using the `cssadmin` tool and not OpenSSL.

7.2. Create a target key and key ring

A Cloud KMS key is a container object that contains zero or more key versions. Each key version contains a cryptographic key.

When you import a key into Cloud KMS or Cloud HSM, the imported key becomes a new key version on an existing Cloud KMS or Cloud HSM key. In the rest of this topic, this key is called the target key. The target key must exist before you can import key material into it.

From the Google Cloud Console Web UI:

1. Go to the **Cryptographic Keys** page.
2. Click **Create key ring**.
    a. enter a name for the key ring in the **Key ring name** field.
    b. select a location from the **Location** dropdown see https://cloud.google.com/kms/docs/locations for more information.
    c. Click **Create** - opens detail page for key ring.
3. Click **Create key**.
    a. Select **Imported key**.

b. Enter name for the key in the **Key name** field.
c. Set **Protection level** to **HSM**.
d. Set **Purpose** to one of Symmetric encryption, Asymmetric signing, Asymmetric decrypt.
e. Set **Key type and Algorithm**.
f. Optionally set up rotation and labels.
g. Click **create**.

Alternatively use the Google Cloud SDK Shell.

```
gcloud kms keyrings create key-ring-name \
  --location location
```

```
gcloud kms keys create key-name \
  --location location \
  --keyring key-ring-name \
  --purpose purpose \
  --skip-initial-version-creation
```

7.3. Generate the Google KMS Wrapping Key

It is necessary to create an **Import Job**.

From the Google Cloud Console Web UI:

1. Go to the Cryptographic Keys page.
2. Click the name of the target key ring.
3. Click **Create import job**.
    a. Set the **Protection level** to either **Software** or **HSM**. Use the same protection level as you set for the target key.
    b. From the **Import method** dropdown, set the import method to either **3072 bit RSA** or **4096 bit RSA**.
    c. Click **Create**.

Alternatively use the Google Cloud SDK Shell.

```
gcloud kms import-jobs create import-job \
  --location location \
  --keyring key-ring-name \
  --import-method import-method \
  --protection-level protection-level
```

7.4. Download the Google KMS Wrapping Key

It is necessary to retrieve the Wrapping Key from Google KMS.

From the Google Cloud Console Web UI

1. Go to the Cryptographic Keys page.
2. Click the name of the key ring that contains your import job.
3. Click on the **Import Jobs** tab.
4. Click the **Actions** icon for the import job.
5. Select **Download Wrapping Key**.

This will save the wrapping key to `<name of import job>.pem` in your browser's `Downloads` folder.

Alternatively use the Google Cloud SDK Shell. You can specify the filename explicitly here. Use the appropriate syntax for your operating system. The example shows Linux syntax.

```
gcloud kms import-jobs describe \
  --location=location \
  --keyring=keyring \
  --format="value(publicKey.pem)" \
  import-job-name > ${HOME}/wrapping-key.pem
```

7.5. Use cssadmin to generate, wrap and export a key

Using the downloaded Wrapping Key file, call `cssadmin` as follows:

```
cssadmin google-cloud-key-management <key name> <wrapping key> --key-type <key type>
```

Where:

wrapping key

The file you downloaded, for example `myimportjob.pem`.

key-type

Optional and defaults to RSA 2048-bit if none is provided.

Note that, unlike with AWS and Google Compute Engine, no wrapping algorithm parameter is required with Google KMS because it only supports one algorithm.

The following target key types are available:

RSA Keys

- 2048-bit
- 3072-bit
- 4096-bit

Elliptic Curve Keys

- NISTP256
- NISTP384

AES Keys

- AES-256

To maintain the security strength of the transferred target key, ensure that the security strength of the chosen Wrapping Key is greater than or equal to the security strength of the chosen target key. For example, for an Elliptic Curve key generated using NISP256, the Wrapping Key must be RSA 3072-bit or greater.

Running the `cssadmin` command creates a file called `<key name>-wrapped.bin`.

7.6. Upload the wrapped key material

To import the target key into Google KMS use the Google Cloud Platform or the Google Cloud SDK Shell CLI to upload the wrapped key file to Google KMS:

From the Google Cloud Console Web UI:

1. Go to the Cryptographic Keys page.
2. Click the name of the key ring that contains your import job.
3. Click on the name of the target key, then click **Import key version**.
4. Select your import job from the **Select import dropdown**.
5. In the **Upload the wrapped key** selector, select the key that you have already wrapped.
6. If you are importing an asymmetric key, select the algorithm from the **Algorithm** drop-down.
7. Click **Import**.

Alternatively use the Google Cloud SDK Shell. You can specify the filename explicitly here.

```
gcloud kms keys versions import \
  --import-job import-job \
```

```
  --location location \
  --keyring key-ring-name \
  --key key-name \
  --algorithm algorithm-name \
  --rsa-aes-wrapped-key-file path-to-wrapped-key-to-import
```

where

rsa-aes-wrapped-key-file

The bin file produced in Use cssadmin to generate, wrap and export a key.

The key-import request is initiated and you can monitor its status. The initial state for an imported key is **PENDING_IMPORT**. When the state is **ENABLED**, the key has been imported successfully. If the import fails, the status is **IMPORT_FAILED**. If the upload is successful, you can use this HSM-protected key in Google KMS.

8. Integrate with Microsoft Azure Key Vault

8.1. Access Microsoft Azure Key Vault

Microsoft Azure Key Vault can be accessed in the Azure Portal or by using the Azure Command-Line Interface (CLI). Follow the Microsoft Azure documentation to ensure that you use the appropriate authentication, and are communicating with a genuine Azure Key Vault server.

This operation requires the use of Key Vault HSM keys which are currently only available using the Key Vault Premium SKU.

We recommend that you use the Microsoft BYOK guide when carrying out the Azure operations because they might have been updated after the publication of this User Guide for the Cloud Integration Option Pack. See https://docs.microsoft.com/en-us/azure/key-vault/keys/hsm-protected-keys-byok.

8.2. Generate the Azure Key Exchange Key

The Key Exchange Key (KEK) is an RSA key generated in a Key Vault HSM. The KEK is used to encrypt the key that you want to import, and must be:

- An RSA-HSM key (2048-bit, 3072-bit or 4096-bit).
- Generated in the same key vault where you intend to import the target key.
- Created with allowed key operations set to import.

The KEK can be generated using the online portal or using the Azure CLI. For example:

```
az keyvault key create --kty RSA-HSM --size 4096 --name <KEK name> --ops import --vault-name <key vault>
```

Make a note of the resulting key identifier (kid) as you will need this value later. It is in the following form:

```
https://mykeyvault.vault.azure.net/keys/my-key/version
```

8.3. Download the Azure Key Exchange Key

Use the Azure CLI to download the KEK public key to a .pem file. For example:

```
az keyvault key download --name <KEK name> --vault-name <key vault> --file <download file>
```

8.4. Use cssadmin to generate, wrap and export a key

Using the downloaded KEK file, call `cssadmin` as follows:

```
cssadmin microsoft-azure <key name> <wrapping key> --azure-kek <azure kid> --key-type <key type>
```

Where:

wrapping key

The file you downloaded, for example `KEKforBYOK.publickey.pem`.

azure kid

The Key Identifier of the KEK, for example:

```
https://mykeyvault.vault.azure.net/keys/my-key/version.
```

key-type

Optional and defaults to RSA 2048-bit if none is provided.

Note that, unlike with AWS and Google Compute Engine, no wrapping algorithm parameter is required with Microsoft Azure because it only supports one algorithm.

The following target key types are available:

RSA Keys

- 2048-bit
- 3072-bit
- 4096-bit

Elliptic Curve Keys

- NISTP256
- NISTP384
- NISTP521
- SECP256K1

To maintain the security strength of the transferred target key, ensure

> that the security strength of the chosen KEK is greater than or equal to the security strength of the chosen target key. For example, for an Elliptic Curve key generated using NISP256, the KEK must be RSA 3072-bit or greater.

Running the `cssadmin` command creates a file called `KeyTransferPackage-<key name>.byok`.

8.5. Upload the wrapped key material

To import the target key into your Azure Key Vault, use the Azure CLI command to upload the BYOK file to the Key Vault HSM:

```
az keyvault key import --vault-name <vault name> --name <target key name> --byok-file <key transfer package>
```

where

key transfer package

The byok file produced in Use cssadmin to generate, wrap and export a key.

You can also do this step using the Azure Portal.

If the upload is successful, you can use this HSM-protected key in your key vault.

9. Integrate with Salesforce

9.1. Access Bring Your Own Key in the Salesforce interface

1. Log in to the Salesforce interface.
2. From **Setup**, in the Quick find box, enter **Platform Encryption**, and then select **Key Management**.
3. Select **Bring Your Own Key**.

We recommend that you use the Salesforce BYOK Guide when you are carrying the Salesforce operations because they might have been updated after the publication of this User Guide for the Cloud Integration Option Pack. Also use the Salesforce BYOK Guide to ensure that you use the appropriate authentication, and that you are communicating with a genuine Salesforce server. See https://help.salesforce.com/articleView?id=security_pe_byok_setup.htm&type=5.

9.2. Download the Salesforce public key certificate

To protect the delivery of the customer-supplied encryption key, Salesforce provide a public key in a certificate to wrap this encryption key.

1. In the Salesforce interface, select **Create Self-signed certificate**.
2. Enter a unique name for your certificate in the Label field.

   The **Unique Name** field automatically assigns a name based on what you enter in the Label field.

3. The **Exportable Private Key**, **Key Size**, and **Use Platform Encryption** settings are preset.

   These settings ensure that your self-signed certificate is compatible with the Salesforce Shield Platform Encryption.

4. Select Download Certificate.

If you want to use a CA-signed certificate, follow the instructions in the Salesforce BYOK Guide at https://help.salesforce.com/articleView?id=security_pe_byok_generate_cert.htm&type=5.

9.3. Use cssadmin to generate, wrap, and export a key and its hash

Using the downloaded public key certificate, call `cssadmin` as follows:

```
cssadmin salesforce <key name> <wrapping key> <wrapping algorithm>
```

Where:

key name

The name of the key to export. If the key does not exist, `cssadmin` generates it.

wrapping key

The certificate file you downloaded, with the `.crt` file extension.

wrapping algorithm

The algorithm to use for wrapping.

At the time of writing, Salesforce requires the wrapping algorithm to be `RSAES_OAEP_SHA_1`.

Running this command creates two files ready for upload:

<key name>-wrapped

The wrapped key file. Salesforce refers to this as the **Encrypted Tenant Secret**.

<key name>-hash

The hashed key file. Salesforce refers to this as the **Hashed Tenant Secret**.

9.4. Upload the wrapped and hashed key data to Salesforce

On the Bring Your Own Key page of the Salesforce interface, upload the two files under **Upload Tenant Secret**. After the upload, the key is listed on the **Key Management page** and is ready for use.

At this point users have now extended trust to Salesforce in terms of use of the key and its destruction.