Microsoft SQL Server Always Encrypted

nShield® HSM Integration Guide

2026-10-29

Member of
Microsoft Intelligent Security Association

Microsoft Security

© Entrust Corporation. All rights reserved.

Table of Contents

1. Introduction . . . 1
    1.1. Product configurations . . . 1
    1.2. Supported nShield hardware and software versions . . . 1
    1.3. Role separation . . . 2
    1.4. Multiple Windows user accounts on a single client server . . . 3
    1.5. Multiple client servers . . . 3
    1.6. Always Encrypted and TDE . . . 3
    1.7. Requirements . . . 3
2. Configure computers and accounts . . . 4
    2.1. Create domain accounts . . . 4
    2.2. Allow domain accounts to remote login . . . 4
    2.3. Install SSMS in client . . . 5
    2.4. Install the SQL database engine in remote Windows server . . . 5
    2.5. Create a test database . . . 5
    2.6. Create the SQL logins . . . 6
3. Install and configure the nShield HSM CNG provider . . . 9
    3.1. Install the Entrust nShield HSM . . . 9
    3.2. Install the Security World software and create a Security World . . . 9
    3.3. Select the protection method . . . 12
    3.4. Create the OCS or Softcard . . . 13
    3.5. Register the CNG provider . . . 15
    3.6. Install and configure SqlServer PowerShell module . . . 17
4. Generate the encryption keys . . . 19
    4.1. Generate the Always Encrypted Column Master Key (CMK) . . . 19
    4.2. Generate My Column Master Key (MyCMK) and My Column Encryption Key (MyCEK) with SSMS . . . 23
    4.3. Generate MyCMK and MyCEK with PowerShell . . . 31
5. Encrypt or decrypt a column with SSMS . . . 33
    5.1. Encrypt a column . . . 33
    5.2. View an encrypted column . . . 37
    5.3. Remove column encryption . . . 38
6. Encrypt or decrypt a column with PowerShell . . . 41
    6.1. Encrypt a column . . . 41
    6.2. Remove column encryption . . . 41
7. Test access to Always Encrypted keys by another user . . . 43
8. Supported PowerShell SqlServer cmdlets . . . 44
9. Additional resources and related products . . . 46

9.1. nShield HSMs 46
9.2. nShield as a Service 46
9.3. Entrust products 46
9.4. nShield product documentation 46

Chapter 1. Introduction

Always Encrypted is a feature in Windows SQL Server designed to protect sensitive data both at rest and in flight between a client application server and Azure or SQL Server database(s).

Data protected by Always Encrypted remains in an encrypted state until it has reached the client application server. This effectively mitigates man-in-the-middle attacks and provides assurances against unauthorized activity from rogue DBAs or admins with access to Azure or SQL server databases.

The Entrust nShield HSM secures the key used to protect the Column Master Key. The Column Master Key is stored in an encrypted state on the client application server.

1.1. Product configurations

Entrust successfully tested nShield HSM integration with Windows SQL Server and the Always Encrypted feature in the following configurations:

1.1.1. Remote server

Product	Version
Base OS	Windows Server Datacenter 2025
SQL Server	Microsoft SQL Server 2025 RTM 17.0.1000.7

1.1.2. Client

Product	Version
Base OS	Windows 11 Enterprise
Microsoft SQL Server Management Studio	22.2.11

1.2. Supported nShield hardware and software versions

Entrust has successfully tested with the following nShield hardware and software versions:

HSM	Security World Software	Firmware	Netimage
Connect 5c	13.6.18 (LTS 8)	13.4.5 (FIPS 140-3 certified)	13.6.18
nShield XC	13.6.18 (LTS 8)	12.72.3 (FIPS 140-2 certified)	13.6.18
nSaaS	12.80.4	12.72.1 (FIPS 140-2 certified)	12.80.5

1.3. Role separation

The generation of keys and the application of these keys for encryption or decryption are separate processes. The processes can be assigned to users with various access permissions, or Duty Roles. The table below shows the processes and duty roles with reference to the Security Administrator and the database Administrator.

Entrust recommends that you allow only unprivileged connections unless you are performing administrative tasks.

Process	Duty Role
Generating the Column Master Key (CMK) and Column Encryption Key (CEK)	Security Administrator
Applying the CMK and CEK in the database	Database Administrator

Four database permissions are required for Always Encrypted.

Operation	Description
`ALTER ANY COLUMN MASTER KEY`	Required to generate and delete a column master key
`ALTER ANY COLUMN ENCRYPTION KEY`	Required to generate and delete a column encryption key
`VIEW ANY COLUMN MASTER KEY`	Required to access and read the metadata of the column master keys to manage keys or query encrypted columns
`VIEW ANY COLUMN ENCRYPTION KEY`	Required to access and read the metadata of the column encryption key to manage keys or query encrypted columns

1.4. Multiple Windows user accounts on a single client server

The Entrust nShield HSM solution for Microsoft SQL Always Encrypted enables keys that are associated with one user to be used by other users, providing secure access to a common database. Ask Entrust Support for hotfix-TAC1266 patch to allow multiple users to use the same always encrypted key.

1.5. Multiple client servers

Each client server wanting access to the content of the encrypted data with a given CEK must have:

- An HSM in the same Security World.
- Hotfix-TAC1266 to allow multiple users to use the same always encrypted key.
- A copy of the CMK key token stored on its local drive.

1.6. Always Encrypted and TDE

The same Security World can be used for Always Encrypted and TDE.

1.7. Requirements

- Knowledge of your organization Certificate Practices Statement and a Security Policy / Procedure in place covering administration of the HSM.
- Access to the Entrust TrustedCare Portal.
- An Entrust nShield HSM.
- Network environment with usable ports 9004 and 9005 for the HSM.

Familiarize yourself with the nShield Documentation.

- The importance of a correct quorum for the Administrator Card Set (ACS).
- Whether Operator Card Set (OCS) protection or Softcard protection is required.
- If OCS protection is to be used, a 1-of-N quorum must be used.
- Whether your Security World must comply with FIPS 140 Level 3 or Common Criteria standards. If using FIPS 140 Level 3, it is advisable to create an OCS for FIPS authorization. For more information see FIPS 140 Level 3 compliance.
- Whether to instantiate the Security World as recoverable or not.

Chapter 2. Configure computers and accounts

For this integration, the Microsoft SQL Server Database Engine was deployed on a remote Windows server, and a Windows PC served as the client. Both systems were joined to the same Windows domain, and Windows authentication was used for additional security. A test database was created to demonstrate data encryption.

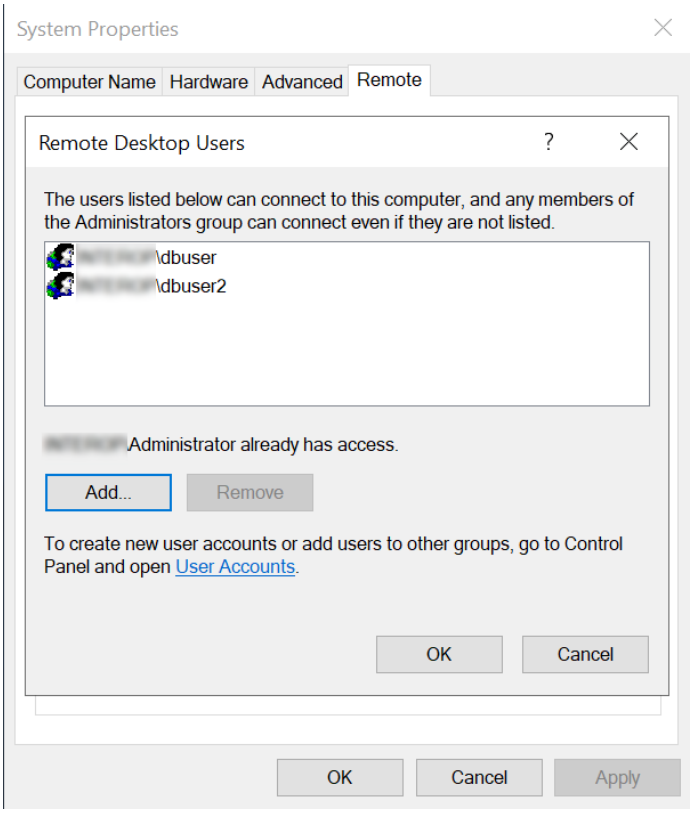
2.1. Create domain accounts

Create two Windows domain accounts. For example:

- <domain>\dbuser
- <domain>\dbuser2

2.2. Allow domain accounts to remote login

1. In the Windows PC (client), enter **advance settings** in the search box and select **View advanced system settings**.
2. Select the **Remote** tab in the **System Properties** dialog. Then select **Select Users...**.
3. Add the following users:
    - `<domain>\dbuser`
    - `<domain>\dbuser2`



2.3. Install SSMS in client

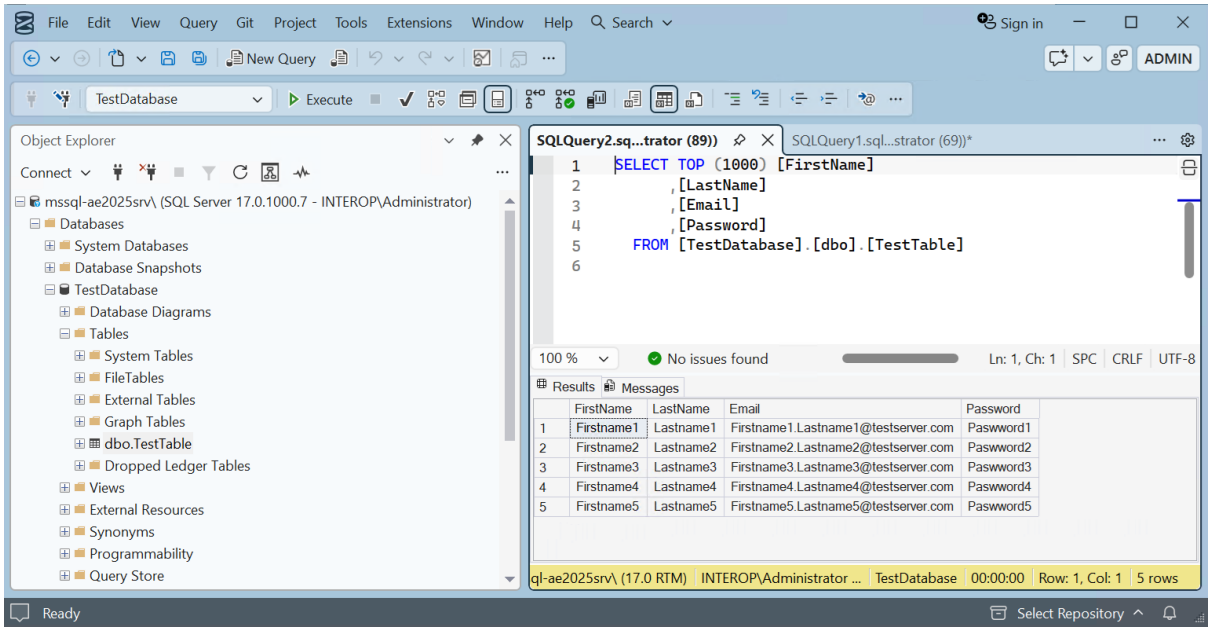
Install Microsoft SQL Server Management Studio in the Windows PC (client).

2.4. Install the SQL database engine in remote Windows server

1. Install the SQL engine in the remote Windows server.
2. Open the firewall ports 1433, 1434, 445, and 49170 for access by the SQL database engine, SQL browser, and Active Directory for domain account authorization.

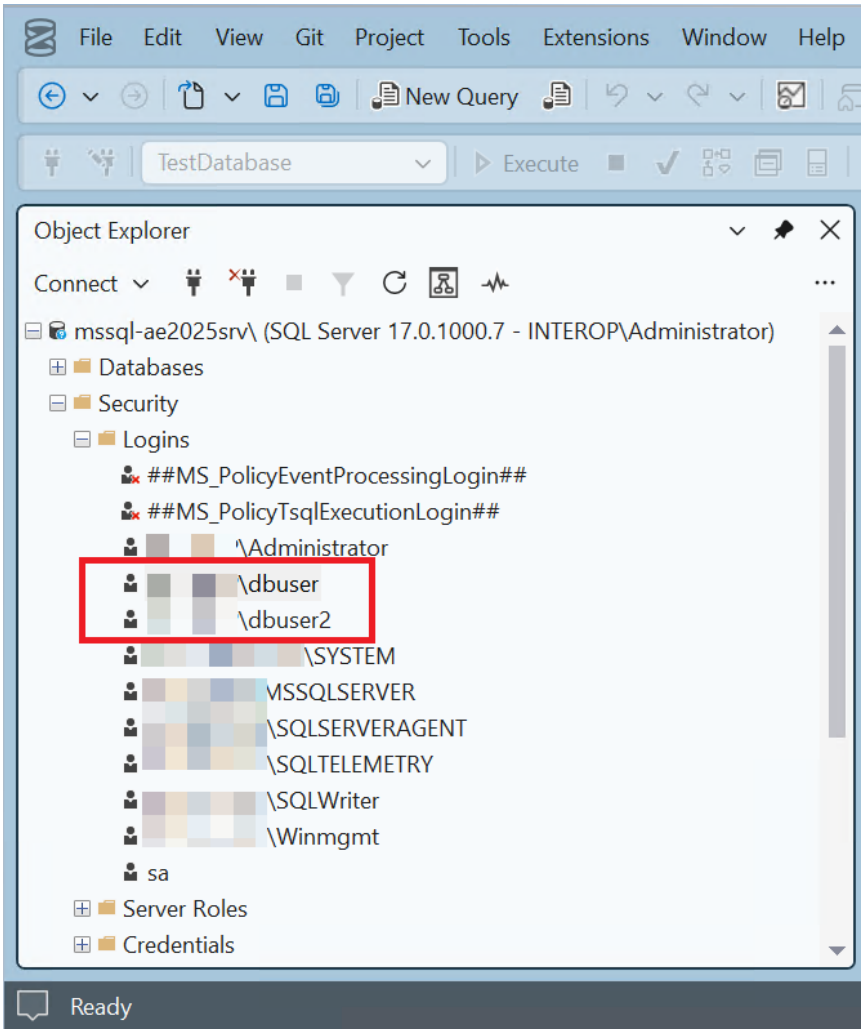
2.5. Create a test database

A test database named **TestDatabase** was created. You can use your own database instead.

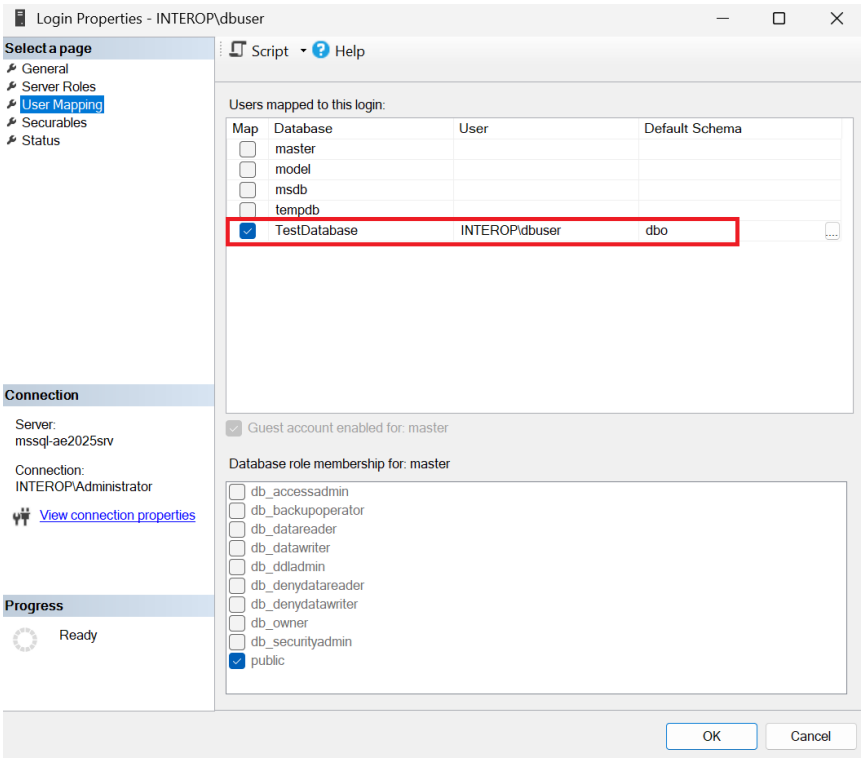


2.6. Create the SQL logins

1. Create two SQL logins with the domain accounts `<domain>\dbuser` and `<domain>\dbuser2` with **Default Database** equal to "TestDatabase".



2. Set the **User Mapping** of the two SQL logins as database owners, **db_owner**, of TestDatabase.


Login Properties - INTEROP\dbuser
Select a page
General
Server Roles
User Mapping
Securables
Status
Script
Help
Users mapped to this login:
Map
Database
User
Default Schema
master
model
msdb
tempdb
TestDatabase
INTEROP\dbuser
dbo
Guest account enabled for: master
Database role membership for: master
db_accessadmin
db_backupoperator
db_datareader
db_datawriter
db_ddladmin
db_denydatareader
db_denydatawriter
db_owner
db_securityadmin
public
Connection
Server:
mssql-ae2025srv
Connection:
INTEROP\Administrator
View connection properties
Progress
Ready
OK
Cancel

Chapter 3. Install and configure the nShield HSM CNG provider

These steps must be performed on the Windows PC (client) using the <domain_name>\Administrator account.

3.1. Install the Entrust nShield HSM

Install the nShield Connect HSM locally, remotely, or remotely via the serial console. Condensed instructions are available in the following Entrust nShield Support articles.

- How To: Locally Set up a new or replacement nShield Connect.
- How To: Remotely Setup a new or replacement nShield Connect.
- How To: Remotely Setup a new or replacement nShield Connect XC Serial Console Model.

For detailed instructions, see the nShield Hardware Install and Setup Guides.

3.2. Install the Security World software and create a Security World

1. Install the Security World software. For detailed instructions, see the nShield Security World Software v13.6.12 Installation Guide.
2. Install hotfix-TAC1266 if multiple Windows user accounts need access to the same data. Contact nShield support to download the Hotfix. To perform the installation:
    a. Open a command window as Administrator and uninstall the CNG:

```
C:\Users\Administrator>"C:\Program Files\nCipher\nfast\bin\cnginstall32" --uninstall
ncksps w.dll removed.

ncpp.dll removed.


C:\Users\Administrator>"C:\Program Files\nCipher\nfast\bin\cnginstall" --uninstall
ncksps w.dll removed.

ncpp.dll removed.
```

    b. Reboot the server.
    c. Copy files as per the installation instructions in the Hotfix package:

```
C:\Users\Administrator>copy C:\Users\Administrator\Downloads\hotfix-TAC1266\hotfix-
TAC1266\nfast\c\caping\vs2022-64\lib\* "C:\Program Files\nCipher\nfast\c\caping\vs2022-64\lib\."
```

```
C:\Users\Administrator\Downloads\hotfix-TAC1266\hotfix-TAC1266\nfast\c\caping\vs2022-
64\lib\nckspsw.dll
Overwrite C:\Program Files\nCipher\nfast\c\caping\vs2022-64\lib\.\nckspsw.dll? (Yes/No/All): All
C:\Users\Administrator\Downloads\hotfix-TAC1266\hotfix-TAC1266\nfast\c\caping\vs2022-
64\lib\nckspsw.lib
C:\Users\Administrator\Downloads\hotfix-TAC1266\hotfix-TAC1266\nfast\c\caping\vs2022-
64\lib\nckspsw.map
C:\Users\Administrator\Downloads\hotfix-TAC1266\hotfix-TAC1266\nfast\c\caping\vs2022-
64\lib\nckspsw.pdb
C:\Users\Administrator\Downloads\hotfix-TAC1266\hotfix-TAC1266\nfast\c\caping\vs2022-64\lib\ncpp.dll
C:\Users\Administrator\Downloads\hotfix-TAC1266\hotfix-TAC1266\nfast\c\caping\vs2022-64\lib\ncpp.lib
C:\Users\Administrator\Downloads\hotfix-TAC1266\hotfix-TAC1266\nfast\c\caping\vs2022-64\lib\ncpp.map
C:\Users\Administrator\Downloads\hotfix-TAC1266\hotfix-TAC1266\nfast\c\caping\vs2022-64\lib\ncpp.pdb
        8 file(s) copied.

C:\Users\Administrator>copy C:\Users\Administrator\Downloads\hotfix-TAC1266\hotfix-
TAC1266\nfast\c\caping\vs2022-32\lib\* "C:\Program Files\nCipher\nfast\c\caping\vs2022-32\lib\."
C:\Users\Administrator\Downloads\hotfix-TAC1266\hotfix-TAC1266\nfast\c\caping\vs2022-
32\lib\nckspsw.dll
Overwrite C:\Program Files\nCipher\nfast\c\caping\vs2022-32\lib\.\nckspsw.dll? (Yes/No/All): All
C:\Users\Administrator\Downloads\hotfix-TAC1266\hotfix-TAC1266\nfast\c\caping\vs2022-
32\lib\nckspsw.lib
C:\Users\Administrator\Downloads\hotfix-TAC1266\hotfix-TAC1266\nfast\c\caping\vs2022-
32\lib\nckspsw.map
C:\Users\Administrator\Downloads\hotfix-TAC1266\hotfix-TAC1266\nfast\c\caping\vs2022-
32\lib\nckspsw.pdb
C:\Users\Administrator\Downloads\hotfix-TAC1266\hotfix-TAC1266\nfast\c\caping\vs2022-32\lib\ncpp.dll
C:\Users\Administrator\Downloads\hotfix-TAC1266\hotfix-TAC1266\nfast\c\caping\vs2022-32\lib\ncpp.lib
C:\Users\Administrator\Downloads\hotfix-TAC1266\hotfix-TAC1266\nfast\c\caping\vs2022-32\lib\ncpp.map
C:\Users\Administrator\Downloads\hotfix-TAC1266\hotfix-TAC1266\nfast\c\caping\vs2022-32\lib\ncpp.pdb
        8 file(s) copied.

C:\Users\Administrator>copy C:\Users\Administrator\Downloads\hotfix-TAC1266\hotfix-
TAC1266\nfast\lib\versions\caping-atv.txt "C:\Program Files\nCipher\nfast\lib\versions\."
        1 file(s) copied.
```

d. Open a command window as Administrator and install the CNG (64 bit). See `C:\Program Files\nCipher\nfast\bin` for 32 bit utilities (e.g. cnginstall32).

```
C:\Users\Administrator>"C:\Program Files\nCipher\nfast\bin\cnginstall" --install
nckspsw.dll installed.

ncpp.dll installed.


C:\Users\Administrator>"C:\Program Files\nCipher\nfast\bin\cngregister"
Provider 'nCipher Primitive Provider' registered successfully
Algorithm SHA1 registered successfully
Algorithm SHA256 registered successfully
Algorithm SHA384 registered successfully
Algorithm SHA512 registered successfully
Algorithm SHA224 registered successfully
Algorithm AES registered successfully
Algorithm 3DES registered successfully
Algorithm 3DES_112 registered successfully
Algorithm RSA registered successfully
Algorithm DSA registered successfully
Algorithm ECDSA_P256 registered successfully
Algorithm ECDSA_P384 registered successfully
Algorithm ECDSA_P521 registered successfully
Algorithm ECDSA_P224 registered successfully
Algorithm DH registered successfully
Algorithm ECDH_P256 registered successfully
Algorithm ECDH_P384 registered successfully
```

```
Algorithm ECDH_P521 registered successfully
Algorithm ECDH_P224 registered successfully
Algorithm RNG registered successfully
Provider 'nCipher Security World Key Storage Provider' registered successfully
Interface KEY_STORAGE registered successfully
Created nShieldServiceAgent Run registry entry
```

e. Reboot the server.

3. Add the Security World utilities path to the system path. This path is typically `C:\Program Files\nCipher\nfast\bin`.
4. Open the firewall port 9004 for the HSM connections.
5. If using remote administration, open firewall port 9005 for the Entrust nShield Trusted Verification Device (TVD).
6. Inform the HSM of the client's location. In this integration the client is this machine. For instructions, see Configuring the nShield HSM to use the client. If it's a high-availability setup, repeat the client configuration for each HSM.
7. Enroll this machine as a client of the HSM. For instructions, see Configuring client computers to use the nShield HSM. If it's a high-availability setup, repeat the enrolment for each HSM.
8. Open a command window and run the following utility to confirm the HSM is **operational**:

```
C:\Users\Administrator>enquiry
Server:
 enquiry reply flags  none
 enquiry reply level  Six
 serial number        6A74-1261-7843 810B-03E0-D947
 mode                 operational
 version              13.6.18
...
Module #1:
 enquiry reply flags  UnprivOnly
 enquiry reply level  Six
 serial number        6A74-1261-7843
 mode                 operational
 version              13.4.5
 ...
 Module #2:
 enquiry reply flags  UnprivOnly
 enquiry reply level  Six
 serial number        810B-03E0-D947
 mode                 operational
 version              12.72.4
 ...
```

9. Create your Security World if one does not already exist or copy an existing one. Follow your organization's security policy for this. For more information see Create a new Security World.

Administrator Card Set (ACS) cards cannot be duplicated after the

> Security World is created. You may want to create extras in case of a card failure or a lost card.

10. Confirm the Security World is **Usable**:

```
>nfkminfo
World
 generation  2
 state       0x3737000c Initialised Usable ...
 ...
Module #1
 generation 2
 state      0x2 Usable
 ...
 Module #2
 generation 2
 state      0x2 Usable
 ...
```

3.3. Select the protection method

The following protection methods can be used to authorize access to the keys protected by the HSM. Typically, an organization's security policies dictate the use of one or the other.

- Operator Cards Set (OCS) are smartcards that are presented to the physical smartcard reader of an HSM. For more information on OCS use, properties, and K-of-N values, see Operator Card Sets (OCS).
- Softcards are logical tokens (passphrases) that protect the key and authorize its use. For more information about using softcards, see Softcards.
- Module protection has no passphrase.

Follow your organization's security policy to select an authorization access method.

Depending on the protection method selected, you may need to define some environment variables. You have the option to set these environment variables with the Windows `set` command, or edit file `C:\Program Files\nCipher\nfast\cknfastrc`. For reference, all environment variables are listed in nShield PKCS #11 library environment variables.

Enable softcard protection:

```
C:\Users\Administrator.INTEROP>set CKNFAST_LOADSHARING=1
```

Enable Module protection:

```
>set CKNFAST_FAKE_ACCELERATOR_LOGIN=1
```

Sample `C:\Program Files\nCipher\nfast\cknfastrc` file:

```
# Enable Softcard protection
CKNFAST_LOADSHARING=1

# Enable Module protection
CKNFAST_FAKE_ACCELERATOR_LOGIN=1

# OCS Preload file location and card set state
NFAST_NFKM_TOKENSFILE="C:\Program Files\nCipher\nfast\preloadtoken"
CKNFAST_NONREMOVABLE=1
```

3.4. Create the OCS or Softcard

If using OCS protection, create the OCS now.

1. If using remote administration, edit file `C:\ProgramData\nCipher\Key Management Data\config\cardlist` adding the serial number of the card(s) to be presented, or the wildcard "*".
2. Open a command window as Administrator.
3. Run the `createocs` utility as described below. Enter a passphrase or password at the prompt.

   Follow your organization's security policy for the values K/N. In this example note that `slot 2` remote via a TVD, is used to present the card, and K=1 and N=1.

   Administrator Card Set (ACS) authorization is required to create an OCS in FIPS 140 level 3.

   After an OCS card set has been created, the cards cannot be duplicated. You may want to create extras in case of a card failure or a lost card.

   The preload utility loads OCS onto the HSM. This feature makes the OCS available for use after been physically removed from the HSM for safe storage or other reasons. Add the `-p` (persistent) option to the command below to have authentication after the OCS card has been removed from the HSM front panel slot, or from the TVD.

   ```
   > createocs -m1 -s2 -N testOCS -Q 1/1

   FIPS 140-2 level 3 auth obtained.

   Creating Cardset:
    Module 1: 0 cards of 1 written
    Module 1 slot 0: Admin Card #1
    Module 1 slot 2: empty
    Module 1 slot 3: empty
    Module 1 slot 2: blank cardSteps:

    Module 1 slot 2:- passphrase specified - writing card
   Card writing complete.
   ```

```
cardset created; hkltu = a165a26f929841fe9ff2acdf4bb6141c1f1a2eed
```

The authentication provided by the OCS as shown in the command line above is non-persistent and only available while the OCS card is inserted in the HSM front panel slot, or the TVD.

4. Verify the OCS created:

```
>nfkminfo -c
Cardset list - 1 cardsets:  (P)ersistent/(N)ot, (R)emoteable/(L)ocal-only
 Operator logical token hash               k/n timeout  name
 7aaf758bc6790206198ea5218040d4faa09f035f  1/5  none-NL testOCSnopassphrase
```

The `rocs` utility also shows the OCS created:

```
>rocs
`rocs' key recovery tool
Useful commands: `help', `help intro', `quit'.
rocs> list cardset
No. Name                    Keys (recov) Sharing
  1 testOCSnopassphrase     0 (0)        1 of 5
rocs> quit
```

If using Softcard protection, create the Softcard now.

1. Run the following utility and enter a passphrase/password at the prompt:

```
>ppmk -n testSC

Enter new pass phrase:
Enter new pass phrase again:
New softcard created: HKLTU d9414ed688c6405aab675471d3722f8c70f5d864
```

2. Verify the Softcard was created:

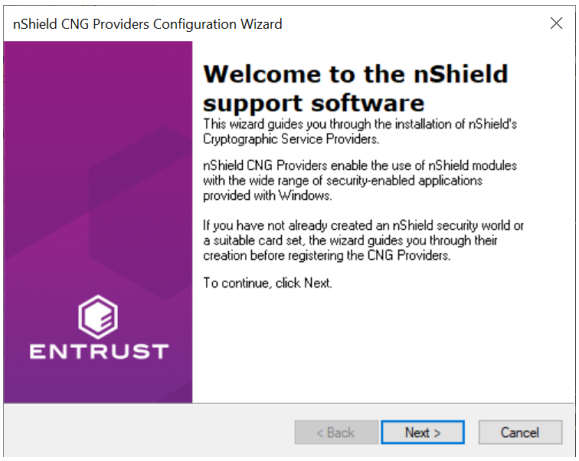
```
>nfkminfo -s
SoftCard summary - 1 softcards:
 Operator logical token hash               name
 d9414ed688c6405aab675471d3722f8c70f5d864  testSC
```

The `rocs` utility also shows the OCS and Softcard created.

```
>rocs
`rocs' key recovery tool
Useful commands: `help', `help intro', `quit'.
rocs> list cardset
No. Name                    Keys (recov) Sharing
  1 testOCS                 0 (0)        1 of 1
  2 testSC                  0 (0)        (softcard)
rocs>quit
```

3.5. Register the CNG provider

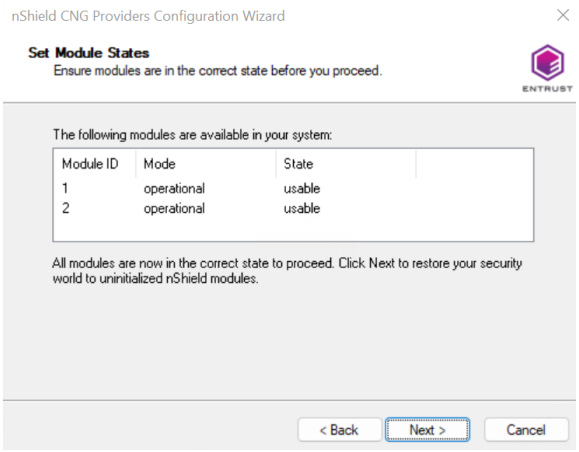
1. Select **Start** > **Entrust nShield Security Worls** > **CNG configuration wizard**.
2. Select **Next** on the **Welcome** window.



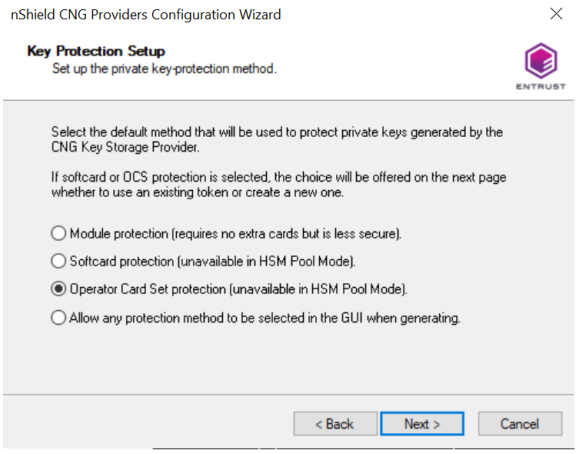
3. Select **Next** on the **Enable HSM Pool Mode** window, leaving **Enable HSM Mode for CNG Providers** un-checked.

   If you intend to use multiple HSMs in a failover and load-sharing capacity, select **Enable HSM Pool Mode for CNG Providers**. If you do, you can only use module protected keys. Module protection does not provide conventional 1 or 2 factor authentication. Instead, the keys are encrypted and stored as an application key token, also referred to as a Binary Large Object (blob), in the `kmdata/local` directory.

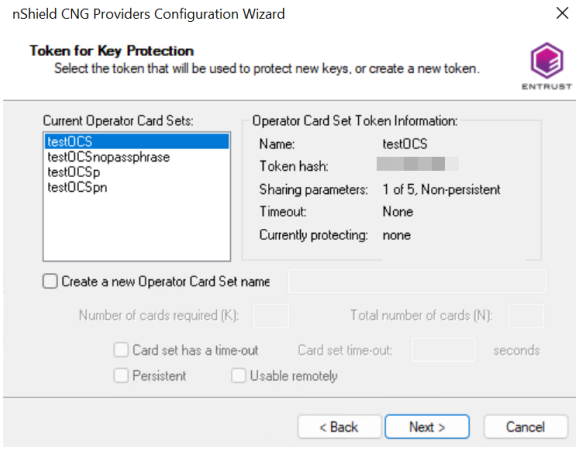
4. In the **Initial setup** window select **Use existing security world**. Then select **Next**.
5. In the **Set Module States** window, select the HSM (Module) if more than one is available. In this example two HSM are **usable** and will be selected. Then select **Next**.



6. In **Key Protection Setup**, select **Operator Card Set protection**. Then select **Next**.



7. Choose from the **Current Operator Card Sets** list. Then select **Next** and **Finish**.

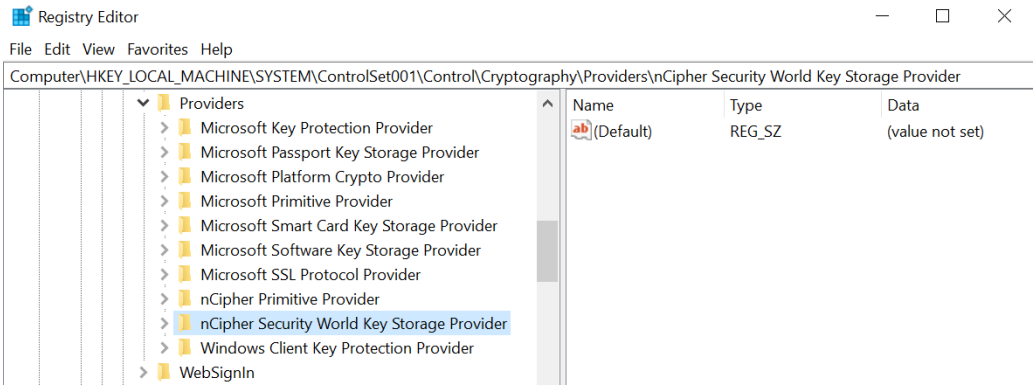


8. Verify the provider with the following commands:

```
>certutil -csplist | findstr nCipher
Provider Name: nCipher Security World Key Storage Provider

>cnglist.exe --list-providers | findstr nCipher
nCipher Primitive Provider
nCipher Security World Key Storage Provider
```

9. Check the registry in `CNGRegistry`:

```
HKEY_LOCAL_MACHINE\SYSTEM\ControlSet001\Control\Cryptography\Providers\nCipherSecurityWorldKeyStorageProvid
er
```



3.6. Install and configure SqlServer PowerShell module

1. Open a PowerShell session as Administrator and run.

```
PS C:\Users\Administrator> [Net.ServicePointManager]::SecurityProtocol = [Net.SecurityProtocolType]::Tls12

PS C:\Users\Administrator> Install-PackageProvider Nuget -force -verbose
...
VERBOSE: Imported provider
'C:\ProgramFiles\PackageManagement\ProviderAssemblies\nuget\2.8.5.208\Microsoft.PackageManagement.NuGetProv
ider.dll' .
```

2. Update PowerShellGet.

```
PS C:\Users\Administrator> Install-Module -Name PowerShellGet -force -verbose
...
VERBOSE: Module 'PowerShellGet' was installed successfully to path
'C:\ProgramFiles\WindowsPowerShell\Modules\PowerShellGet\2.2.5'.
```

3. Download and install the SqlServer module to configure Always Encrypted using Power Shell:

```
PS C:\Users\Administrator> Install-Module -Name SqlServer -force -verbose -AllowClobber
...
VERBOSE: Module 'SqlServer' was installed successfully to path
'C:\ProgramFiles\WindowsPowerShell\Modules\SqlServer\22.4.5.1'.
```

   The `-AllowClobber` parameter allows you to import the specified command if it exists in the current session.

4. Once installed, confirm the install by running the command below.

   If you are using PowerShell ISE, refresh the Commands pane. If you are using PowerShell, open a new session.

```
PS C:\Users\Administrator> Get-Module -list -Name SqlServer
```

```
    Directory: C:\Program Files\WindowsPowerShell\Modules


ModuleType Version    Name                                ExportedCommands
---------- -------    ----                                ----------------
Script     22.4.5.1   SqlServer                           {Add-RoleMember, Add-SqlAvailabilityDatabase, Add-
SqlAvail...
```

Chapter 4. Generate the encryption keys

4.1. Generate the Always Encrypted Column Master Key (CMK)

The CMK is protected by the nShield HMS.

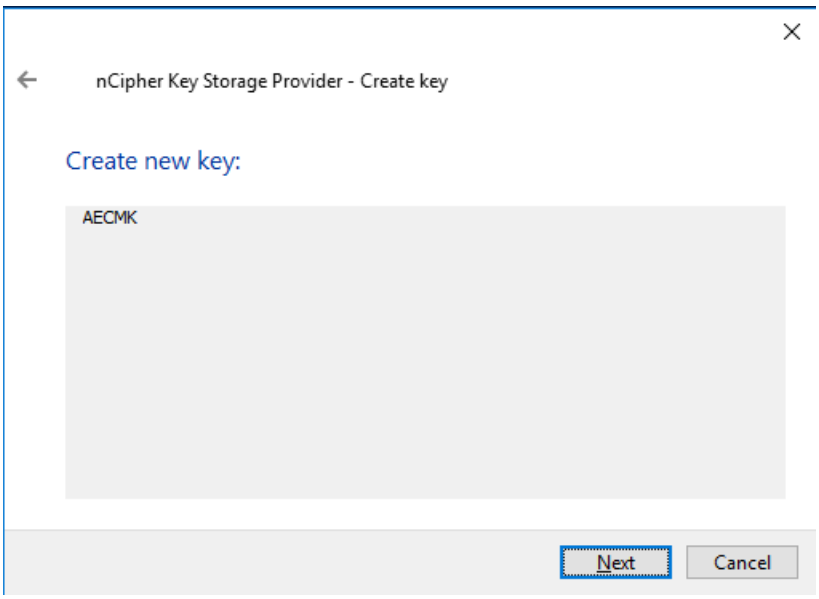
1. Log in to the client using the <domain>\dbuser, or a suitable security administrator account.
2. Launch PowerShell and run the `Generate_AECMK.ps1` script (shown below).

```
$cngProviderName = "nCipher Security World Key Storage Provider"

$cngAlgorithmName = "RSA"

$cngKeySize = 2048

$cngKeyName = "AECMK"

$cngProvider = New-Object System.Security.Cryptography.CngProvider($cngProviderName)

$cngKeyParams = New-Object System.Security.Cryptography.CngKeyCreationParameters

$cngKeyParams.provider = $cngProvider

$cngKeyParams.KeyCreationOptions =
[System.Security.Cryptography.CngKeyCreationOptions]::OverwriteExistingKey

$keySizeProperty = New-Object System.Security.Cryptography.CngProperty("Length",
[System.BitConverter]::GetBytes($cngKeySize), [System.Security.Cryptography.CngPropertyOptions]::None);

$cngKeyParams.Parameters.Add($keySizeProperty)

$cngAlgorithm = New-Object System.Security.Cryptography.CngAlgorithm($cngAlgorithmName)

$cngKey = [System.Security.Cryptography.CngKey]::Create($cngAlgorithm, $cngKeyName, $cngKeyParams)
```

   a. Run the following command:

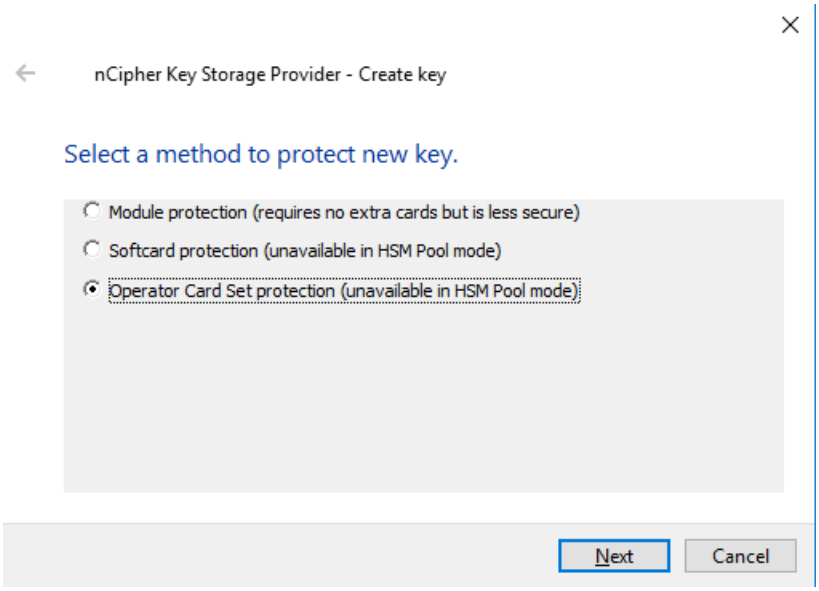
```
> PowerShell -ExecutionPolicy Bypass -File Generate_AECMK.ps1
```

   The following dialog appears.

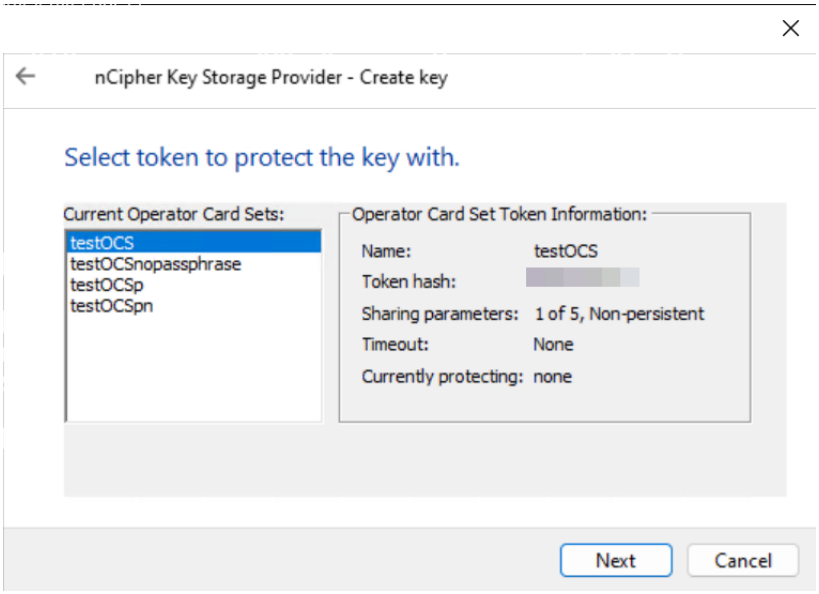


b. Select **Next**.

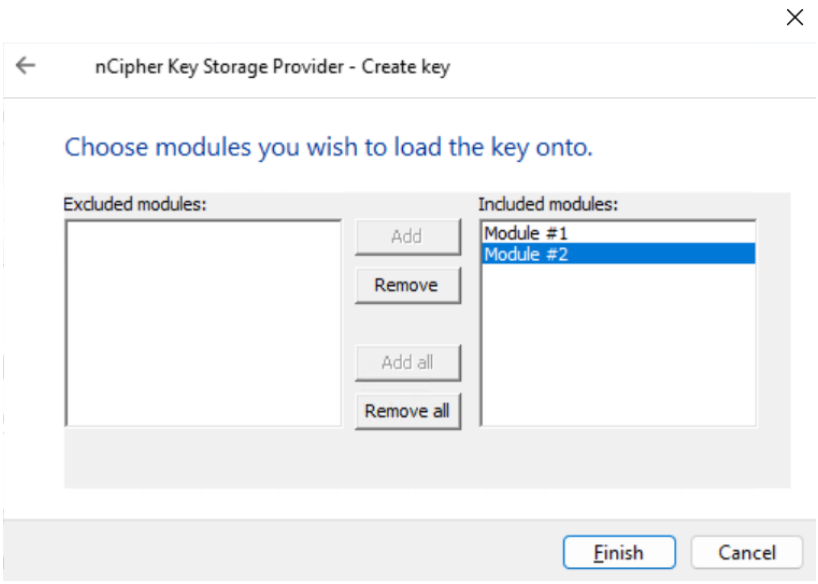
c. Select the **Operator Card Set Protection**. Insert the OCS card in the HSM or via the TVB is using remote administration and select **Next**.



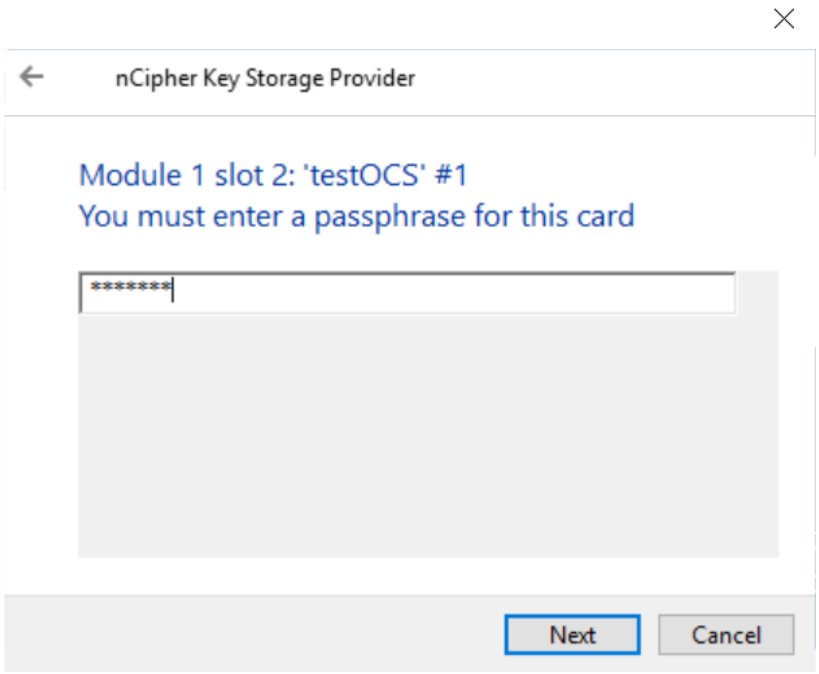
d. Select the OCS and then select **Next**.



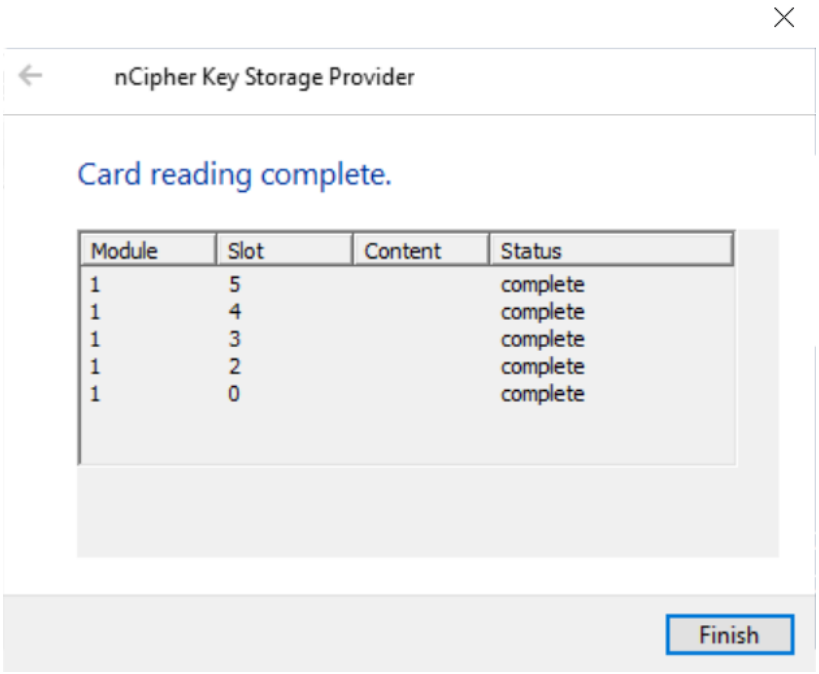
e. Select the HSM and select **Finish**.



f. Enter the OCS passphrase and select **Next**.



g. Select **Finish**.



A 2048-bit RSA key pair, called `AECMK`, has been generated. The key is encrypted in the HSM and then pushed to the requesting client server, where it is stored as an Application Key Token in the `%NFAST_KMDATA%\local` folder. This folder is typically `C:\ProgramData\nCipher\Key Management Data\local`.

3. Verify the new key:

```
>nfkminfo -k
```

```
Key list - 1 keys
 AppName caping              Ident user--e57798f862740453d02379579c1758ddfa2189db
```

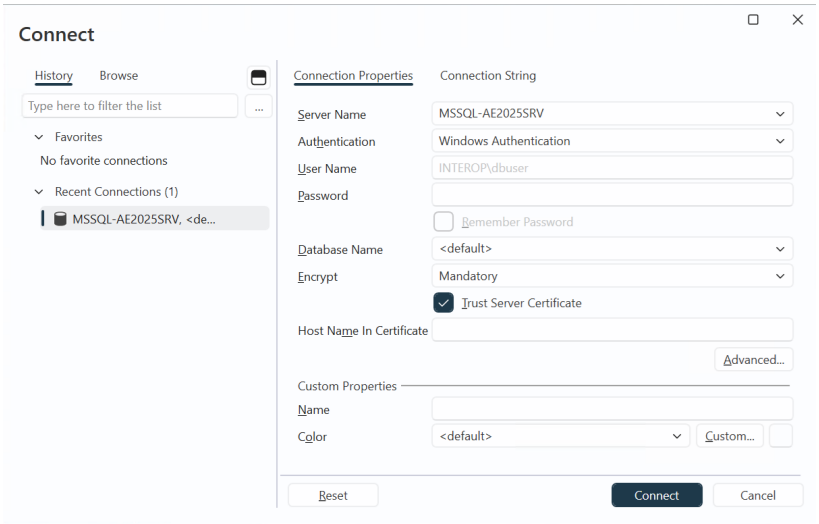
4. Display the information about the key by copy-pasting the key name above as follows:

```
>nfkminfo -k caping user--e57798f862740453d02379579c1758ddfa2189db
Key AppName caping Ident user--e57798f862740453d02379579c1758ddfa2189db
 BlobKA length         1128
 BlobPubKA length      484
 BlobRecoveryKA length 1496
 name                  "AECMK"
 hash                  a9b00ff93e06b578c1dc8b2ad50e7eae218b2660
 recovery              Enabled
 protection            CardSet
 other flags           PublicKey !SEEAppKey !NVMemBlob +0x0
 cardset               edb3d45a28e5a6b22b033684ce589d9e198272c2
 gentime               2026-10-28 15:22:40
 SEE integrity key     NONE

BlobKA
 format                6 Token
 other flags           0x0
 hkm                   b01a7d6ac910b720bf4319f5067a4569f087f81b
 hkt                   edb3d45a28e5a6b22b033684ce589d9e198272c2
 hkr                   none

BlobRecoveryKA
 format                9 UserKey
 other flags           0x0
 hkm                   none
 hkt                   none
 hkr                   d00f8956fcda01bd4c7f539ee042ef6b5ac75917

BlobPubKA
 format                5 Module
 other flags           0x0
 hkm                   c2be99fe1c77f1b75d48e2fd2df8dffc0c969bcb
 hkt                   none
 hkr                   none

Extra entry #1
 typecode              0x10000 65536
 length                60
Not a blob
```

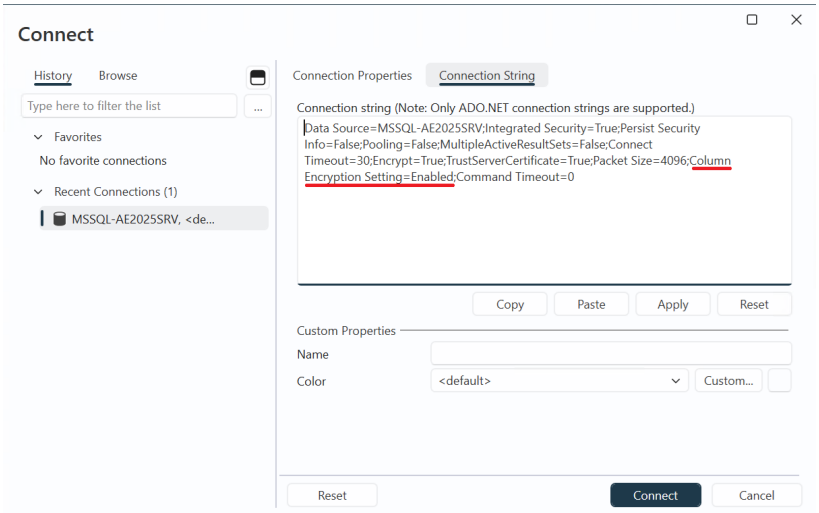
4.2. Generate My Column Master Key (MyCMK) and My Column Encryption Key (MyCEK) with SSMS

This key will encrypt all subsequent Column Encryption keys (CEKs) in your database.

1. Log in to the client using the <domain>\dbuser account.
2. Launch **Microsoft SQL Server Management Studio**.
3. Connect to the database on the remote SQL server:
    a. Select the **Login** tab and set it as follows:

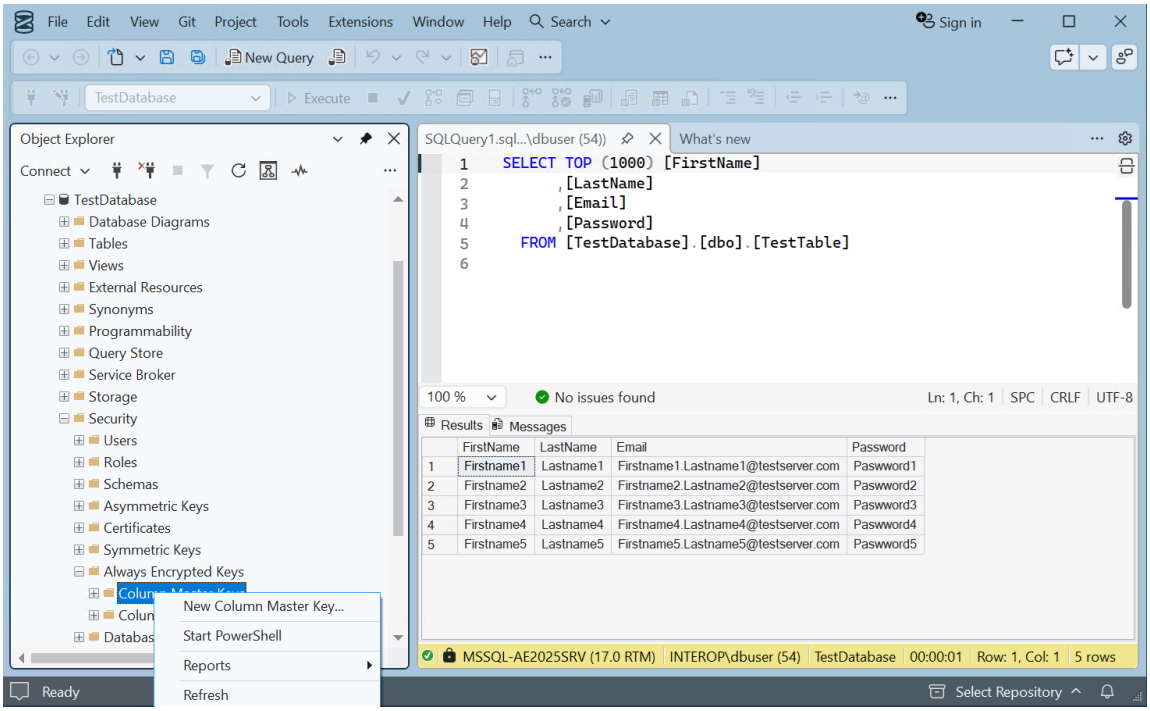


b. Select the **Connection String** tab and check that **Column Encryption Setting** is **Enabled** (this is the default).
This enables the Always Encrypted feature.



c. Select **Connect**.

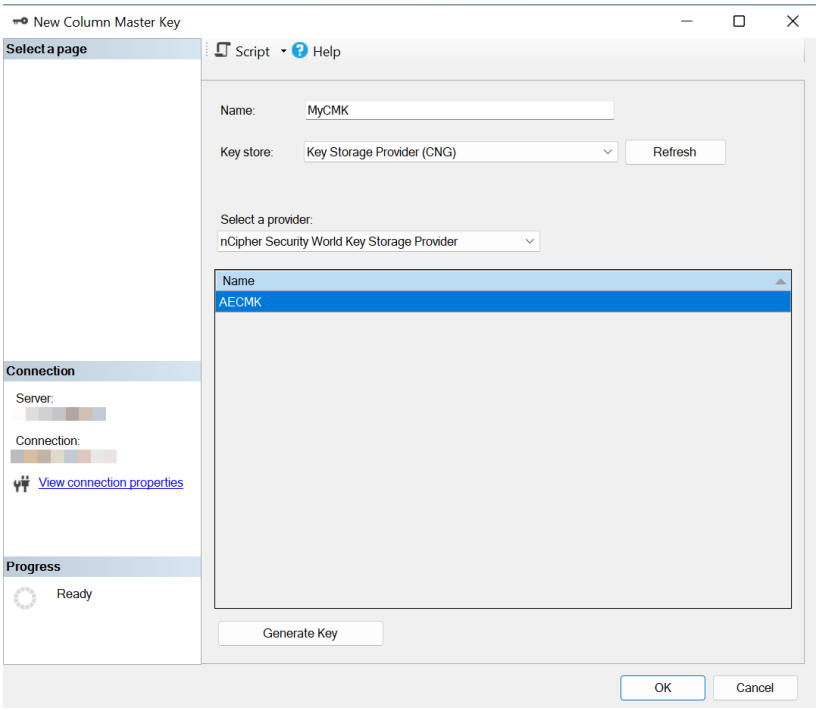
4. Using the **Object Explorer**, select the **Security** directory under the required database, then select **Always Encrypted Keys** > **Column Master Key** > **New Column Master Key**.



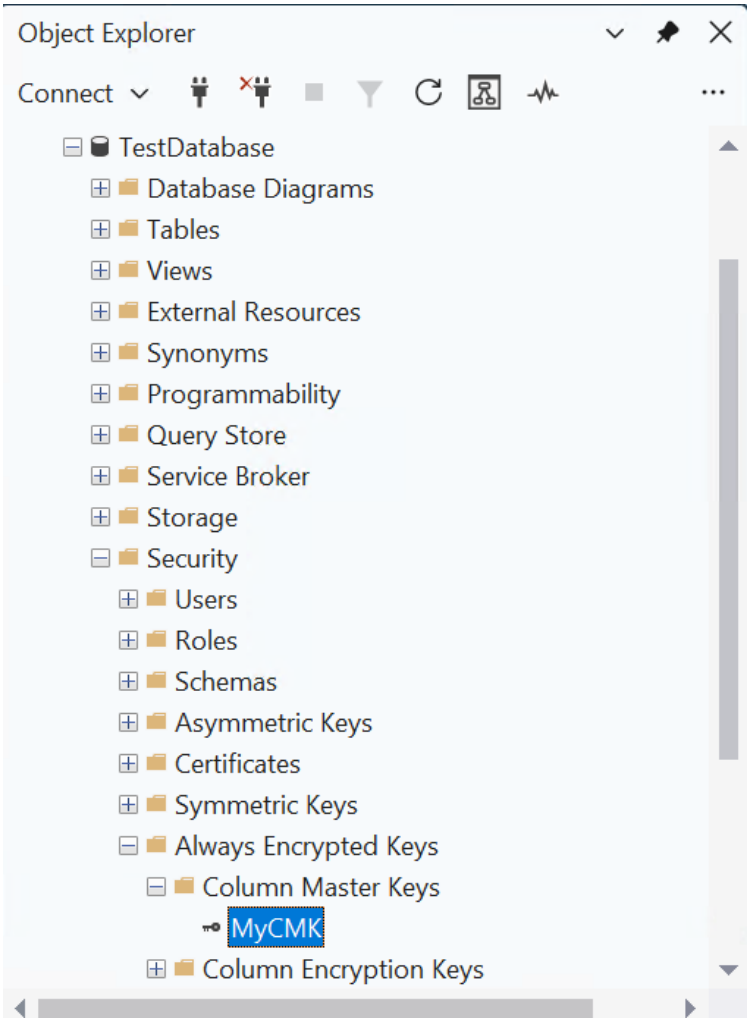
5. Enter the following information on the **Column Master Keys** dialog:
   a. Enter a **Name**, for example **MyCMK**.
   b. Select **Key Storage Provider (CNG)** from the **Key store** drop-down list and then **Select a provider**.
   c. Select **nCipher Security World Key Storage Provider** from the drop-down list.

      The **AECMK** key created in an earlier step appears in **Name**.

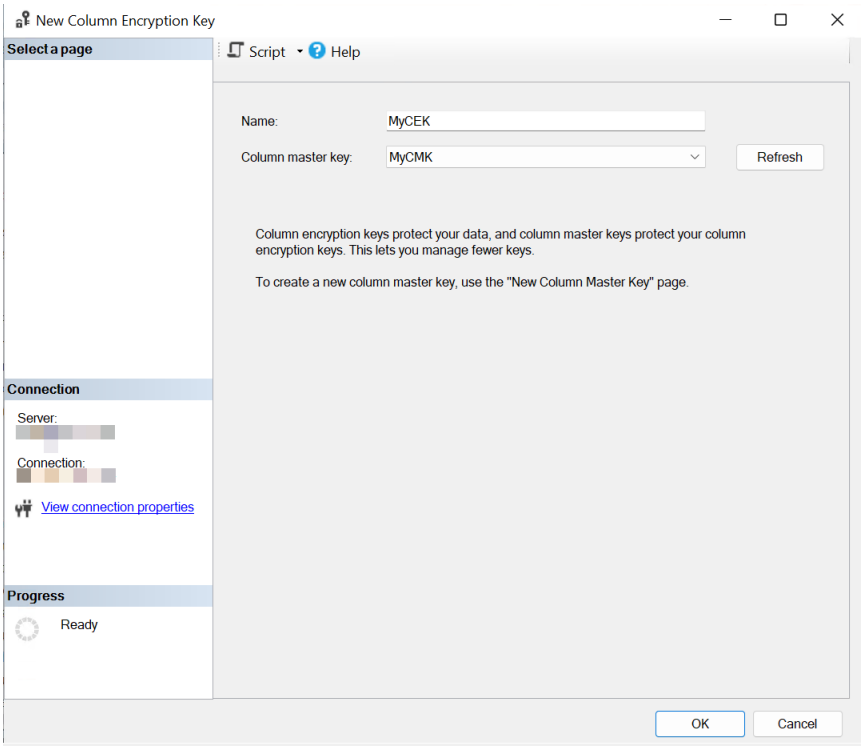
   d. Select **OK** to create a new key using the nShield HSM and CNG KSP.



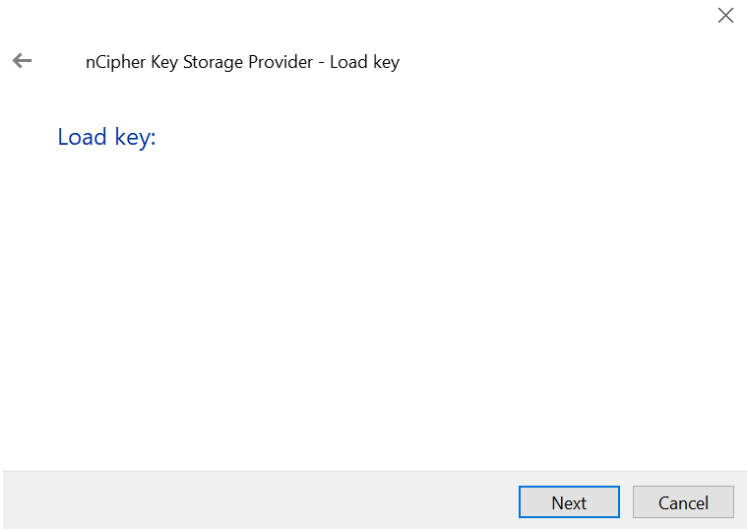
6. The newly created **MyCMK** is in the database under **Security** > **Always Encrypted Keys** > **Column Master Keys**.



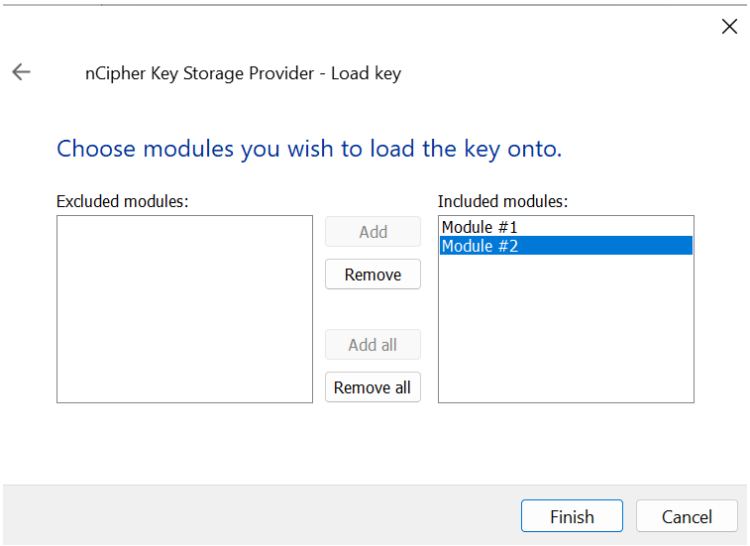
7. Using **Object Explorer**, select the **Security** directory under the required database. Select **Always Encrypted Keys** to expand it, then select **New Column Encryption Key**.
8. Enter **Name**, select the CMK, then select **OK**.



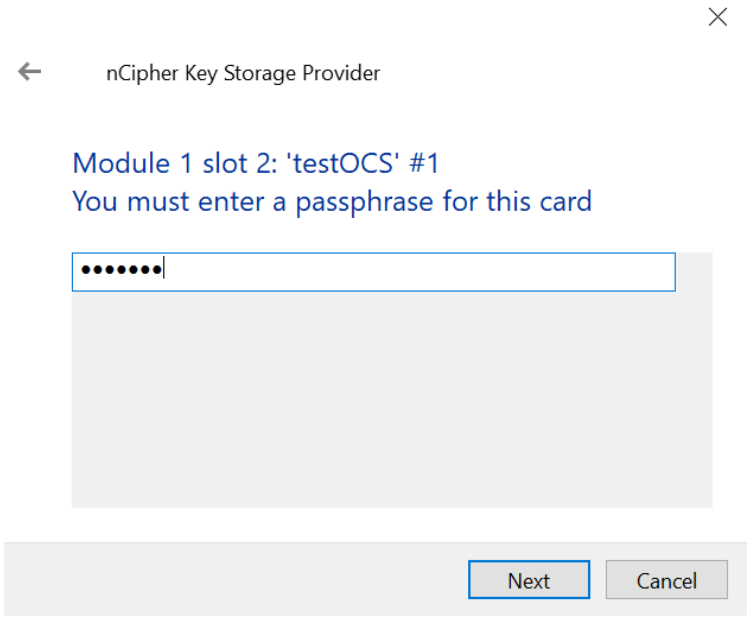
9. Insert the OCS card in the HSM or via the TVB is using remote administration. Then select **Next**.



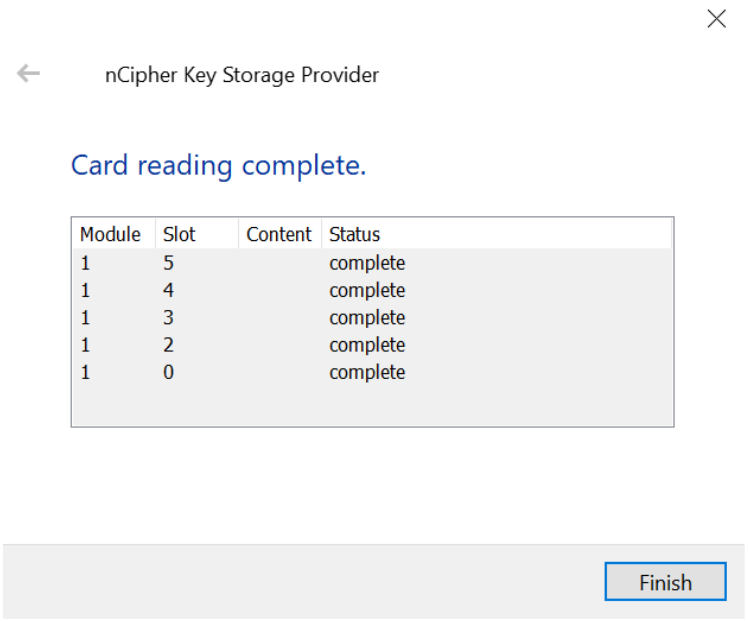
10. Select the HSM and then select **Finish**.



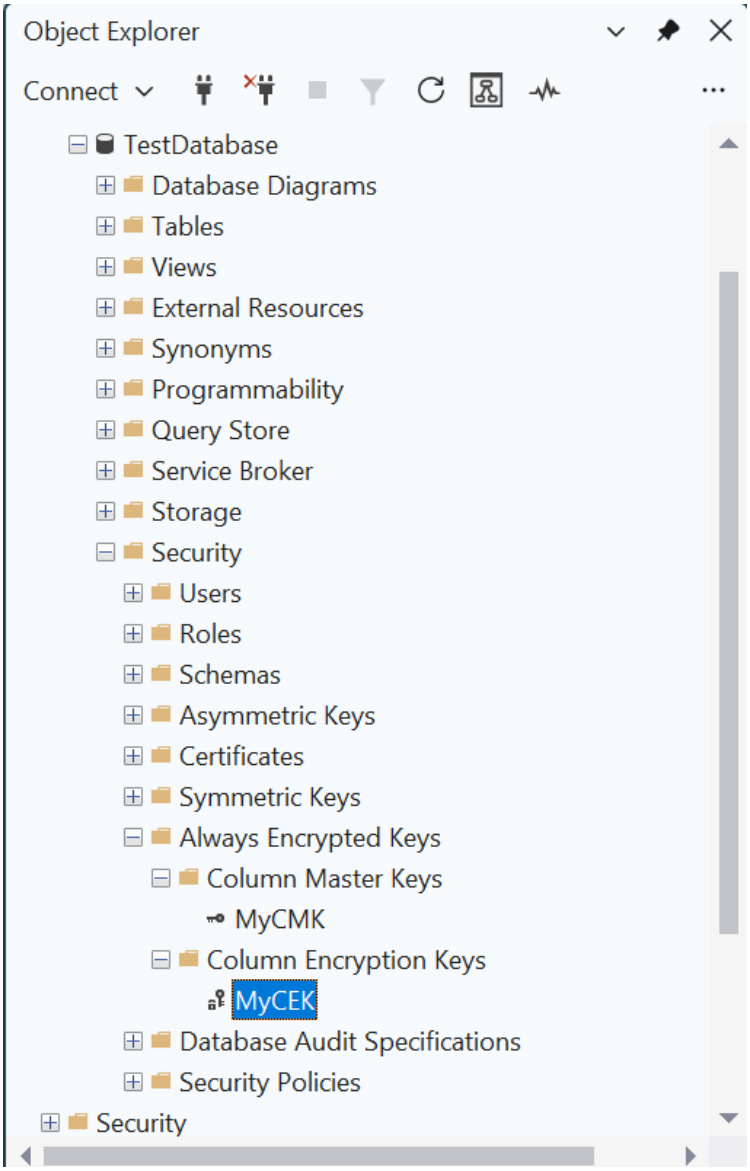
11. Enter the passphrase and then select **Next**.



12. Select **Finish** after the OCS card reading completes.



The newly-created **MyCEK** is in the database under **Security** > **Always Encrypted Keys** > **Column Encryption Keys**.



4.3. Generate MyCMK and MyCEK with PowerShell

To generate MyCMK and MyCEK with PowerShell:

1. Delete MyCEK and MyCMK in that order created above by right-clicking each key and selecting **Delete**.
2. Launch PowerShell and run the `Generate_MyCMK_and_MyCEK.ps1` script (below).

```
# Import the SqlServer module.
Import-Module SqlServer

# Connect to database.
$ConnectionString = "Data Source=<SQLSERVER_HOST>;Initial
Catalog=<DATABASE_NAME>;Trusted_Connection=True;MultipleActiveResultSets=False;Encrypt=True;TrustServerCert
ificate=True;Packet Size=4096;Application Name=`"Microsoft SQL Server Management Studio`""
$Database = Get-SqlDatabase -ConnectionString $ConnectionString
```

```
# Create a SqlColumnMasterKeySettings object for your column master key.
$cmkSettings = New-SqlCngColumnMasterKeySettings -CngProviderName "nCipher Security World Key Storage
Provider" -KeyName "AECMK"

# Create column master key metadata in the database.
New-SqlColumnMasterKey -Name "MyCMK" -InputObject $Database -ColumnMasterKeySettings $cmkSettings

# Generate a column encryption key, encrypt it with the column master key and create column encryption key
metadata in the database.
New-SqlColumnEncryptionKey -Name "MyCEK" -InputObject $Database -ColumnMasterKey "MyCMK"
```

The command line is:

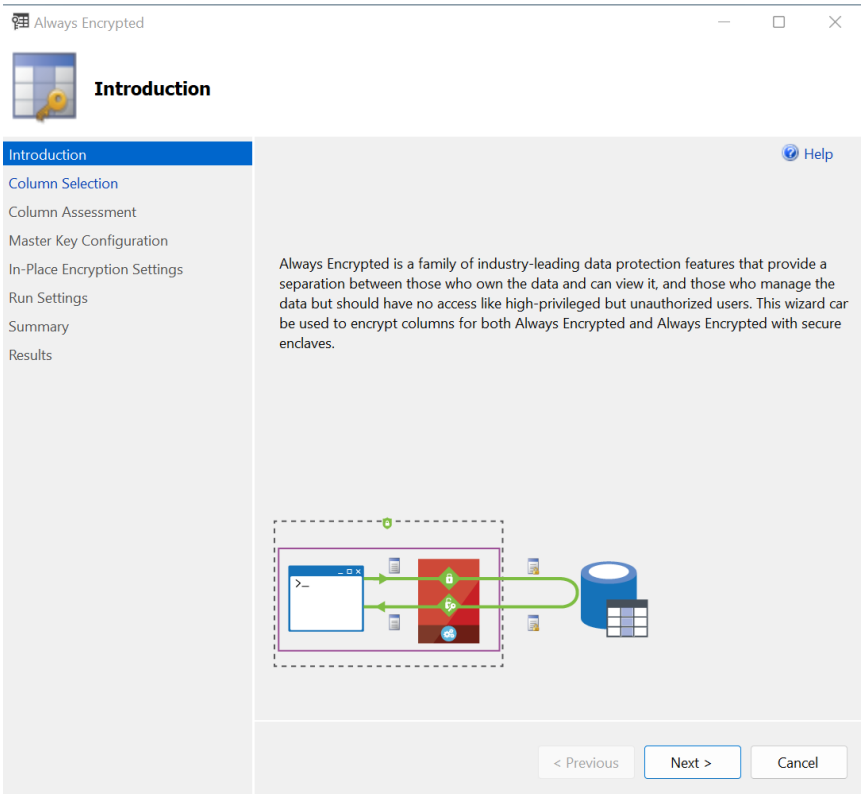
```
> PowerShell -ExecutionPolicy Bypass -File Generate_MyCMK_and_MyCEK.ps1

Name
-----
MyCMK
MyCEK
```

3. Present the OCS, select the HSM, and enter the passphrase.
4. Check the newly-created **MyCMK** and **MyCEK** are present.

Chapter 5. Encrypt or decrypt a column with SSMS

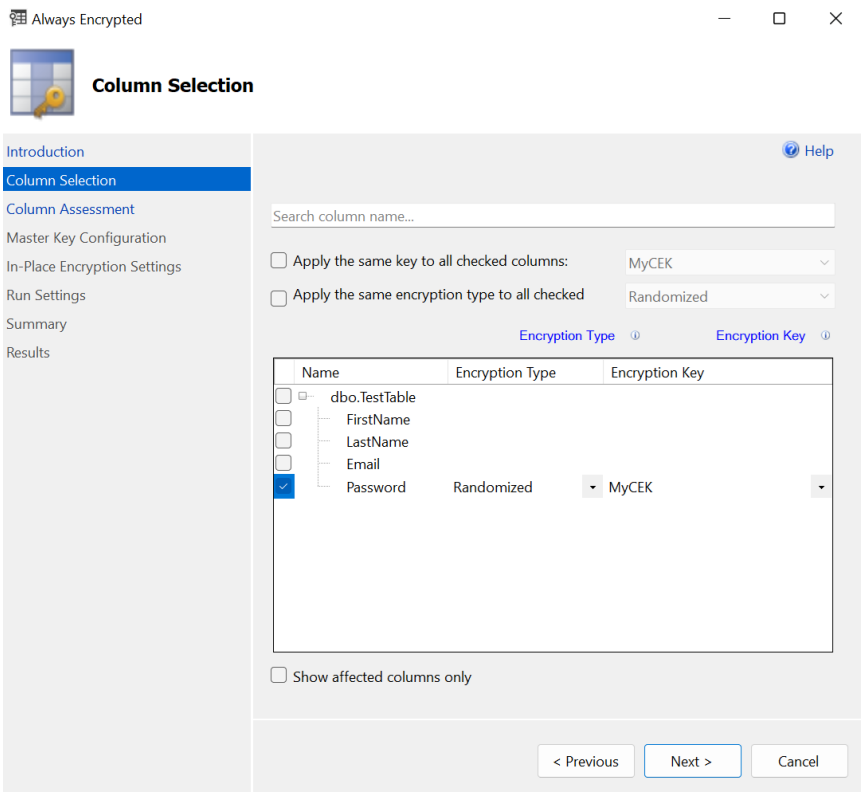
5.1. Encrypt a column

1. Log in to the client with the <domain>\dbuser account.
2. Launch **Microsoft SQL Server Management Studio**.
3. Connect to the database on the remote SQL Server instance using a **Connection String** in which **Column Encryption Setting=Enabled** (the default setting).
4. In the **Object Explorer**, expand **Databases** > **TestDatabase** > **Tables** > **dbo.TestTable**.
5. Right-select **dbo.TestTable** and select **Always Encrypted Wizard...**.
6. In the **Introduction** window, select **Next**.

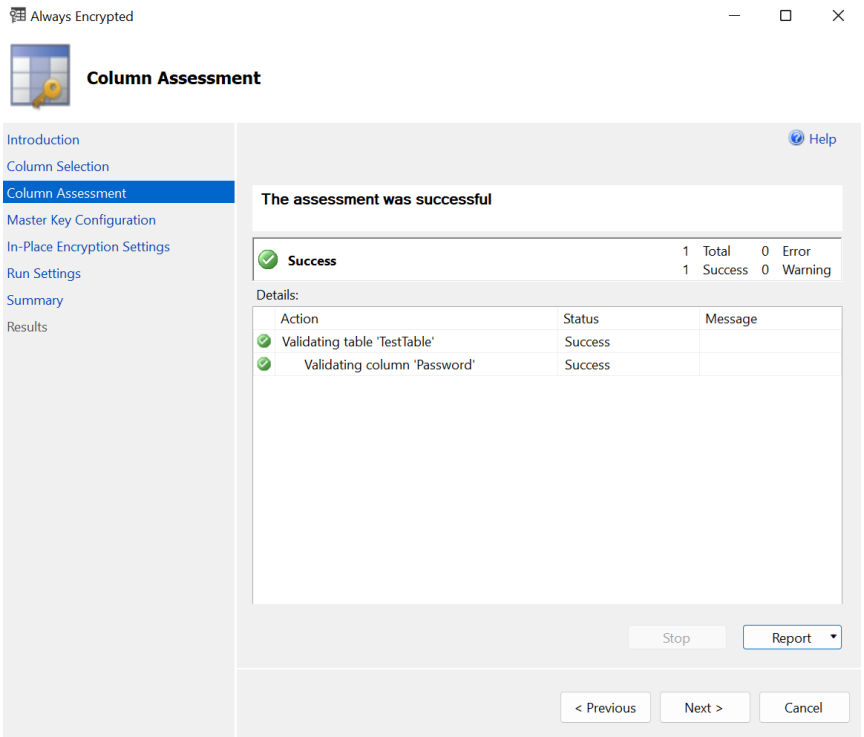


7. In the **Column Selection** window, select one or more columns to encrypt, select the **Encryption Type**, and **Encryption Key**, and then select **Next**.

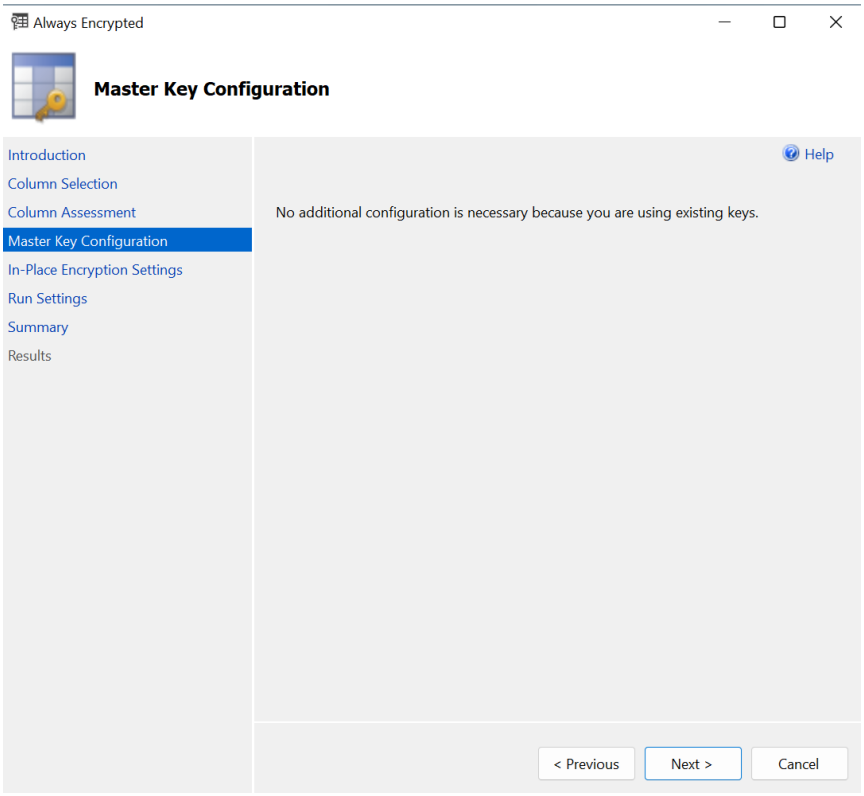
   For example:



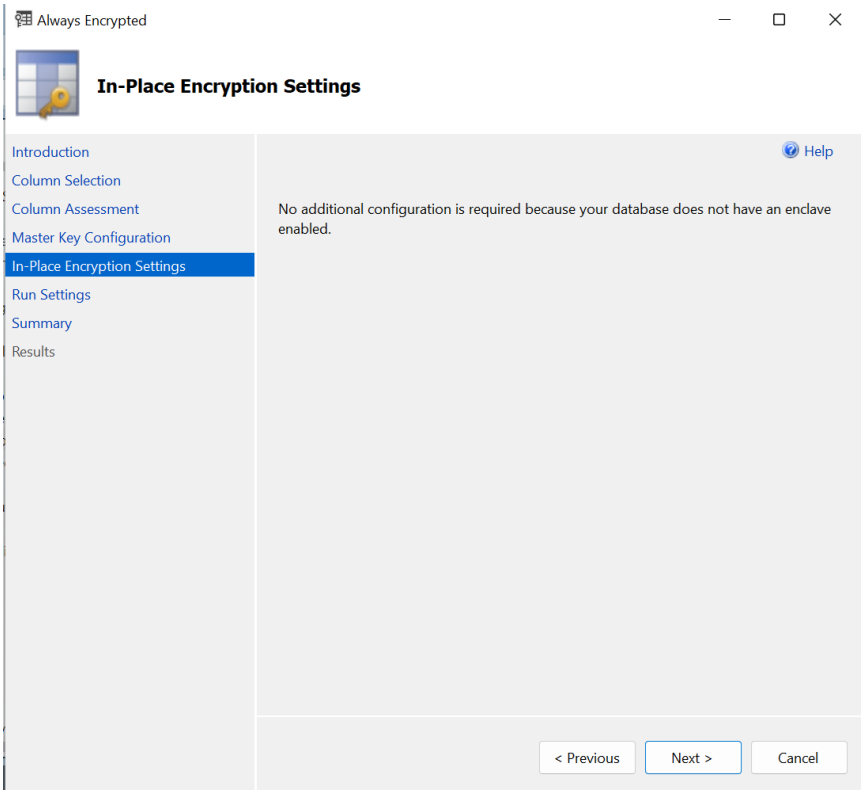
8. In the **Column Assessment** window, select **Next**.



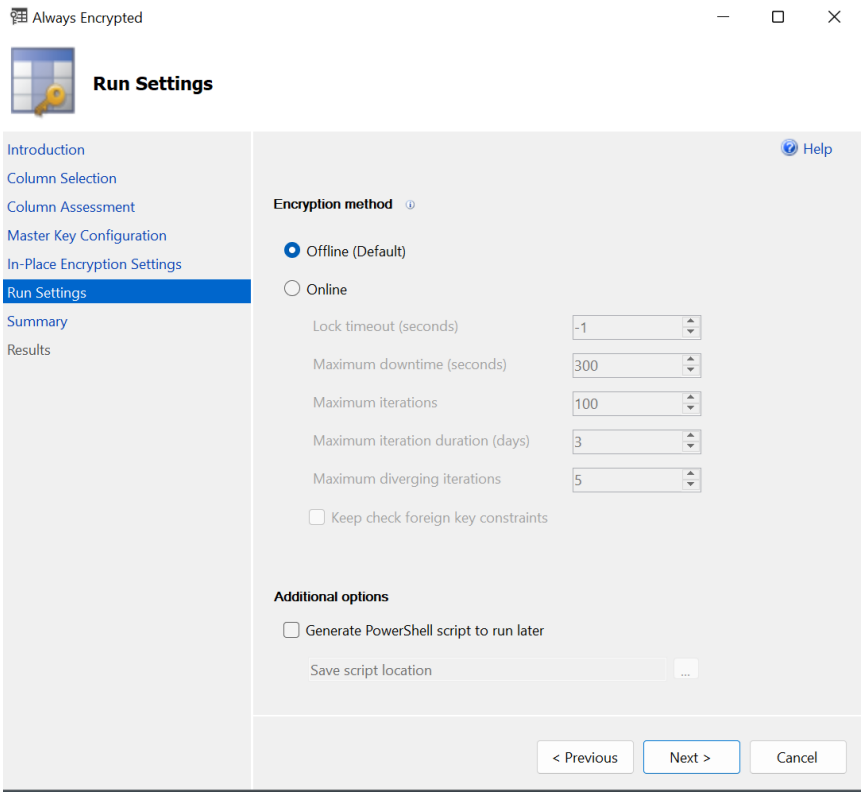
9. In the **Master Key Configuration** window, select **Next**.



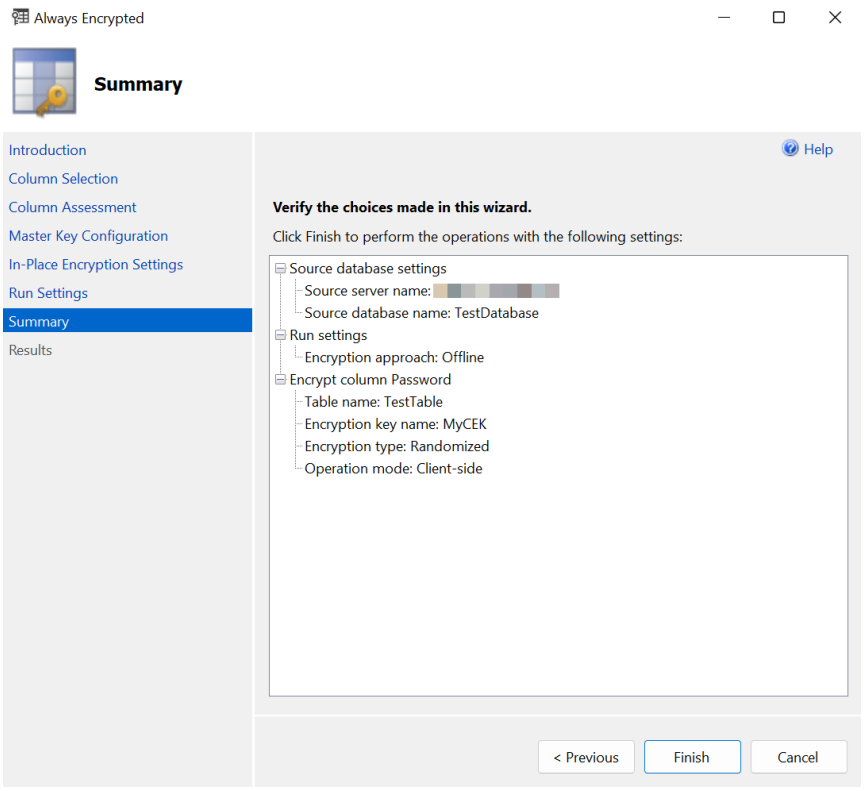
10. In the **In-Place Encryption Settings** window, select **Next**.



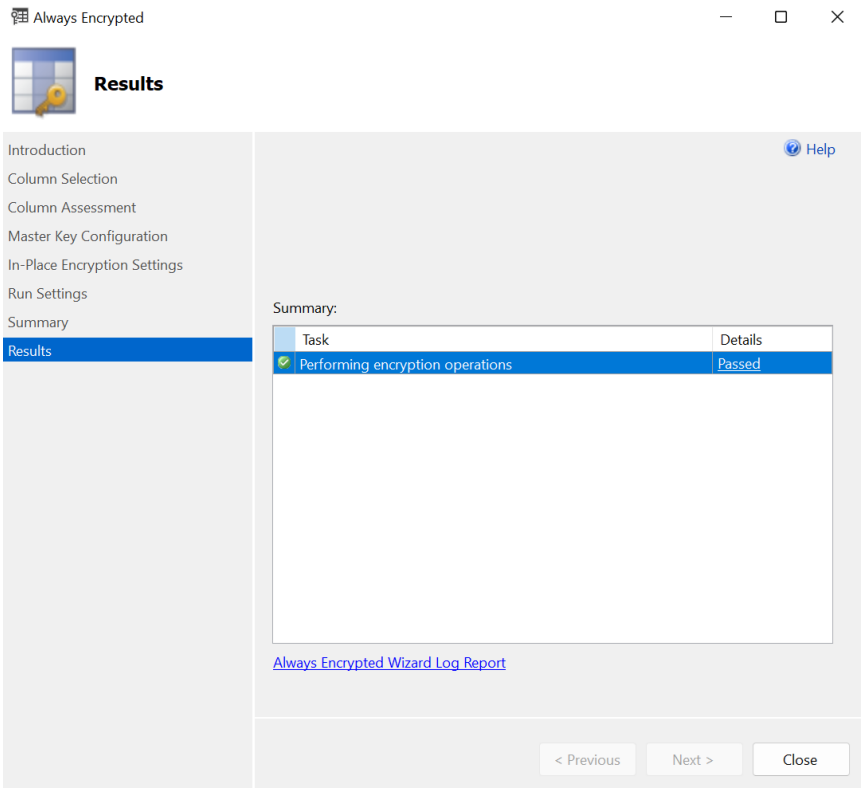
11. In the **Run Settings** window, select **Next**.



12. In the **Summary** window, verify the configuration choices, then select **Finish**.



13. Present the OCS, select the HSM, and enter the passphrase.
14. Check that **Passed** appears in the **Details** column of the **Results** window.

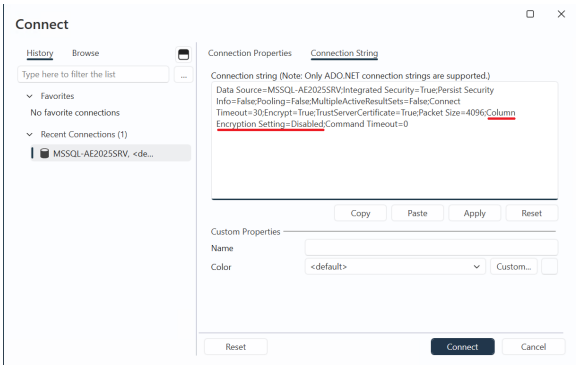


The column is encrypted in the SQL server, but it shows as clear text on the **Microsoft SQL Server Management Studio** GUI on the client. This is because **Always Encrypted** is performing the decryption at the client site.

15. Select **Close**.

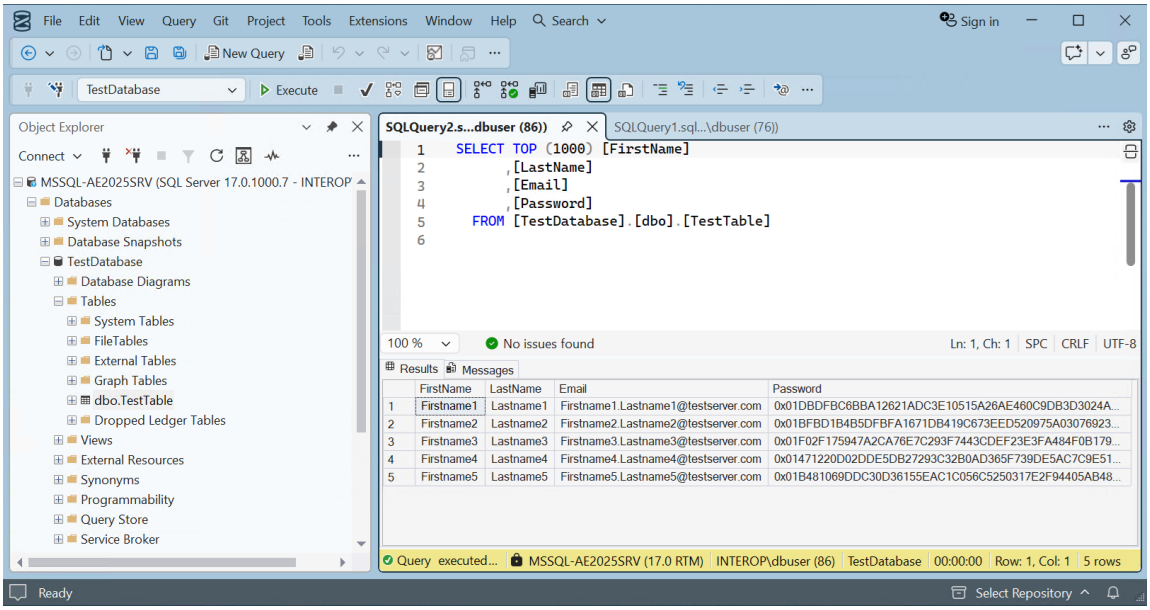
5.2. View an encrypted column

1. Connect to the SQL server instance with **Column Encryption Setting=Disabled** to view the encrypted data stored on the server.

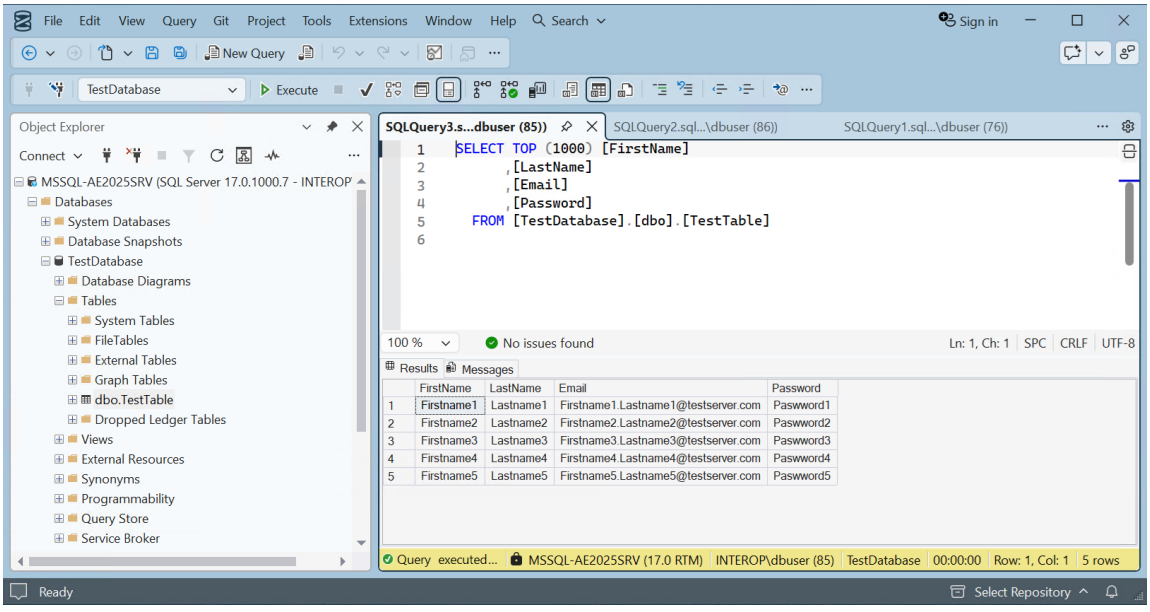


2. Right-click **dbo.Table** and select **Select Top 1000 Rows**. The column that was chosen

for encryption now appears as ciphertext, that is, as an encrypted value.



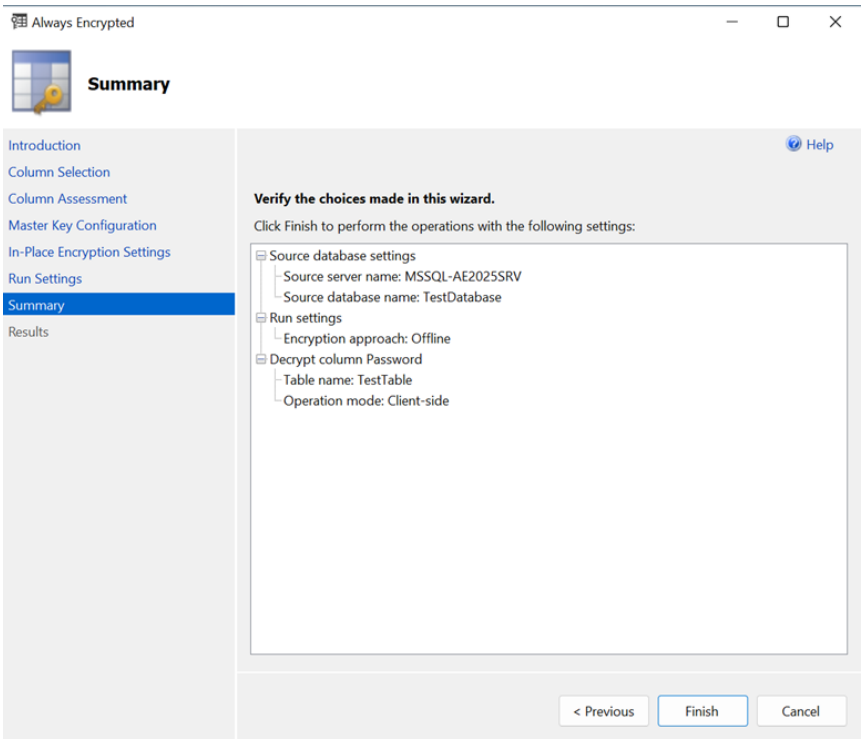
3. Reconnect to the SQL server instance with **Column Encryption Setting=Enabled**.
4. Present the OCS, select the HSM, and enter the passphrase.
5. Right-click **dbo.Table** and select **Select Top 1000 Rows**. The column that was chosen for encryption is now being decrypted by **Always Encrypted** with the key protected by the nShield HSM.



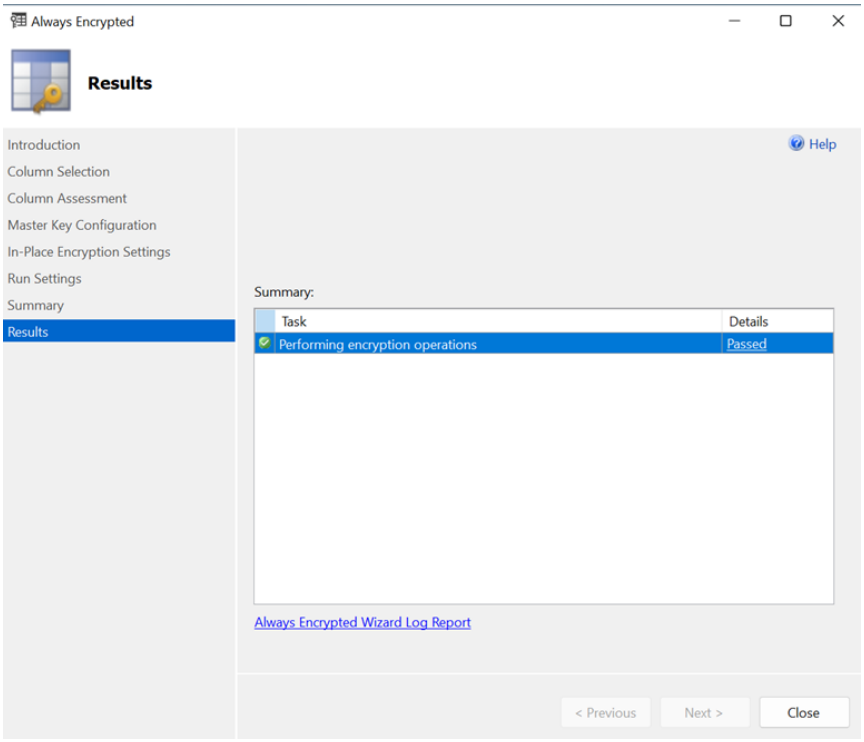
5.3. Remove column encryption

1. In the **Object Explorer**, expand **Databases** > **TestDatabase** > **Tables** > **dbo.TestTable**.
2. Right-select **dbo.TestTable** and select **Always Encrypted Wizard...**.

3. In the **Introduction** window, select **Next**.
4. In the **Column Selection** window, for **Encryption Type** select **Plaintext**, then select **Next**.
5. In the **Column Assessment** window, select **Next**.
6. In the **Master Key Configuration** window, select **Next**.
7. In the **In-Place Encryption Settings** window, select **Next**.
8. In the **Run Settings** window, select **Next**.
9. In the **Summary** window, verify the configuration choices, then select **Finish**.



10. Present the OCS, select the HSM, and enter the passphrase.
11. Check that **Passed** appears in the **Details** column of the **Results** window.



The column has been decrypted in the SQL server. To view the plaintext data stored on the SQL Server instance, reconnect to the server with **Column Encryption Setting=Disabled**. For instructions, see View an encrypted column.

12. Select **Close**.

Chapter 6. Encrypt or decrypt a column with PowerShell

6.1. Encrypt a column

1. Log in to the client using the <domain>\dbuser account.
2. Launch PowerShell and run the `Encrypt_Column_Named_Password.ps1` script (below).

```
# Import the SqlServer module.
Import-Module SqlServer -MinimumVersion 22.0.59

# Set up connection and database SMO objects
$sqlConnectionString = "Data Source=<SQLSERVER_HOST>;Initial Catalog=<DATABASE_NAME>;Integrated
Security=True;Multiple Active Result Sets=False;Connect Timeout=30;Encrypt=True;Trust Server
Certificate=True;Packet Size=4096;Application Name=`"Microsoft SQL Server Management Studio`""
$smoDatabase = Get-SqlDatabase -ConnectionString $sqlConnectionString

# Change encryption schema
$encryptionChanges = @()

# Add changes for table [dbo].[TestTable]
$encryptionChanges += New-SqlColumnEncryptionSettings -ColumnName dbo.TestTable.Password -EncryptionType
Randomized -EncryptionKey "MyCEK"
Set-SqlColumnEncryption -ColumnEncryptionSettings $encryptionChanges -InputObject $smoDatabase
```

   The command line is:

```
> PowerShell -ExecutionPolicy Bypass -File Encrypt_Column_Named_Password.ps1
```

3. Present the OCS, select the HSM, and enter the passphrase.
4. Launch **Microsoft SQL Server Management Studio**. Do as indicated in View an encrypted column to verify the column has been encrypted.

6.2. Remove column encryption

1. Log in to the client using the <domain>\dbuser account.
2. Launch PowerShell and run the `Decrypt_Column_Named_Password.ps1` script (below).

```
# Import the SqlServer module.
Import-Module SqlServer -MinimumVersion 22.0.59

# Set up connection and database SMO objects
$sqlConnectionString = "Data Source=<SQLSERVER_HOST>;Initial Catalog=<DATABASE_NAME>;Integrated
Security=True;Multiple Active Result Sets=False;Connect Timeout=30;Encrypt=True;Trust Server
Certificate=True;Packet Size=4096;Application Name=`"Microsoft SQL Server Management Studio`""
$smoDatabase = Get-SqlDatabase -ConnectionString $sqlConnectionString

# Change encryption schema
```

```
$encryptionChanges = @()

# Add changes for table [dbo].[TestTable]
$encryptionChanges += New-SqlColumnEncryptionSettings -ColumnName dbo.TestTable.Password -EncryptionType
Plaintext
Set-SqlColumnEncryption -ColumnEncryptionSettings $encryptionChanges -InputObject $smoDatabase
```

The command line is:

```
> PowerShell -ExecutionPolicy Bypass -File Decrypt_Column_Named_Password.ps1
```

3. Present the OCS, select the HSM, and enter the passphrase.
4. Launch **Microsoft SQL Server Management Studio**. Do as indicated in View an encrypted column to verify the column has been decrypted.

Chapter 7. Test access to Always Encrypted keys by another user

To test access to Always Encrypted keys by another user:

1. Log in to the client using the <domain>\dbuser2 account.
2. Launch **Microsoft SQL Server Management Studio**.
3. Connect to the database on the remote SQL server, enabling **Always Encrypted**.
4. Present the OCS, select the HSM, and enter the passphrase.
5. Perform operations on the TestDatabase, which is possible since <domain>\dbuser2 has access to the same MyCMK and MyCEK keys created by <domain>\dbuser.

Chapter 8. Supported PowerShell SqlServer cmdlets

PowerShell cmdlet	Description
`Add-SqlColumnEncryptionKeyValue`	Adds a new encrypted value for an existing column encryption key object in the database.
`Complete-SqlColumnMasterKeyRotation`	Completes the rotation of a column master key.
`Get-SqlColumnEncryptionKey`	Returns all column encryption key objects defined in the database, or returns one column encryption key object with the specified name.
`Get-SqlColumnMasterKey`	Returns the column master key objects defined in the database, or returns one column master key object with the specified name.
`Invoke-SqlColumnMasterKeyRotation`	Initiates the rotation of a column master key.
`New-SqlAzureKeyVaultColumnMasterKeySettings`	Creates a `SqlColumnMasterKeySettings` object describing an asymmetric key stored in Azure Key Vault.
`New-SqlCngColumnMasterKeySettings`	Creates a `SqlColumnMasterKeySettings` object describing an asymmetric key stored in a key store supporting the Cryptography Next Generation (CNG) API.
`New-SqlColumnEncryptionKey`	Creates a new column encryption key object in the database.
`New-SqlColumnEncryptionKeyEncryptedValue`	Produces an encrypted value of a column encryption key.
`New-SqlColumnEncryptionSettings`	Creates a new `SqlColumnEncryptionSettings` object that encapsulates information about a single column's encryption, including CEK and encryption type.

PowerShell cmdlet	Description
`New-SqlColumnMasterKey`	Creates a new column master key object in the database.
`New-SqlCspColumnMasterKeySettings`	Creates a `SqlColumnMasterKeySettings` object describing an asymmetric key stored in a key store with a Cryptography Service Provider (CSP) supporting Cryptography API (CAPI).
`Remove-SqlColumnEncryptionKey`	Removes the column encryption key object from the database.
`Remove-SqlColumnEncryptionKeyValue`	Removes an encrypted value from an existing column encryption key object in the database.
`Remove-SqlColumnMasterKey`	Removes the column master key object from the database.
`Set-SqlColumnEncryption`	Encrypts, decrypts or re-encrypts specified columns in the database.

The full list of cmdlets and additions to the `SqlServer` module can be found in the Microsoft Online Documentation.

Chapter 9. Additional resources and related products

9.1. nShield HSMs

9.2. nShield as a Service

9.3. Entrust products

9.4. nShield product documentation