Member of
Microsoft Intelligent
Security Association

Microsoft Security

Microsoft SQL Server

nShield® HSM Integration Guide

2023-12-05

Table of Contents

Chapter 1. Introduction

This document describes how to integrate Microsoft SQL Server with the nShield Database Security Option Pack (nDSOP V2.1) using an Entrust nShield hardware security module (HSM) as a Root of Trust.

1.1. Product configurations

Entrust tested the integration with the following versions:

Product	Version
Base OS	Windows Server Datacenter 2019 and 2022
SQL Server	Mircrosoft SQL Server Enterprise 2016, 2019, and 2022
Microsoft SQL Server Management Studio	v19.1

1.2. Supported nShield hardware and software versions

Entrust tested the integration with the following nShield HSM hardware and software versions, and SQLEKM provider:

Product	Security World	Firmware	Netimage
Connect XC	12.60.11	12.50.11 FIPS Certified	12.60.10
Connect XC	12.80.4	12.50.11 FIPS Certified	12.80.4
Connect XC	12.80.4	12.72.1 FIPS Certified	12.80.5
nShield 5c	13.3.2	13.2.2 FIPS Pending	13.3.2

Supported nShield SQLEKM provider:

Product	Version
nDSOP	hotfix-Z166345-TAC1058

1.3. Supported nShield functionality

Functionality	Support
FIPS 140 Level 3	Yes
Key Management	Yes
Key Generation	Yes
Key Recovery	Yes
1 of N Card Set	Yes
Softcards	Yes
Module Only Key	No
Fail Over	Yes
Load Balancing	Yes
nSaaS	Yes

1.4. Requirements

Be familiar with:

- The Microsoft SQL Server features and documentation.
- The Microsoft SQL Server Management Studio features and documentation.
- The T-SQL language. The minimum requirement for T-SQL is a basic understanding of SQL tasks such as creating a database or tables.
- Database security concepts and practices.
- The documentation for the HSM.

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

1.5. Terms

Acronym	Definition
SQLEKM	SQL Server Extensible Key Management
TDEKEK	TDE Key Encrpytion Key
TDEDEK	TDE Database Encrpytion Key

Chapter 2. Setup

Prerequisites:

- A Windows Server with Microsoft SQL server.
- SQL Server Management Studio installed.
- The database `TestDatabase` has been created and is available for the integration.

Perform the following steps:

1. Install the Security World software and create a Security World
2. Install the nShield nDSOP
3. Create the Operator Card Set (OCS) or Softcard
4. Enable EKM and register the SQLEKM provider
5. Verify the SQLEKM provider configuration
6. Create the user SQL Server credential

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

To install the Security World software and create a Security World:

1. Install the Security World software by double-clicking on the `SecWorld_Windows-xx.xx.xx.iso` file. For detailed instructions, see the *Installation Guide* and the *User Guide* for the HSM available from the installation disc.
2. Add the Security World utilities path `C:\Program Files\nCipher\nfast\bin` to the Windows system path.
3. Open the firewall port 9004 for the HSM connections.
4. Install the nShield Connect HSM locally, remotely, or remotely via the serial console. See the following nShield Support articles, and the *Installation Guide* for the HSM:
    - How to locally set up a new or replacement nShield Connect
    - How to remotely set up a new or replacement nShield Connect
    - How to remotely set up a new or replacement nShield Connect XC Serial Console model

    Access to the Entrust nShield Support Portal is available to customers under maintenance. To request an account,

> contact nshield.support@entrust.com.

5. Open a command window and run the following to confirm that the HSM is `operational`:

```
> enquiry
Server:
 enquiry reply flags  none
 enquiry reply level  Six
 serial number        530E-02E0-D947 7724-8509-81E3 09AF-0BE9-53AA 9E10-03E0-D947
 mode                 operational
...
Module #1:
 enquiry reply flags  none
 enquiry reply level  Six
 serial number        530E-02E0-D947
 mode                 operational
 ...
```

6. Create a Security World if one does not already exist or copy an existing one. Follow your organization's security policy for this. Create extra ACS cards as spares in case of a card failure or lost. These cannot be duplicated after the Security World is created.
7. Confirm that the Security World is `usable`:

```
> nfkminfo
World
 generation  2
 state       0x37270008 Initialised Usable ...
 ...
Module #1
 generation 2
 state      0x2 Usable
 ...
```

2.2. Install the nShield nDSOP

To install the nShield nDSOP:

1. Mount the `nDSOP_Windows-x.x.x.iso` file.
2. Double-click the `setup` file and follow the instructions.

2.3. Create the Operator Card Set (OCS) or Softcard

The OCS or Softcard and associated passphrase will be used to authorize access to specific keys protected by the SQLEKM provider. Typically, an organization's security policies dictate the use of one or the other.

2.3.1. Create the OCS

A SQL Server credential (as used for EKM) maps one protecting token to one stored passphrase. It can store information for only one token at a time. An OCS does have a quorum of one.

Recovering from a power failure requires the OCS to be inserted in the HSM or the TVD.

1. Ensure the cardlist file located in the path `C:\ProgramData\nCipher\Key Management Data\config\` contains the serial number of the card(s) to be presented or the wildcard value.
2. Open a command window as administrator.
3. Execute the following command. Enter a passphrase or password at the prompt. Follow your organization's security policy for the OCS values. After an OCS card set has been created, the cards cannot be duplicated. Notice that `slot 2`, remote via a Trusted Verification Device (TVD), is used to present the card.

```
> 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: blank card
 Module 1 slot 3: empty
 Module 1 slot 2:- passphrase specified - writing card
Card writing complete.

cardset created; hkltu = edb3d45a28e5a6b22b033684ce589d9e198272c2
```

   Add the `-p` (persistent) option to the command above if you want:

   - to be able to encrypt/decrypt the database after the OCS card has been removed from the HSM front panel slot or from the TVD.
   - the ability to persist after a reboot.

   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. If the TVD loses connection to the Remote Administration client the database will be inaccessible.

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
edb3d45a28e5a6b22b033684ce589d9e198272c2 1/1  none-NL testOCS
```

2.3.2. Create the Softcard

A SQL Server credential (as used for EKM) maps one protecting token to one stored passphrase. Softcards are singular and do not have a quorum, so the SQL Server credential matches them quite well.

Unlike OCS protection, which requires a smart card and a passcode, a softcard does not require additional input for recovery after a power failure.

1. Ensure the `C:\Program Files\nCipher\nfast\cknfastrc` file exists with the following content. Otherwise, create it.

```
> type "C:\Program Files\nCipher\nfast\cknfastrc"
CKNFAST_LOADSHARING=1
```

2. Execute the following command. Enter a passphrase at the prompt.

```
> ppmk -n testSC

Enter new pass phrase:
Enter new pass phrase again:
New softcard created: HKLTU 925f67e72ea3c354cae4e6797bde3753d24e7744
```

3. Verify the Softcard created:

```
> nfkminfo -s
SoftCard summary - 1 softcards:
 Operator logical token hash              name
 925f67e72ea3c354cae4e6797bde3753d24e7744 testSC
```

   The `rocs` utility shows the OCS and Softcard created:

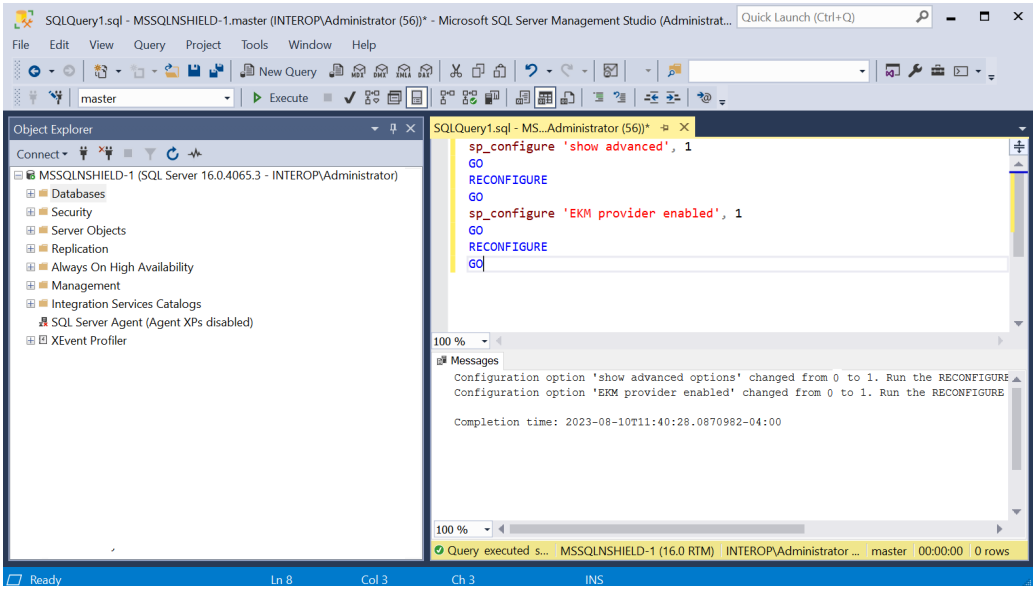
```
> rocs
'rocs' key recovery tool
Useful commands: 'help', 'help intro', 'quit'.
rocs> list cardset
No. Name                    Keys (recov) Sharing
  1 testOCS                 0 (0)        1 of 5
  2 testSC                  0 (0)        (softcard)
rocs> quit
```

2.4. Enable EKM and register the SQLEKM provider

To enable EKM and register the SQLEKM provider:

1. Launch the SQL Server Management Studio GUI.
2. Enable EKM by executing the following query:

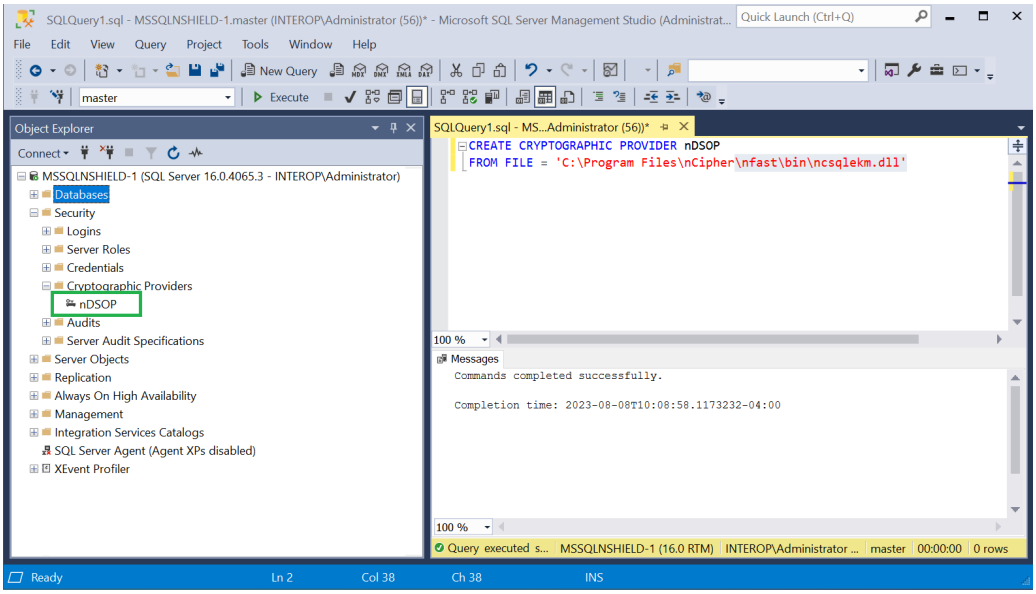
```
sp_configure 'show advanced', 1
GO
RECONFIGURE
GO
sp_configure 'EKM provider enabled', 1
GO
RECONFIGURE
GO
```



3. Register the SQLEKM provider with the SQL Server by executing the following query:

```
CREATE CRYPTOGRAPHIC PROVIDER nDSOP
    FROM FILE = 'C:\Program Files\nCipher\nfast\bin\ncsqlekm.dll'
```

4. Check the SQLEKM provider is listed in the SQL Server Management Studio GUI. Go to **Security > Cryptographic Providers**. **nDSOP** should be visible. Right-click it to verify that it is enabled.

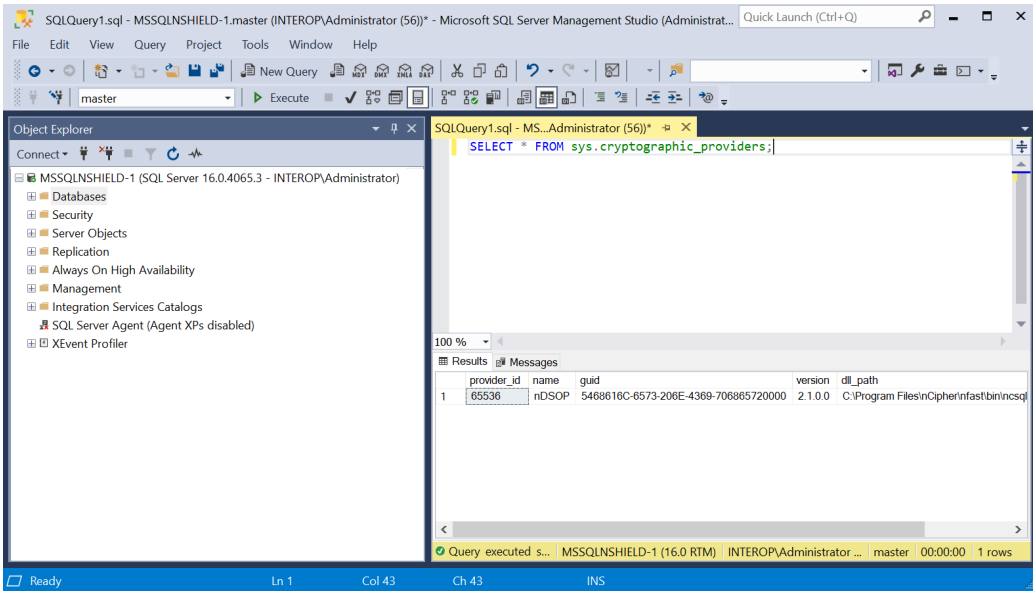


2.5. Verify the SQLEKM provider configuration

To verify the SQLEKM provider configuration:

1. Run the following query:

```
SELECT * FROM sys.cryptographic_providers;
```



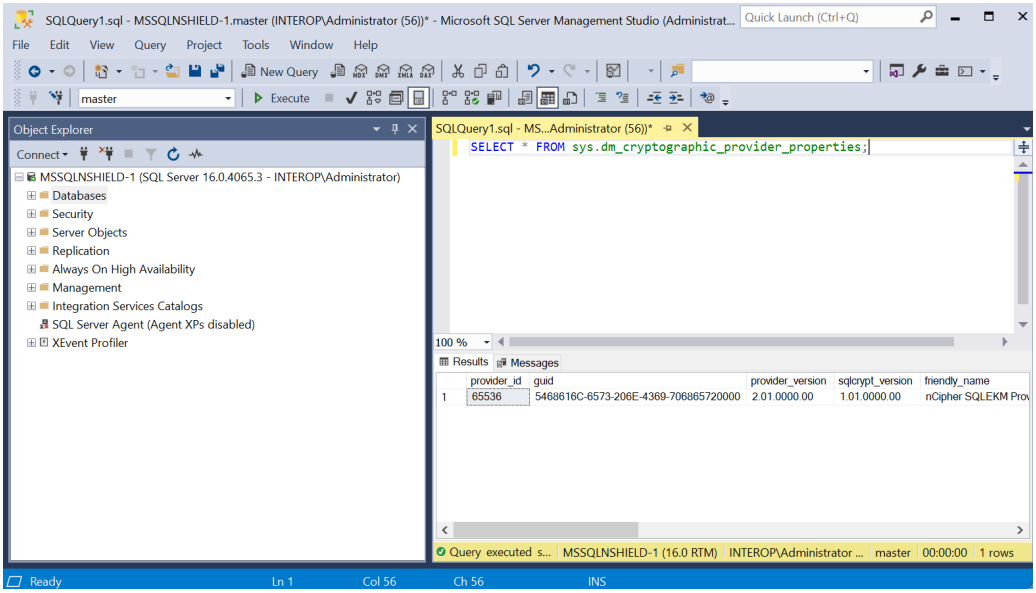
Verify the following:

- The version matched that of the nDSOP installation `iso`.
- Path to `dll` is correct.

- **is_enabled** column set to **1**.

2. Run the following query:

```
SELECT * FROM sys.dm_cryptographic_provider_properties;
```

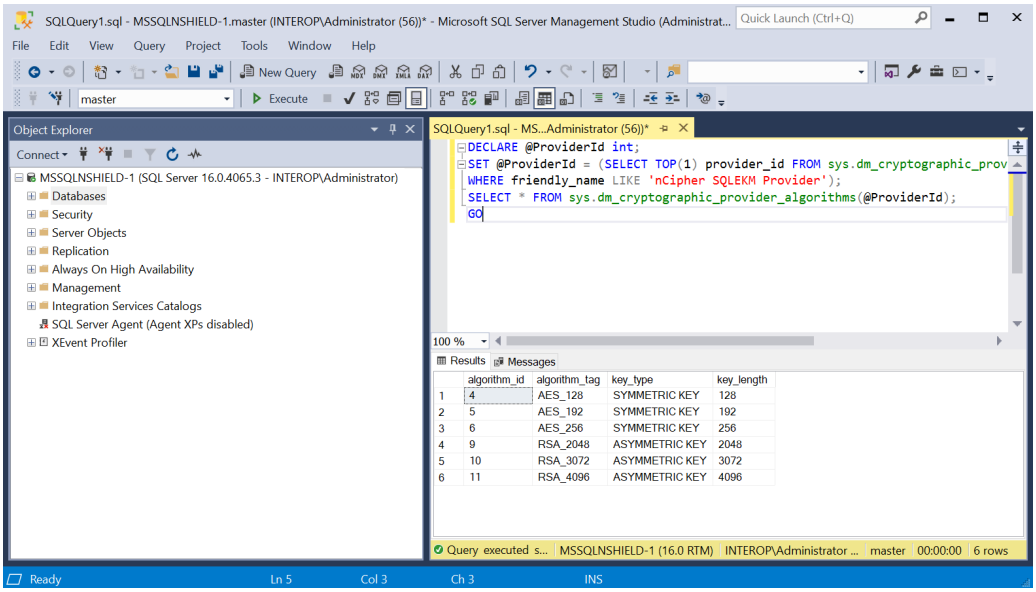


Verify the following:

Column	Value
friendly_name	nCipher SQLEKM Provider
authentication_type	BASIC
symmetric_key_support	1
asymmetric_key_support	1

3. Verify the supported cryptographic algorithms can be queried by running the following query:

```
DECLARE @ProviderId int;
SET @ProviderId = (SELECT TOP(1) provider_id FROM sys.dm_cryptographic_provider_properties
WHERE friendly_name LIKE 'nCipher SQLEKM Provider');
SELECT * FROM sys.dm_cryptographic_provider_algorithms(@ProviderId);
GO
```



Notice each key type has its set of valid algorithms.

Key Type	Algorithm
Symmetric	AES_128, AES_192, ASE_256
Asymmetric	RSA_2048, RSA_3072, RSA_4096

2.6. Create the user SQL Server credential

To create the user SQL Server credential:

1. Verify the OCS or Softcard created above:

```
> nfkminfo -c
Cardset list - 1 cardsets:  (P)ersistent/(N)ot, (R)emoteable/(L)ocal-only
 Operator logical token hash               k/n timeout  name
 edb3d45a28e5a6b22b033684ce589d9e198272c2  1/5  none-NL testOCS

> nfkminfo -s
SoftCard summary - 1 softcards:
 Operator logical token hash               name
 925f67e72ea3c354cae4e6797bde3753d24e7744  testSC
```

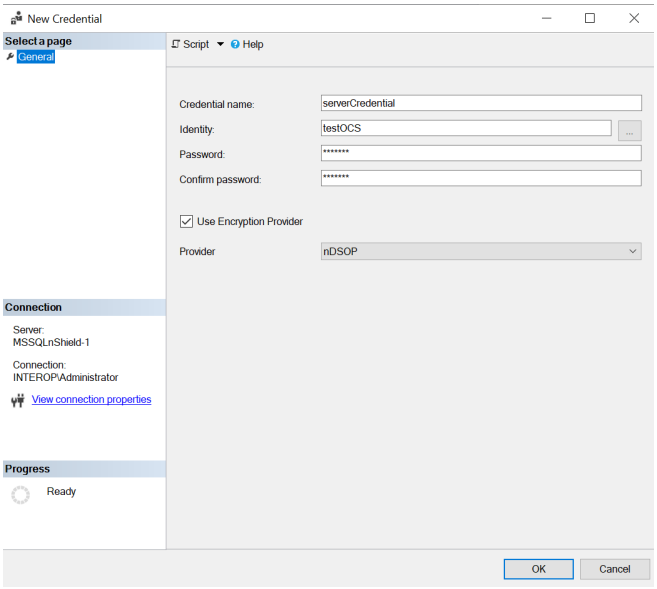
2. Navigate to **Security > Credentials** in SQL Server Management Studio.
3. Right-click **Credentials**, then select **New Credential**.
4. Under **New Credential**:
    a. Enter the **Credential name**.
    b. For **Identity**, enter the OCS card name.

c. Enter a **Password**, and confirm the password.

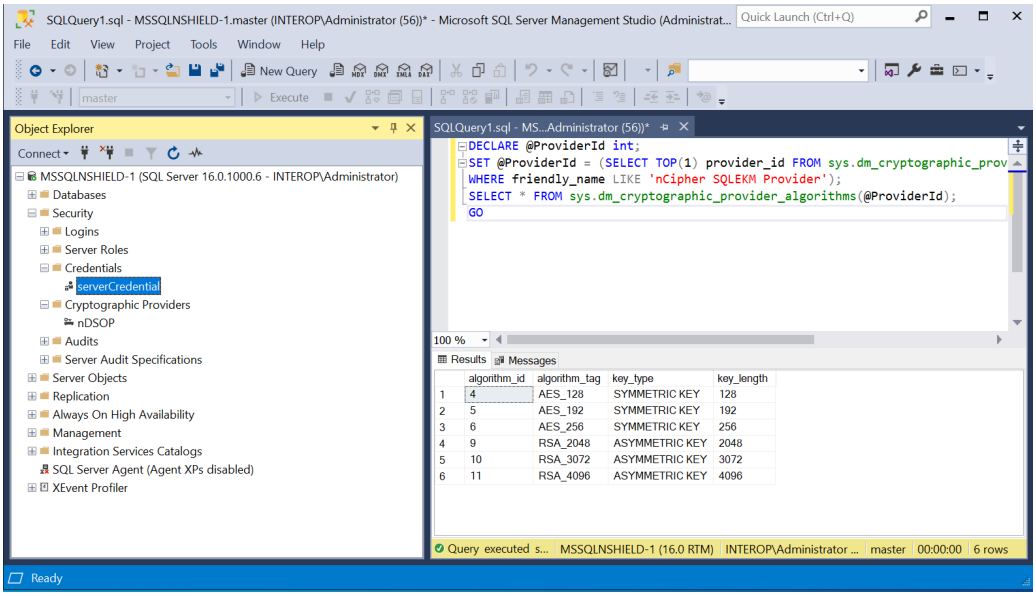
d. Select **Use Encryption Provider**.

e. For **Provider**, select **nDSOP**.

f. Select **OK**.

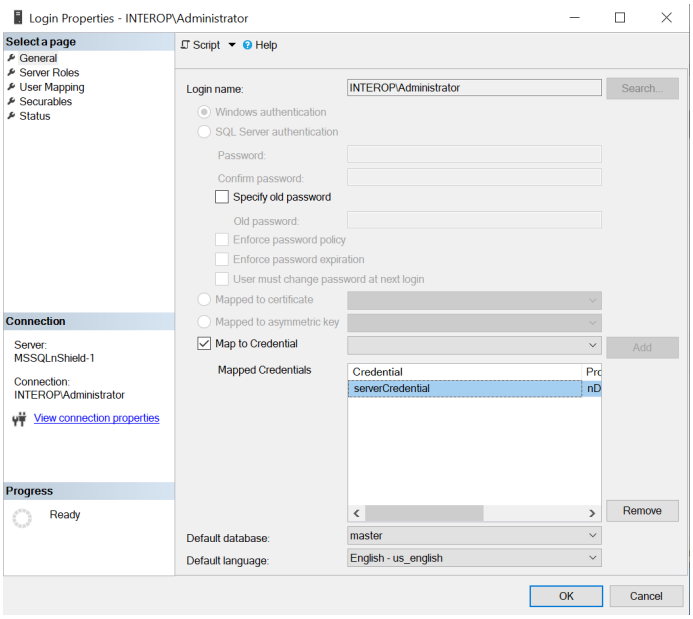


5. Verify the new credential in **Security > Credentials**. You may need to right-click and select **Refresh**.



6. Navigate to **Security > Logins**. Right-click the login used to access the TestDatabase and select **Properties**.

7. Check **Map to Credentials** in the dialog. Select the server credential created above in the drop-down to the right. Then select **Add**, and select **OK**.


Login Properties - INTEROP\Administrator
Select a page
General
Server Roles
User Mapping
Securables
Status
Script
Help
Login name:
INTEROP\Administrator
Search...
Windows authentication
SQL Server authentication
Password:
Confirm password:
Specify old password
Old password:
Enforce password policy
Enforce password expiration
User must change password at next login
Mapped to certificate
Mapped to asymmetric key
Map to Credential
Add
Mapped Credentials
Credential
Pro
serverCredential
nD
Remove
Connection
Server:
MSSQLnShield-1
Connection:
INTEROP\Administrator
View connection properties
Progress
Ready
Default database:
master
Default language:
English - us_english
OK
Cancel

Chapter 3. Configure TDE

The TDE Database Encryption Key (TDEDEK) is a symmetric key that is used to perform the actual encryption of the database and are unique to a given database. It is created by SQL Server and cannot be exported from the database, meaning it cannot be created or directly protected by the SQLEKM provider (nShield HSM).

The TDEDEK is protected within the database by encrypting it with a wrapping key. The wrapping key is called the TDE Key Encryption Key (TDEKEK). The TDEKEK is an asymmetric key protected by the SQLEKM provider in the nShield HSM. It is possible to have a single TDEKEK for multiple databases, or different TDEKEKs for different databases.

The TDEKEK must be created under the tdeLogin/tdeCredential. However, the current user does not have to use the tdeCredential, so long as the user credential is using the same OCS or Softcard as the tdeCredential.

1. Create a TDEKEK
2. Create a TDE login and credential
3. Create the TDEDEK and switch on encryption
4. Key rotation - Replace the TDEKEK
5. Key rotation - Replace the TDEDEK

3.1. Create a TDEKEK

To create a TDEKEK in the master database:

1. Insert the OCS in the HSM slot or TVD. If using Softcard protection, no action is needed.
2. Run the following query:

```
USE master;
CREATE ASYMMETRIC KEY "<name_of_key_in_database>"
FROM PROVIDER "<SQLEKM_provider>"
WITH
PROVIDER_KEY_NAME = '<name_of_key_in_SQLEKM_provider>',
CREATION_DISPOSITION = CREATE_NEW,
ALGORITHM = <asymmetric_algorithm_desc>;
GO
```

Where:

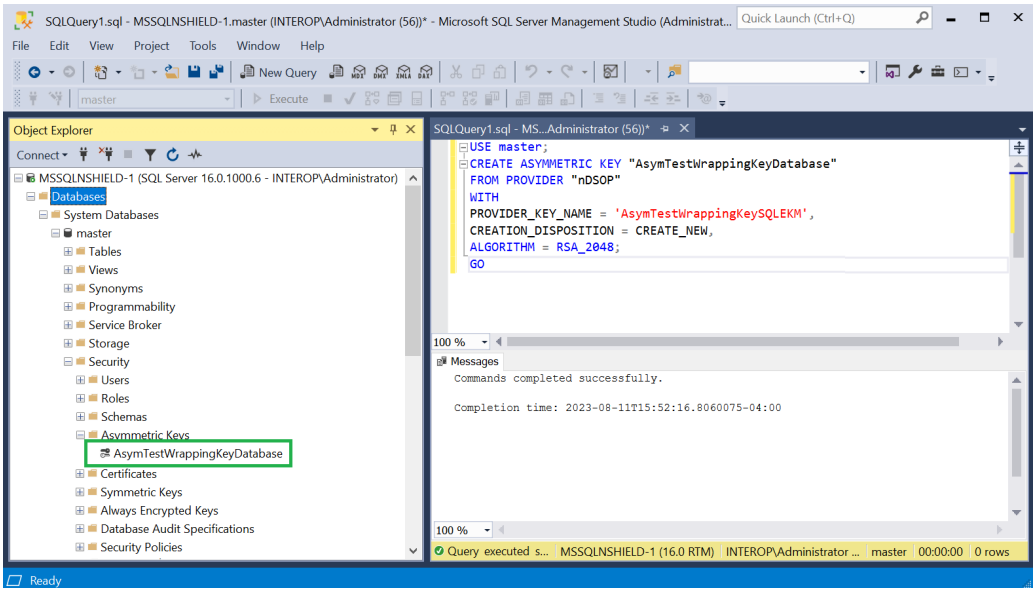
name_of_key_in_database	The name given to the key in the database.
name_of_key_in_SQLEKM_provider	The name given to the key in the SQLEKM provider.
asymmetric_algorithm_desc>	A valid asymmetric key algorithm descriptor.

See Verify the SQLEKM provider configuration.

For example:

```
USE master;
CREATE ASYMMETRIC KEY "AsymTestWrappingKeyDatabase"
FROM PROVIDER "nDSOP"
WITH
PROVIDER_KEY_NAME = 'AsymTestWrappingKeySQLEKM',
CREATION_DISPOSITION = CREATE_NEW,
ALGORITHM = RSA_2048;
GO
```

Notice the newly created key highlighted in the object explorer.



3. The key generated can also be verified using a CLI command:

```
> nfkminfo -l

Keys protected by cardsets:
 key_simple_sqlekm-edb3d45a28e5a6b22b033684ce589d9e198272c2-ecaaf2c3e8cb8f0dd3756678b757468a4de120c4
`AsymTestWrappingKeySQLEKM'
```

The `rocs` utility shows the names and protection methods of the keys.

```
> rocs
'rocs' key recovery tool
Useful commands: 'help', 'help intro', 'quit'.
rocs> list keys
  No. Name                      App        Protected by
    1 AsymTestWrappingKeySQLEK simple     testOCS
rocs> exit
```

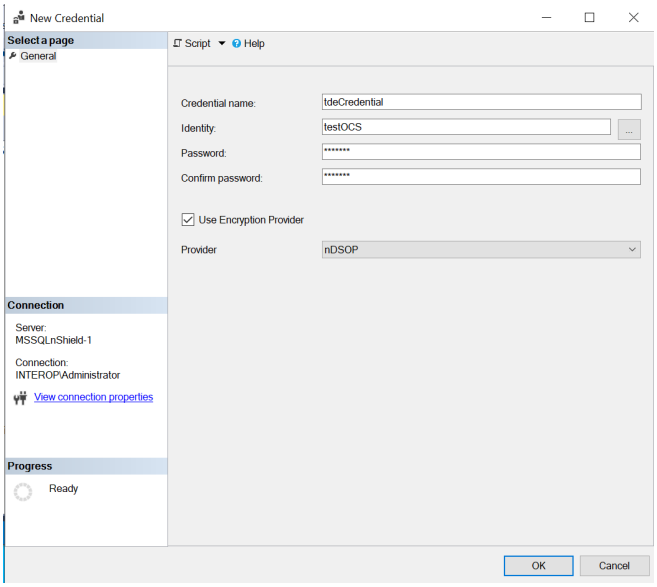
3.2. Create a TDE login and credential

A tdeLogin and tdeCredential allows an ordinary database user, who is fully authorized to use the database, but has no SQLEKM credentials of their own, to perform query operations using a TDE encrypted database. Without the tdeLogin and tdeCredential, then every user would need their own credentials. It is beyond the scope of this document to provide an example of how to use these credentials, only on how to create them.

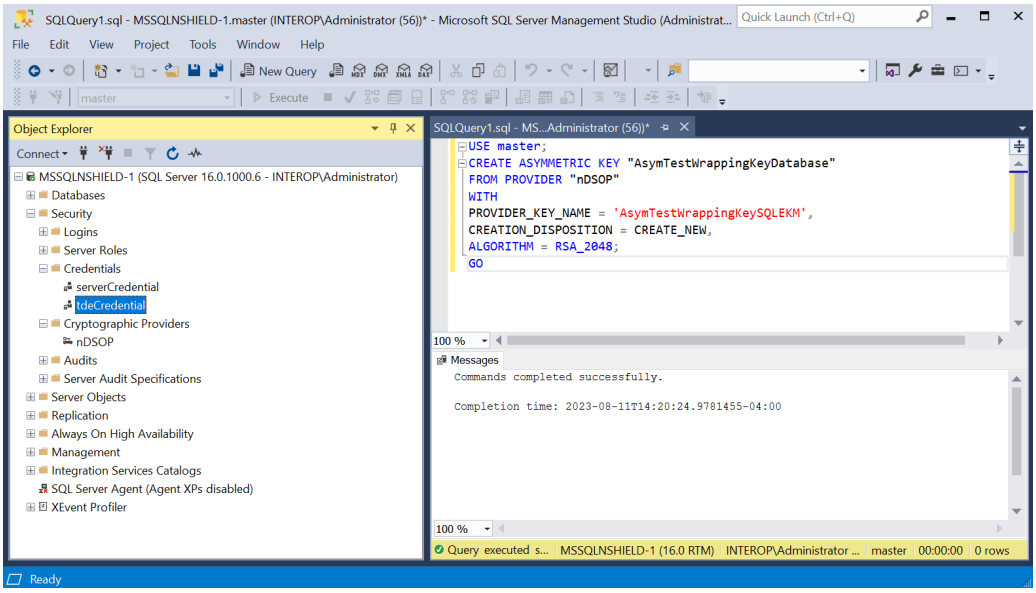
3.2.1. Create a TDE credential

To create a TDE credential:

1. In SQL Server Management Studio, navigate to **Security > Credentials**.
2. Right-click **Credentials**, then select **New Credential**.
3. Under **New Credential**:
    a. Enter the **Credential name**.
    b. For **Identity**, enter the OCS card name.
    c. Enter a **Password**, and confirm the password.
    d. Select **Use Encryption Provider**.
    e. For **Provider**, select **nDSOP**.
    f. Select **OK**.



4. Notice the credential created.

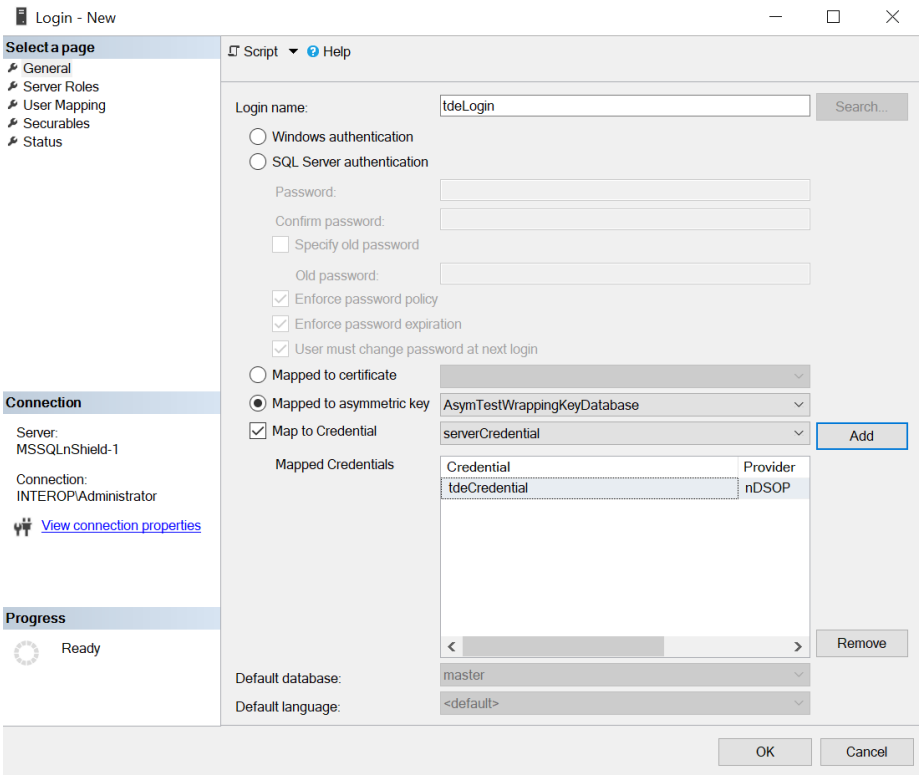


3.2.2. Create a TDE login

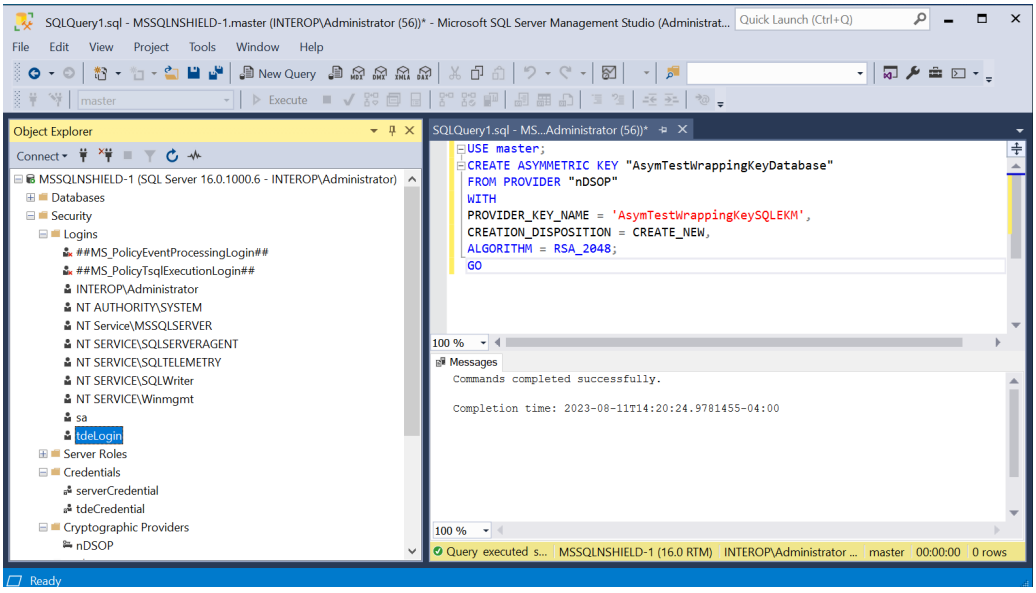
To create a TDE login:

1. In SQL Server Management Studio, navigate to **Security > Logins**.
2. Right-click **Logins**, then select **New Login**.
3. Enter the **Login name**.
4. Select **Mapped to asymmetric key**. Then select the asymmetric key created earlier.
5. Select **Map to Credential**. Then select the TDE credential created earlier. Then select **Add**.

6. Select **OK**.



7. Notice the login created.

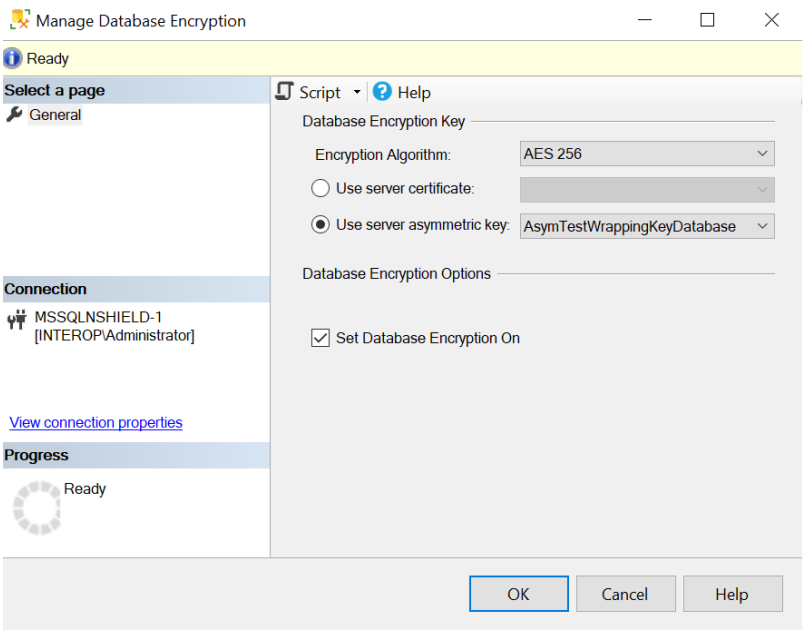


3.3. Create the TDEDEK and switch on encryption

To create the TDEDEK and switch on encryption:

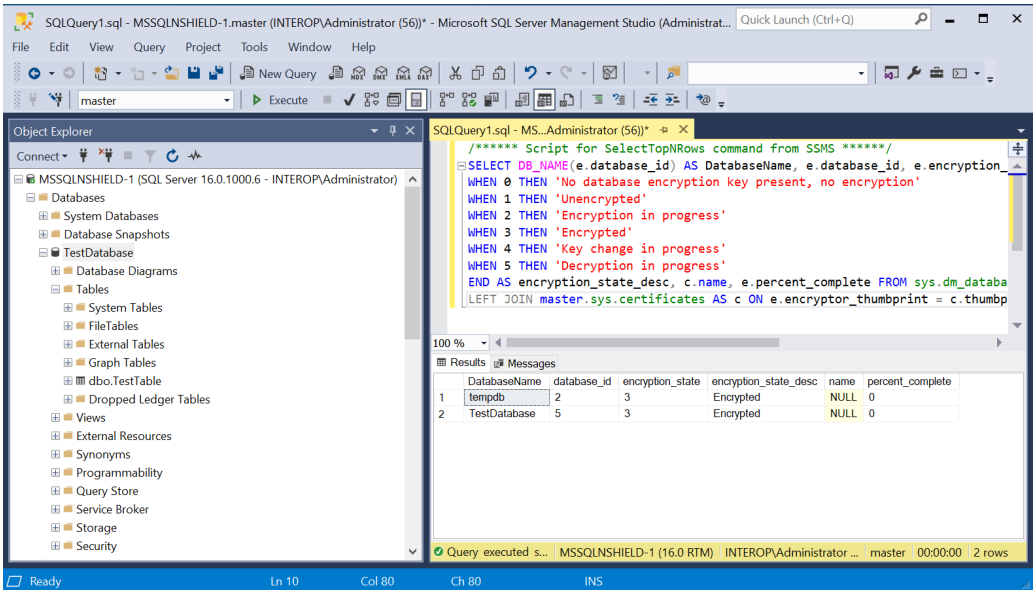
1. In SQL Server Management Studio, navigate to **Databases > TestDatabase**.

2. Right-click TestDatabase, then select **Tasks > Manage Database Encryption**.
3. Set **Encryption Algorithm** to **AES 256** or your choice.
4. Select **Use server asymmetric key**. Then select the asymmetric key created earlier.
5. Select **Set Database Encryption On**. Then select **OK**. Restart the Microsoft SQL Server Management Studio and repeat these steps if it fails.



6. Run the following query to verify the encryption state:

```
/****** Script for SelectTopNRows command from SSMS  ******/
SELECT DB_NAME(e.database_id) AS DatabaseName, e.database_id, e.encryption_state, CASE e.encryption_state
WHEN 0 THEN 'No database encryption key present, no encryption'
WHEN 1 THEN 'Unencrypted'
WHEN 2 THEN 'Encryption in progress'
WHEN 3 THEN 'Encrypted'
WHEN 4 THEN 'Key change in progress'
WHEN 5 THEN 'Decryption in progress'
END AS encryption_state_desc, c.name, e.percent_complete FROM sys.dm_database_encryption_keys AS e
LEFT JOIN master.sys.certificates AS c ON e.encryptor_thumbprint = c.thumbprint
```



The following table shows the value returned for encryption state and the meaning.

Encryption state	Meaning
0	Encryption disabled (or no encryption key)
1	Unencrypted or Decrypted
2	Unencrypted or Decrypted
3	Encrypted
4	Key change in progress
5	Decryption in progress
6	Protection change in progress (The certificate or asymmetric key that is encrypting the database encryption key is being changed)

7. Turn off encryption of by clearing **Set Database Encryption On** in the steps above.

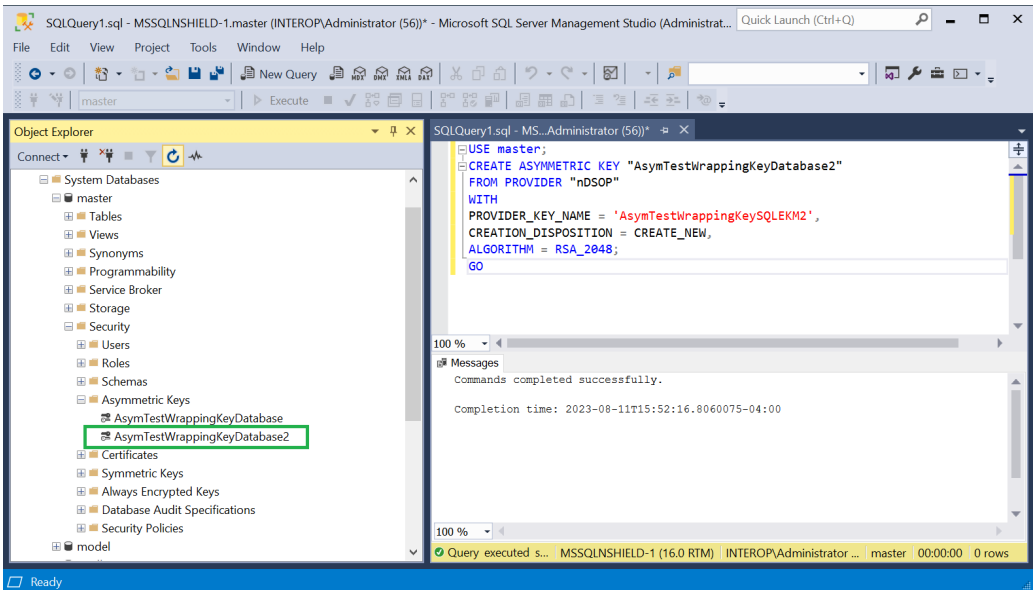
3.4. Key rotation - Replace the TDEKEK

This is the wrapping key called TDE Key Encryption Key, an asymmetric key protected by the SQLEKM provider in the nShield HSM.

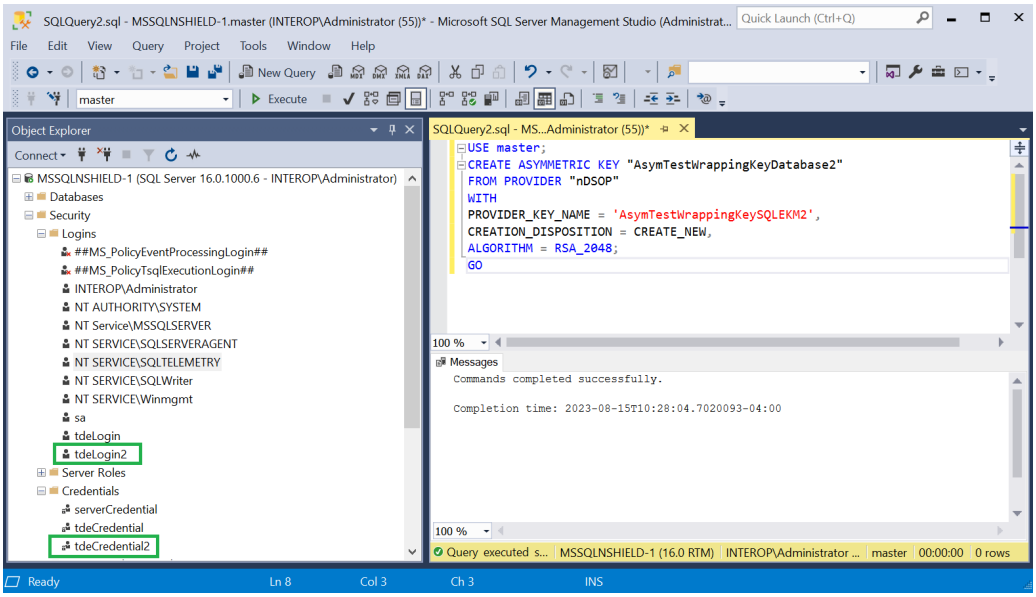
1. Create a new asymmetric TDEKEK. Follow the procedure in Create a TDEKEK. For example:

```
USE master;
CREATE ASYMMETRIC KEY "AsymTestWrappingKeyDatabase2"
FROM PROVIDER "nDSOP"
WITH
PROVIDER_KEY_NAME = 'AsymTestWrappingKeySQLEKM2',
CREATION_DISPOSITION = CREATE_NEW,
ALGORITHM = RSA_2048;
GO
```

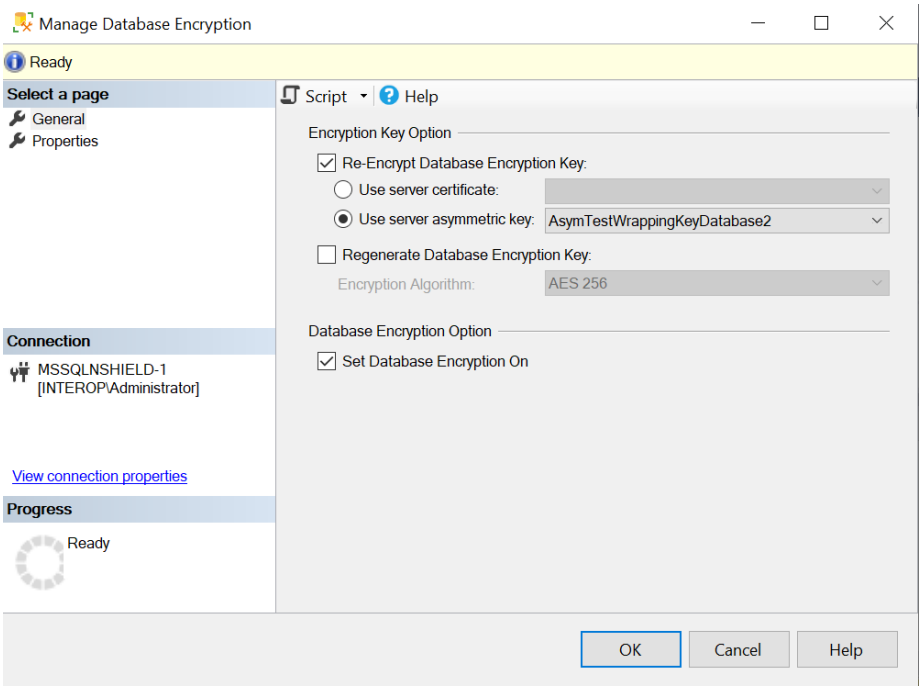
```
> nfkminfo -l

Keys protected by cardsets:
 key_simple_sqlekm-edb3d45a28e5a6b22b033684ce589d9e198272c2-ecaaf2c3e8cb8f0dd3756678b757468a4de120c4
`AsymTestWrappingKeySQLEKM'
 key_simple_sqlekm-edb3d45a28e5a6b22b033684ce589d9e198272c2-f110419800476ccf0bd04b3cd531a59ce3cd2af6
`AsymTestWrappingKeySQLEKM2'
```



2. Create a new TDE credential. Follow the procedure in Create a TDE credential.
3. Create a new TDE login. Follow the procedure in Create a TDE login.



4. In SQL Server Management Studio, navigate to **Databases > TestDatabase**.
5. Right-click TestDatabase, then select **Tasks > Manage Database Encryption**.
6. Select **Re-Encrypt Database Encryption Key** and **Use server asymmetric**.
7. Select the newly created asymmetric key **AsymTestWrappingKeyDatabase2**.
8. Deselect **Regenerate Database Encryption Key**.
9. Select **Set Database Encryption On**.
10. Select **OK**.

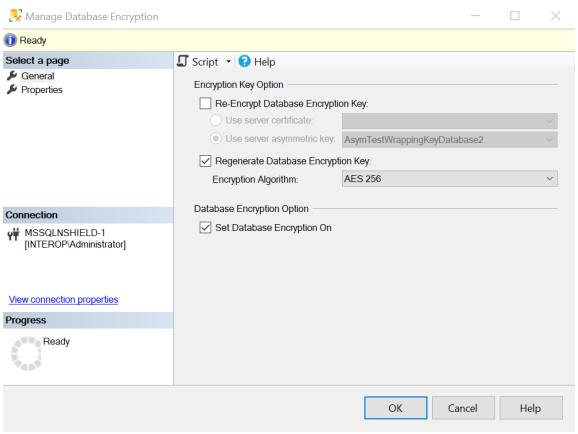


11. Verify the encryption state as shown in Create the TDEDEK and switch on encryption.

3.5. Key rotation - Replace the TDEDEK

This is the key called TDE Database Encryption Key, a symmetric used to perform the actual encryption of the database. It is created by SQL Server and cannot be exported from the database. It is protected within the database by encrypting it with a wrapping key TDEKEK.

1. In SQL Server Management Studio, navigate to **Databases > TestDatabase**.
2. Right-click TestDatabase, then select **Tasks > Manage Database Encryption**.
3. Deselect **Re-Encrypt Database Encryption Key**.
4. Select **Regenerate Database Encryption Key**.
5. Select **AES 256**.
6. Select **Set Database Encryption On**.
7. Select **OK**.



8. Verify the encryption state as shown in Create the TDEDEK and switch on encryption.

Chapter 4. Perform backup and recovery

A rigorous backup regimen is recommended to provide a means to recover both the database and associated keys used for encryption. Consult your corporate IT and security team for best practice and corporate policy requirements.

1. Back up the Security World
2. Restore the Security World
3. Back up the database
4. Restore the database

4.1. Back up the Security World

The Security World data is inherently encrypted and does not require any further encryption operation to protect it. It can only be used by someone who has access to a quorum of the correct ACS cards, or the OCS card, Softcard, their passphrases, an nShield HSM and nShield Security World Software. Therefore, backup simply consists of making a copy of the Security World files and saving the copy in a safe location, as necessary to restore the keys used by the database.

1. Back up `C\:ProgramData\nCipher\Key Management Data`.
2. Securely store and keep a record of ACS and OCS cards associated with each Security World, preferable using the serial number on the cards.
3. The Softcard, used instead of OCS, resides in the `Key Management Data` folder. It is backed up at `C\:ProgramData\nCipher\Key Management Data`.
4. Keep a record of which database and which Security World backups correspond to each other.

4.2. Restore the Security World

Restoring a Security World simply means restoring a backup copy of the Security World folder `C\:ProgramData\nCipher\Key Management Data`.

The ACS is required if the Security World being restored is not already loaded onto the HSM. See the *Installation Guide* and the *User Guide* for the HSM. A short version is available at How to locally set up a new or replacement nShield Connect.

Access to the Entrust nShield Support Portal is available to customers under maintenance. To request an account, contact

> nshield.support@entrust.com.

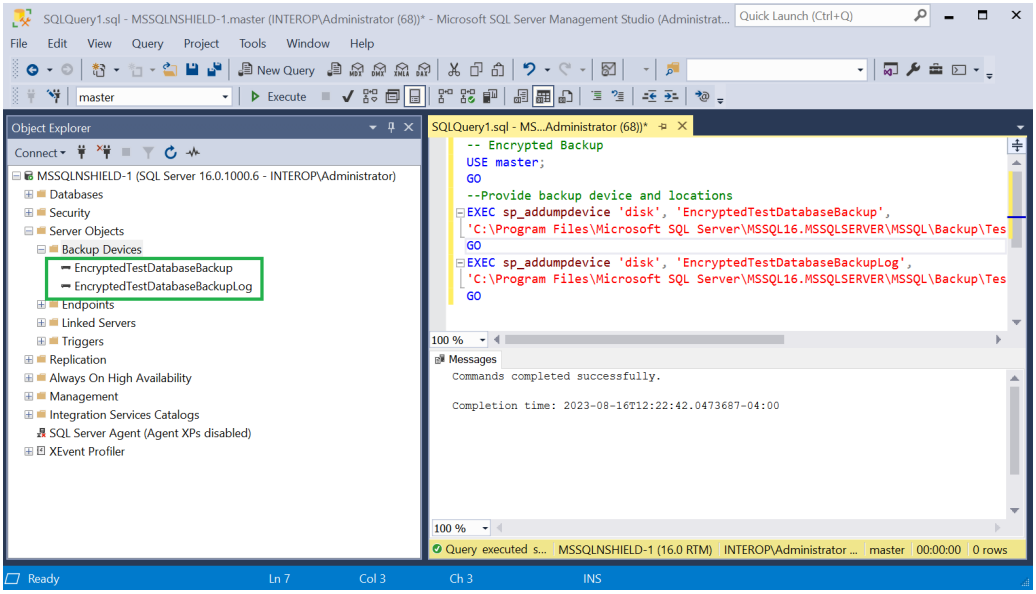
4.3. Back up the database

To back up the database:

1. Create the backup devices by running the following query:

```
-- Encrypted Backup
USE master;
GO

--Provide backup device and locations
EXEC sp_addumpdevice 'disk', 'EncryptedTestDatabaseBackup',
'C:\Program Files\Microsoft SQL Server\MSSQL16.MSSQLSERVER\MSSQL\Backup\TestDatabaseEncrypted.bak';
GO

EXEC sp_addumpdevice 'disk', 'EncryptedTestDatabaseBackupLog',
'C:\Program Files\Microsoft SQL Server\MSSQL16.MSSQLSERVER\MSSQL\Backup\TestDatabaseEncryptedLog.bak';
GO
```

   Notice the devices created.



2. Create the backup by running the following query:

```
-- Encrypted Backup
USE master;
GO

ALTER DATABASE TestDatabase
SET RECOVERY FULL;
GO

-- Back up the encrypted database
BACKUP DATABASE TestDatabase TO EncryptedTestDatabaseBackup;
GO
```

```
-- Back up the encrypted log
BACKUP LOG TestDatabase TO EncryptedTestDatabaseBackupLog;
GO
```

Notice the backup files created.

```
C:\Program Files\Microsoft SQL Server\MSSQL16.MSSQLSERVER\MSSQL\Backup>dir
 Volume in drive C has no label.
 Volume Serial Number is CC11-1791

 Directory of C:\Program Files\Microsoft SQL Server\MSSQL16.MSSQLSERVER\MSSQL\Backup

08/16/2023  12:26 PM    <DIR>          .
08/02/2023  11:27 AM    <DIR>          ..
08/16/2023  12:26 PM         4,743,680 TestDatabaseEncrypted.bak
08/16/2023  12:26 PM            86,528 TestDatabaseEncryptedLog.bak
               2 File(s)      4,830,208 bytes
               2 Dir(s)   6,326,734,848 bytes free
```

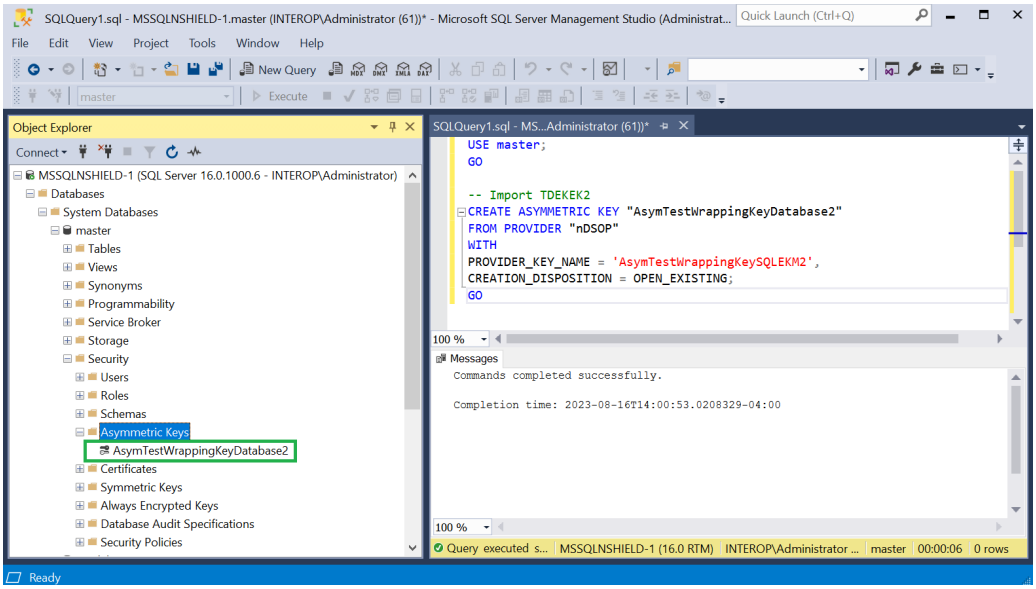
If the database is encrypted, the backup will also be encrypted. If the database is not encrypted, then the backup will not be encrypted. If you want to create an encrypted backup from a non-encrypted database, you will have to create the non-encrypted backup file, and then encrypt the file using an independent encryption tool.

4.4. Restore the database

Restore a TDE encrypted database in a similar manner as an un-encrypted database. But for TDE encrypted database the Security World needs to be restored before restoring the encrypted database. The OCS, if used, needs to be inserted in the HSM before restoring the encrypted database. Otherwise, the restored database will appear as **(Restore Pending)**.

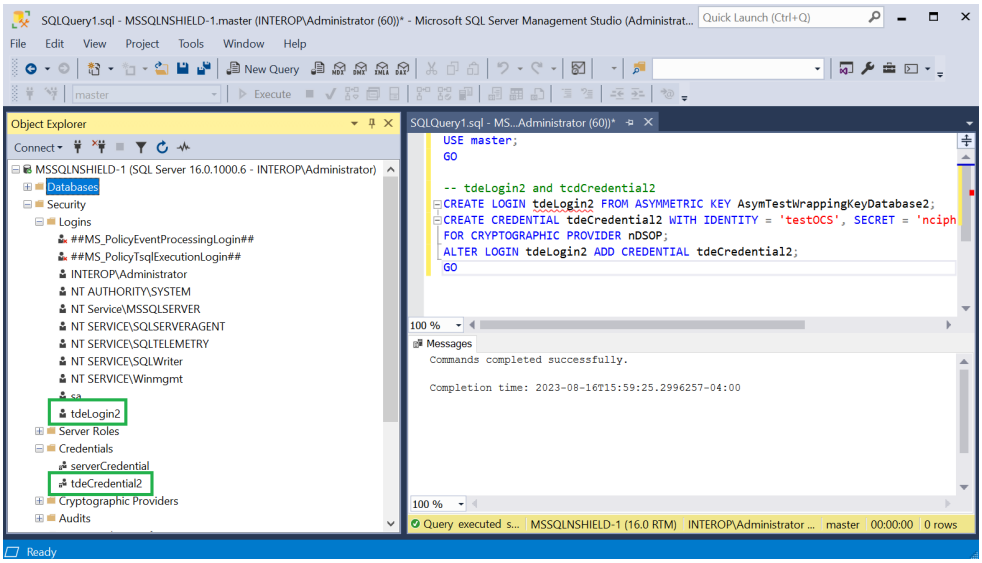
1. Install the Security World software and the nShield nDSOP if rebuilding the server. Do not create a Security World.
2. Restore the Security World.
3. Insert the OCS in the HSM front panel slot, or the TVD if using OCS protection.
4. Enable EKM and register the SQLEKM provider if rebuilding the server.
5. Create the SQL Server credential if rebuilding the server. The OCS and Softcard are in the restored Security World.
6. Verify the SQLEKM provider configuration if rebuilding the server.
7. Import the database wrapping key (TDEKEK) into the master database by running the following query. This is the TDEKEK last used to encrypt the database. This key should already exist in the restored Security World.

```
USE master;
GO

-- Import TDEKEK2
CREATE ASYMMETRIC KEY "AsymTestWrappingKeyDatabase2"
FROM PROVIDER "nDSOP"
WITH
PROVIDER_KEY_NAME = 'AsymTestWrappingKeySQLEKM2',
CREATION_DISPOSITION = OPEN_EXISTING;
GO
```



8. Recreate the TDE login and credential by running the following query. These are the TDE login and credential last used to encrypt the database. Notice the name of the OCS (**nDSOPocs**), and Softcard (**nDSOPsoftcard**) created earlier.
    - OCS:

```
USE master;
GO

-- tdeLogin2 and tcdCredential2
CREATE LOGIN tdeLogin2 FROM ASYMMETRIC KEY AsymTestWrappingKeyDatabase2;
CREATE CREDENTIAL tdeCredential2 WITH IDENTITY = 'testOCS', SECRET = 'ncipher'
FOR CRYPTOGRAPHIC PROVIDER nDSOP;
ALTER LOGIN tdeLogin2 ADD CREDENTIAL tdeCredential2;
GO
```

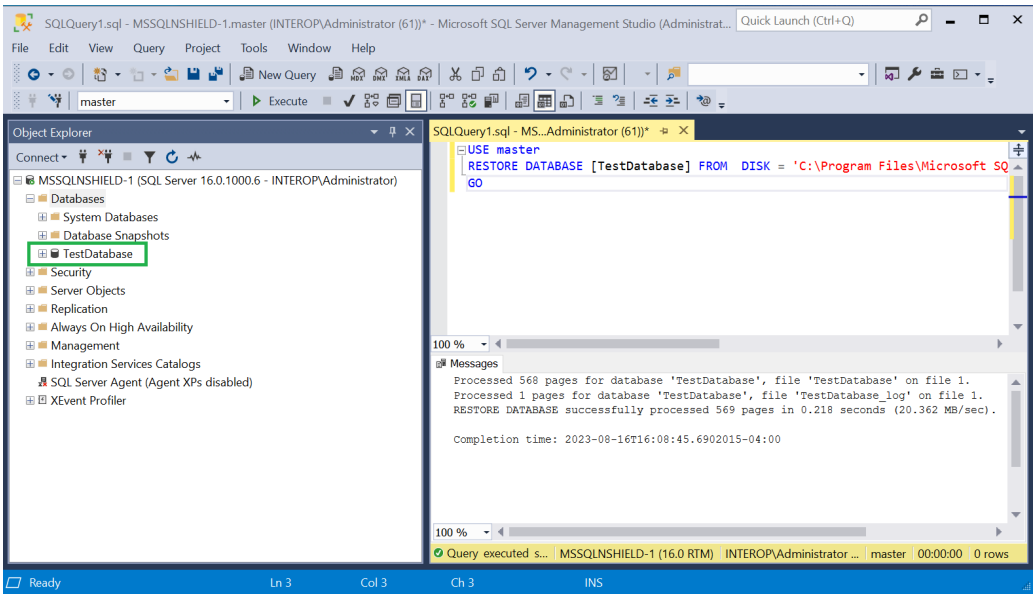


- Softcard:

```
USE master;
GO

-- tdeLogin2 and tdeCredential2
CREATE LOGIN tdeLogin2 FROM ASYMMETRIC KEY AsymTestWrappingKeyDatabase2;
CREATE CREDENTIAL tdeCredential2 WITH IDENTITY = 'testSC', SECRET = 'ncipher'
FOR CRYPTOGRAPHIC PROVIDER nDSOP;
ALTER LOGIN tdeLogin2 ADD CREDENTIAL tdeCredential2;
GO
```

9. Restore the database by running the following query:

```
USE master
RESTORE DATABASE [TestDatabase] FROM  DISK = 'C:\Program Files\Microsoft SQL
Server\MSSQL16.MSSQLSERVER\MSSQL\Backup\TestDatabaseEncrypted.bak'
GO
```



10. Return to multiple user mode by running the following script:

```
USE master;
ALTER DATABASE TestDatabase SET
MULTI_USER;
GO
```

Chapter 5. Column level encryption

Table Column data can be protected by an Entrust nShield HSM protected key. These nDSOP EKM keys can encrypt/decrypt data in a column.

1. Create a new key
2. Import an existing key
3. Encrypt a column with a symmetric key
4. Encrypt a column with an asymmetric key
5. Encrypt a column with the imported asymmetric key

5.1. Create a new key

Create a new key within the SQL Server database to encrypt a column. This key will be protected by the Entrust nShield HSM.

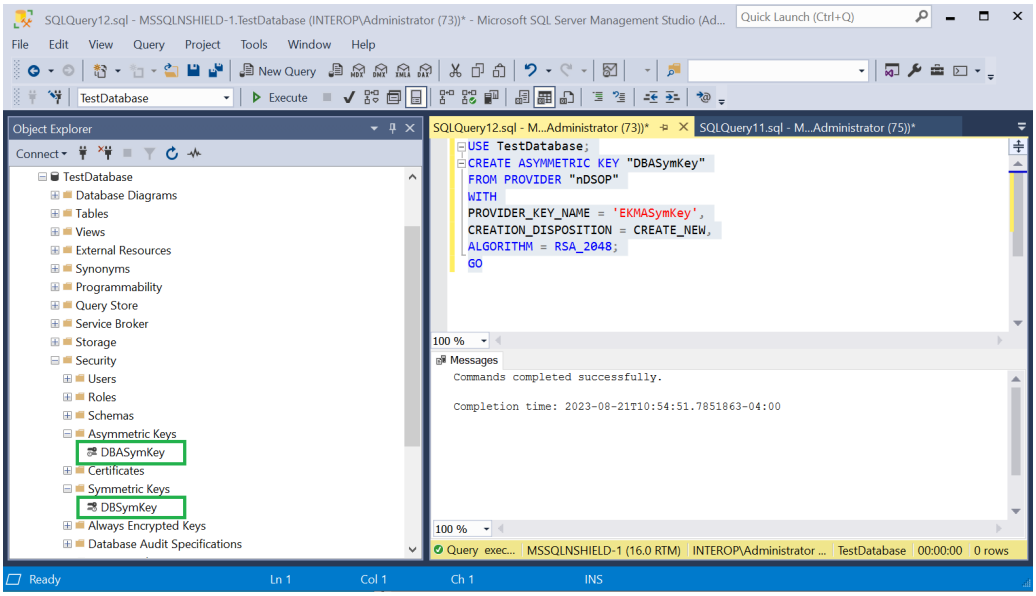
1. Insert the OCS in the HSM slot or TVD. If using Softcard protection, no action is needed.
2. To create an symmetric key, run the following query:

```
USE TestDatabase;
CREATE SYMMETRIC KEY "DBSymKey"
FROM PROVIDER "nDSOP"
WITH
PROVIDER_KEY_NAME = 'EKMSymKey', IDENTITY_VALUE = '$DBSymKey',
CREATION_DISPOSITION = CREATE_NEW,
ALGORITHM = AES_256;
GO
```

3. To create a asymmetric key, run the following query:

```
USE TestDatabase;
CREATE ASYMMETRIC KEY "DBASymKey"
FROM PROVIDER "nDSOP"
WITH
PROVIDER_KEY_NAME = 'EKMASymKey',
CREATION_DISPOSITION = CREATE_NEW,
ALGORITHM = RSA_2048;
GO
```

4. Verify the keys created above.




```
> nfkminfo -l

Keys protected by cardsets:
 key_simple_sqlekm-edb3d45a28e5a6b22b033684ce589d9e198272c2-94fa54413d4f9064af7bb47f553d18109c6c9585
`EKMASymKey'
 key_simple_sqlekm-edb3d45a28e5a6b22b033684ce589d9e198272c2-ecaaf2c3e8cb8f0dd3756678b757468a4de120c4
`AsymTestWrappingKeySQLEKM'
 key_simple_sqlekm-edb3d45a28e5a6b22b033684ce589d9e198272c2-ecc0d19430a8052bf3a55617a0b13522a917f039
`EKMSymKey'
 key_simple_sqlekm-edb3d45a28e5a6b22b033684ce589d9e198272c2-f110419800476ccf0bd04b3cd531a59ce3cd2af6
`AsymTestWrappingKeySQLEKM2'
```

5.2. Import an existing key

The Entrust nShield HSM utility **generatekey** will be used to create a asymmetric key. Then this key will be imported in the SQL Server.

1. Run the utility **generatekey** interactive as show below

The **ident: Key identifier? []** must begin with **sqlekm-**.

```
> generatekey simple
protect: Protected by? (token, softcard, module) [token] >
slot: Slot to read cards from? (0-5) [0] >
recovery: Key recovery? (yes/no) [yes] >
type: Key type? (AES, DES2, DES3, DH, DHEx, DSA, EC, ECDH, ECDSA, HMACSHA1,
                 HMACSHA256, HMACSHA384, HMACSHA512, Rijndael, RSA) [RSA]
>
size: Key size? (bits, minimum 1024) [2048] >
OPTIONAL: pubexp: Public exponent for RSA key (hex)? []
>
ident: Key identifier? [] > sqlekm-EKMExistingASymKey
plainname: Key name? [] > EKMExistingASymKey
nvram: Blob in NVRAM (needs ACS)? (yes/no) [no] >
key generation parameters:
 operation    Operation to perform              generate
```

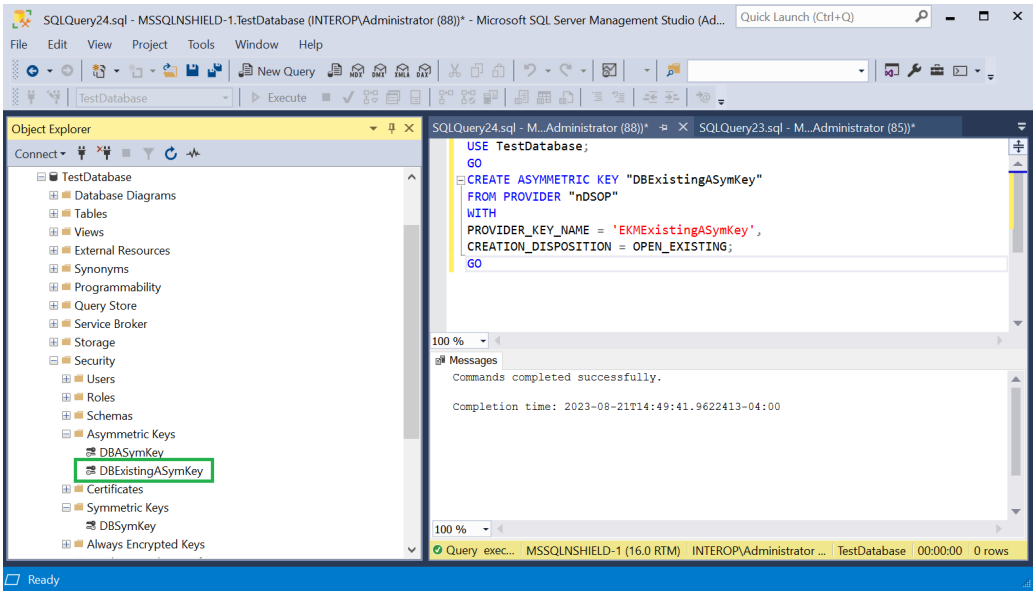
```
 application  Application                        simple
 protect      Protected by                      token
 slot         Slot to read cards from           0
 recovery     Key recovery                      yes
 verify       Verify security of key            yes
 type         Key type                          RSA
 size         Key size                          2048
 pubexp       Public exponent for RSA key (hex)
 ident        Key identifier                    sqlekm-EKMExistingASymKey
 plainname    Key name                          EKMExistingASymKey
 nvram        Blob in NVRAM (needs ACS)         no

Loading cardset(s):
 Module 1 slot 0: `testOCS' #2
 Module 1 slot 2: Admin Card #15
 Module 1 slot 3: empty
 Module 1 slot 4: empty
 Module 1 slot 5: empty
 Module 1 slot 0:- passphrase supplied - reading card
Card reading complete.

Key successfully generated.
Path to key: C:\ProgramData\nCipher\Key Management Data\local\key_simple_sqlekm-ekmexistingasymkey
```

2. Notice the newly created key.

```
> nfkminfo -l

Keys protected by cardsets:
 key_simple_sqlekm-edb3d45a28e5a6b22b033684ce589d9e198272c2-94fa54413d4f9064af7bb47f553d18109c6c9585
`EKMASymKey'
 key_simple_sqlekm-edb3d45a28e5a6b22b033684ce589d9e198272c2-ecaaf2c3e8cb8f0dd3756678b757468a4de120c4
`AsymTestWrappingKeySQLEKM'
 key_simple_sqlekm-edb3d45a28e5a6b22b033684ce589d9e198272c2-ecc0d19430a8052bf3a55617a0b13522a917f039
`EKMSymKey'
 key_simple_sqlekm-edb3d45a28e5a6b22b033684ce589d9e198272c2-f110419800476ccf0bd04b3cd531a59ce3cd2af6
`AsymTestWrappingKeySQLEKM2'
 key_simple_sqlekm-ekmexistingasymkey `EKMExistingASymKey'
```

3. Import the newly created key by running the following query.

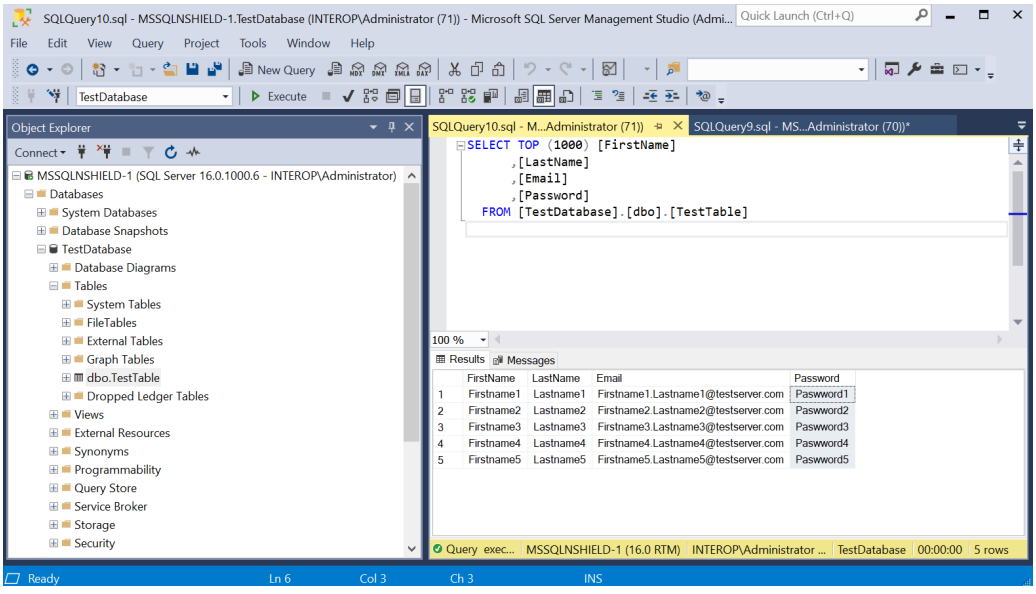
```
USE TestDatabase;
GO
CREATE ASYMMETRIC KEY "DBExistingASymKey"
FROM PROVIDER "nDSOP"
WITH
PROVIDER_KEY_NAME = 'EKMExistingASymKey',
CREATION_DISPOSITION = OPEN_EXISTING;
GO
```



5.3. Encrypt a column with a symmetric key

To encrypt a column with a symmetric key:

1. Consider the table **TestTable** in database **TestDatabase**.

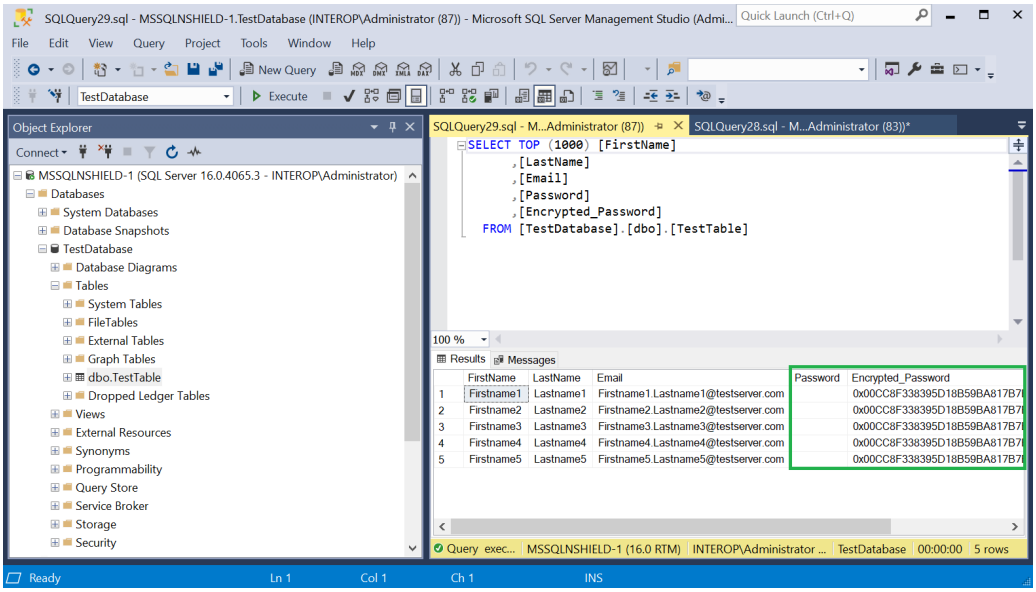


2. Run the following query to create a new **Encrypted_Password** column containing the encrypted passwords with the symmetric key created above, and populate the **Password** column with blanks.

```
USE TestDatabase;
ALTER TABLE TestTable
ADD Encrypted_Password VARBINARY (256);
GO
UPDATE TestTable
```

```
SET Encrypted_Password = ENCRYPTBYKEY(KEY_GUID('DBSymKey'), Password);
UPDATE TestTable
SET Password = '';
GO
```

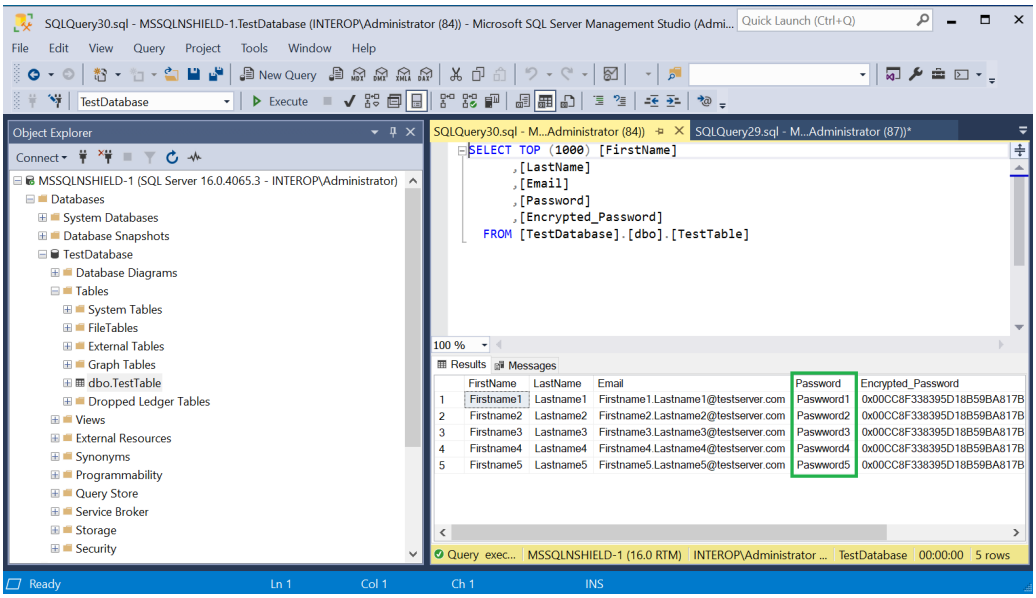
3. Notice the new **Encrypted_Password** column containing the encrypted passwords.



4. Run the following query to decrypted the column above.

```
USE TestDatabase;
UPDATE TestTable
SET Password = DecryptByKey(Encrypted_Password);
GO
```

5. Notice the **Password** column is now populated with the decrypted password.



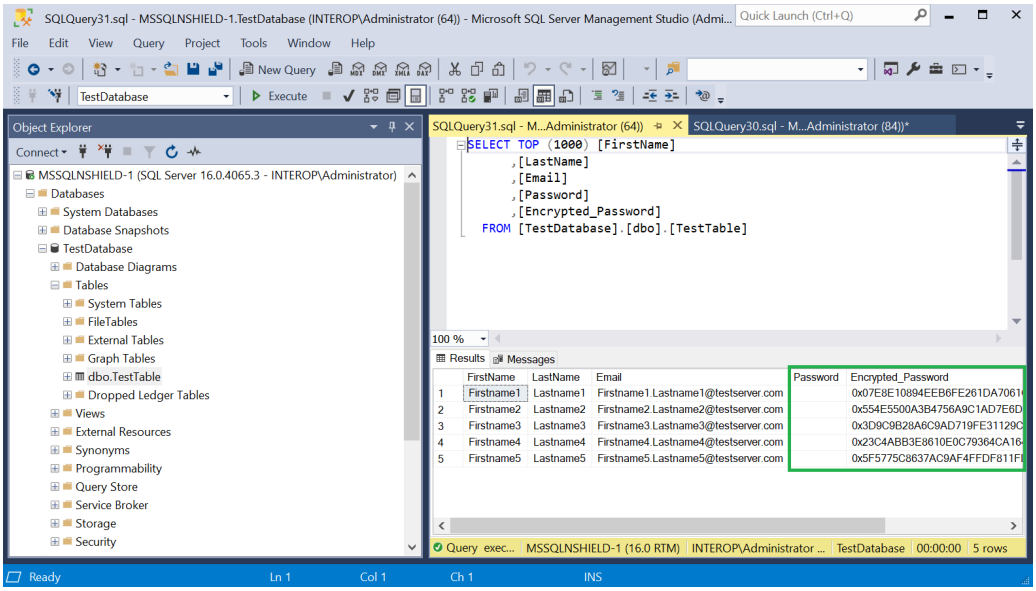
5.4. Encrypt a column with an asymmetric key

To encrypt a column with an asymmetric key:

1. Run the following query to encrypt the passwords with the asymmetric key created above, and populate the **Password** column with blanks.

```
USE TestDatabase;
UPDATE TestTable
SET Encrypted_Password = ENCRYPTBYASYMKEY(ASYMKEY_ID('DBASymKey'), Password);
UPDATE TestTable
SET Password = '';
GO
```

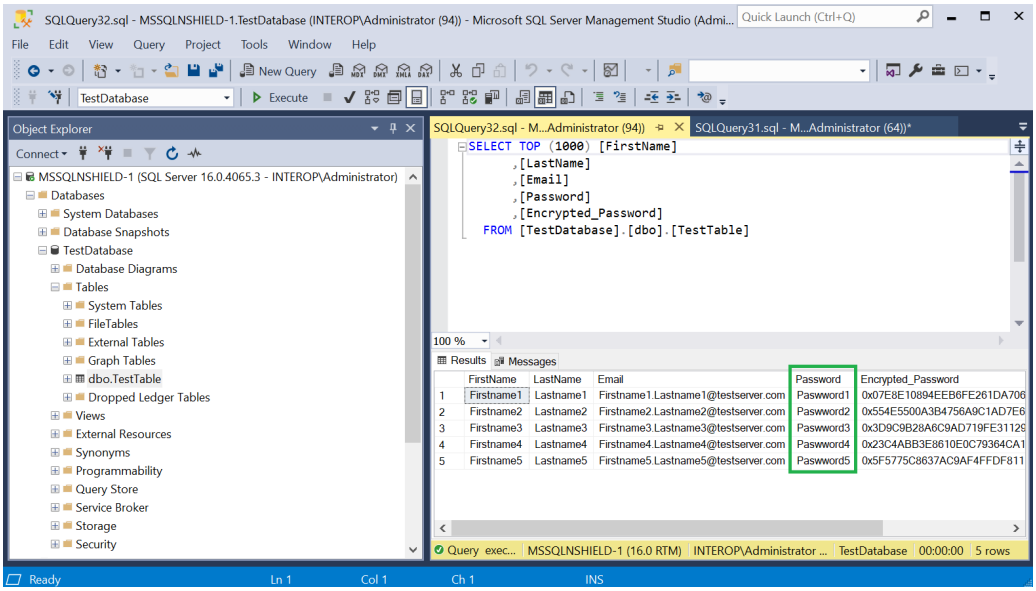
2. Notice the **Encrypted_Password** column has new values corresponding to the asymmetric key.



3. Run the following query to decrypted the column above.

```
USE TestDatabase;
UPDATE TestTable
SET Password = DECRYPTBYASYMKEY(ASYMKEY_ID('DBASymKey'), Encrypted_Password);
GO
```

4. Notice the **Password** column is now populated with the decrypted password.



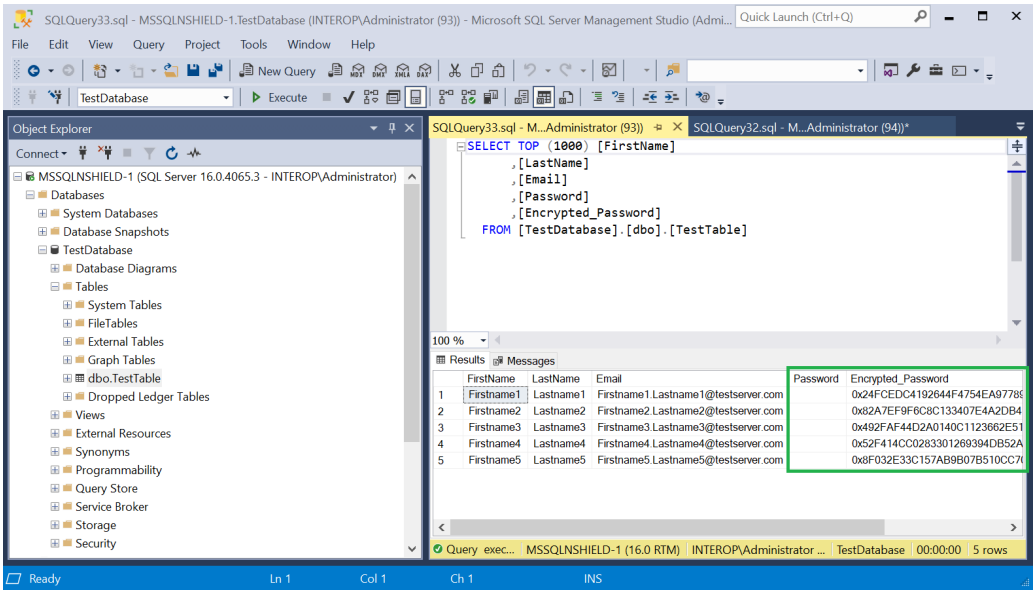
5.5. Encrypt a column with the imported asymmetric key

To encrypt a column with the imported asymmetric key:

1. Run the following query to encrypt the passwords with the imported asymmetric, and populate the **Password** column with blanks.

```
USE TestDatabase;
UPDATE TestTable
SET Encrypted_Password = ENCRYPTBYASYMKEY(ASYMKEY_ID('DBExistingASymKey'), Password);
UPDATE TestTable
SET Password = '';
GO
```

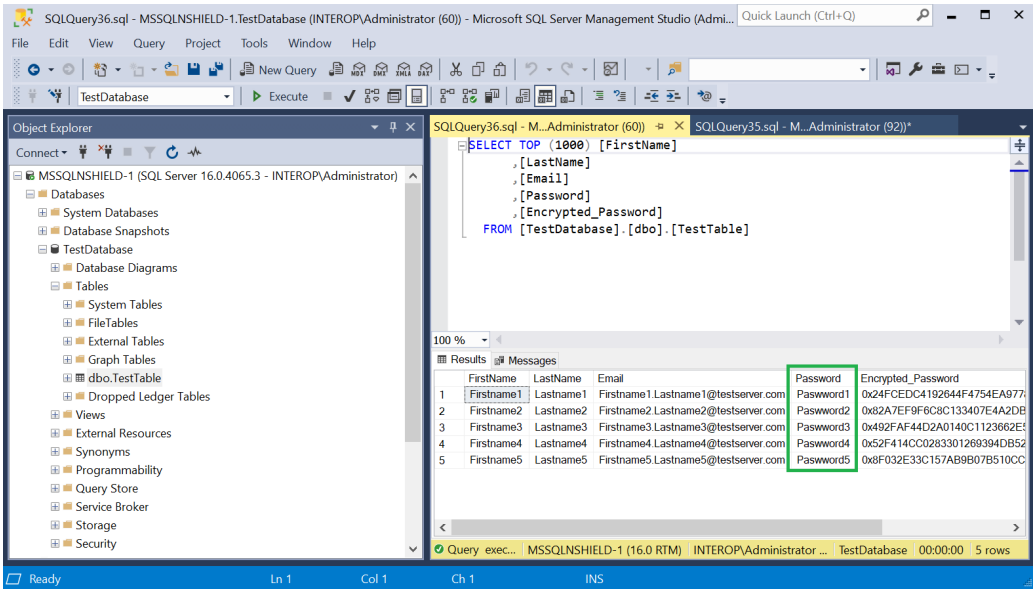
2. Notice the **Encrypted_Password** column has new values corresponding to the imported key.



3. Run the following query to decrypted the column above.

```
USE TestDatabase;
UPDATE TestTable
SET Password = DECRYPTBYASYMKEY(ASYMKEY_ID('DBExistingASymKey'), Encrypted_Password);
GO
```

4. Notice the **Password** column is now populated with the decrypted password.



Chapter 6. Upgrade nDSOP

This section will perform the migration of the Entrust Database Security Option Pack (nDSOP).

From Version	To Version
v1.0	v2.1

6.1. Product configurations

Product	Version
Base OS	Windows Server 2016 Datacenter
SQL Server	Microsoft 2016 Enterprise with Service Pack 2
Microsoft SQL Server Management Studio	v18.8

6.2. Supported nShield hardware and software versions

Product	Security World	Firmware	Netimage
Connect XC	12.60.11 with v2 Compatibility Package	12.50.11 (FIPS Certified)	12.60.10

6.3. Procedure

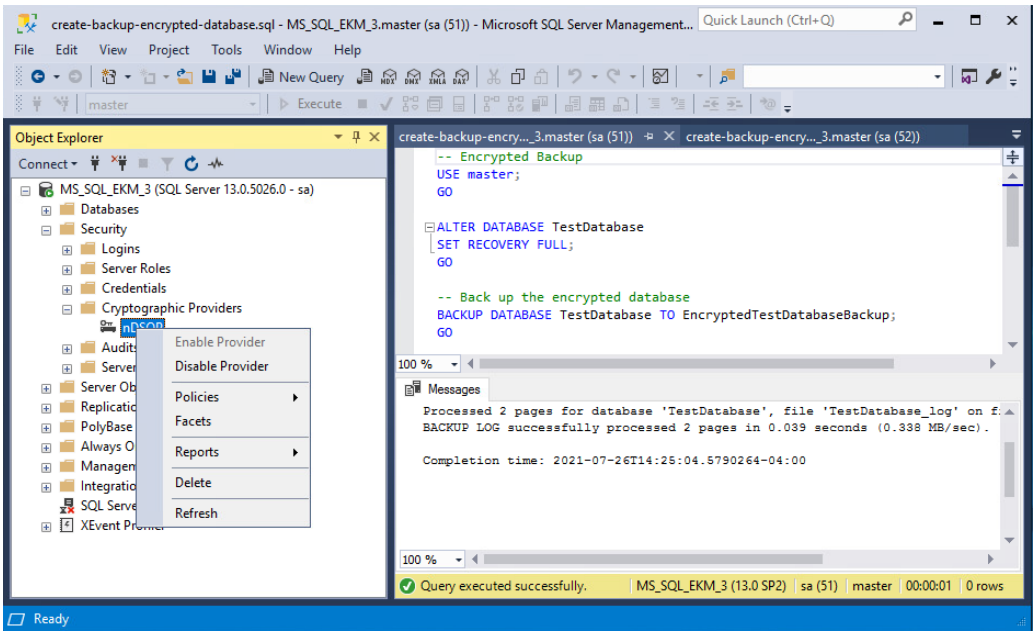
The following procedure will be performed on a Windows Server 2016 with Microsoft SQL Server 2106, and nDSOP v1.0. A database called TestDatabase has been created and encrypted and will be used in this procedure.

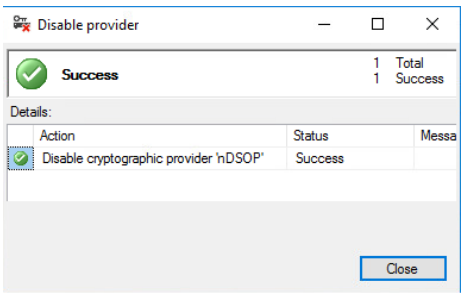
1. Backup the Security World.
2. Backup the database.
3. Run the following query to verify the encryption state.

```
/****** Script for SelectTopNRows command from SSMS  ******/
```

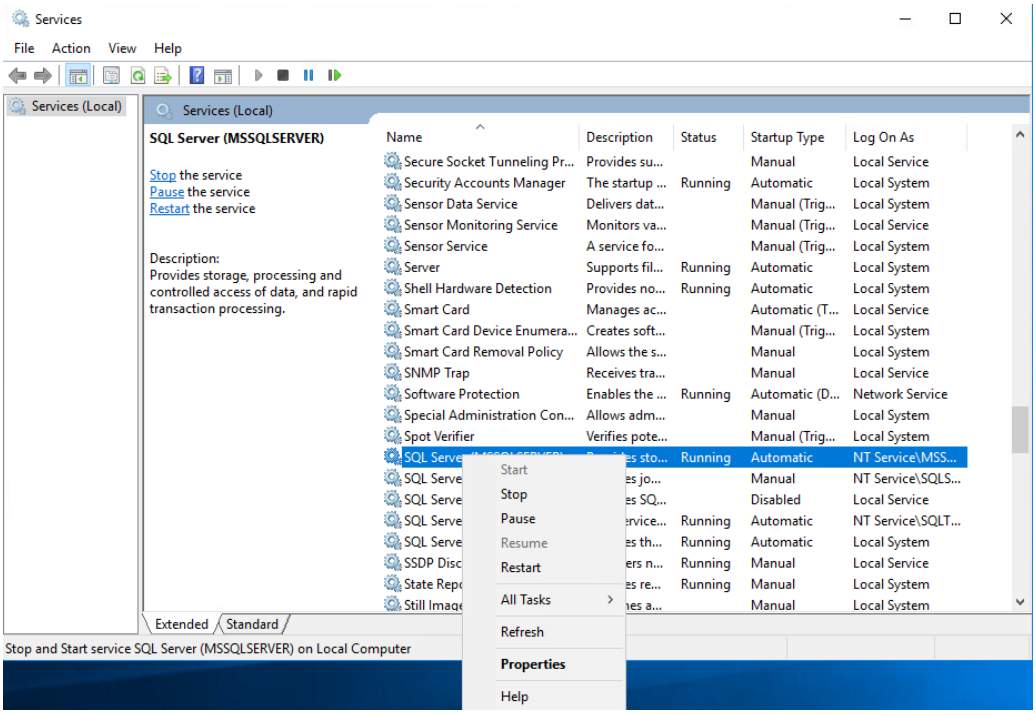
```
SELECT DB_NAME(e.database_id) AS DatabaseName, e.database_id, e.encryption_state, CASE e.encryption_state
WHEN 0 THEN 'No database encryption key present, no encryption'
WHEN 1 THEN 'Unencrypted'
WHEN 2 THEN 'Encryption in progress'
WHEN 3 THEN 'Encrypted'
WHEN 4 THEN 'Key change in progress'
WHEN 5 THEN 'Decryption in progress'
END AS encryption_state_desc, c.name, e.percent_complete FROM sys.dm_database_encryption_keys AS e
LEFT JOIN master.sys.certificates AS c ON e.encryptor_thumbprint = c.thumbprint
```

4. Disable the EKM provider. Select **Security Cryptographic Providers**. Right-click on the provider and select **Disable**.

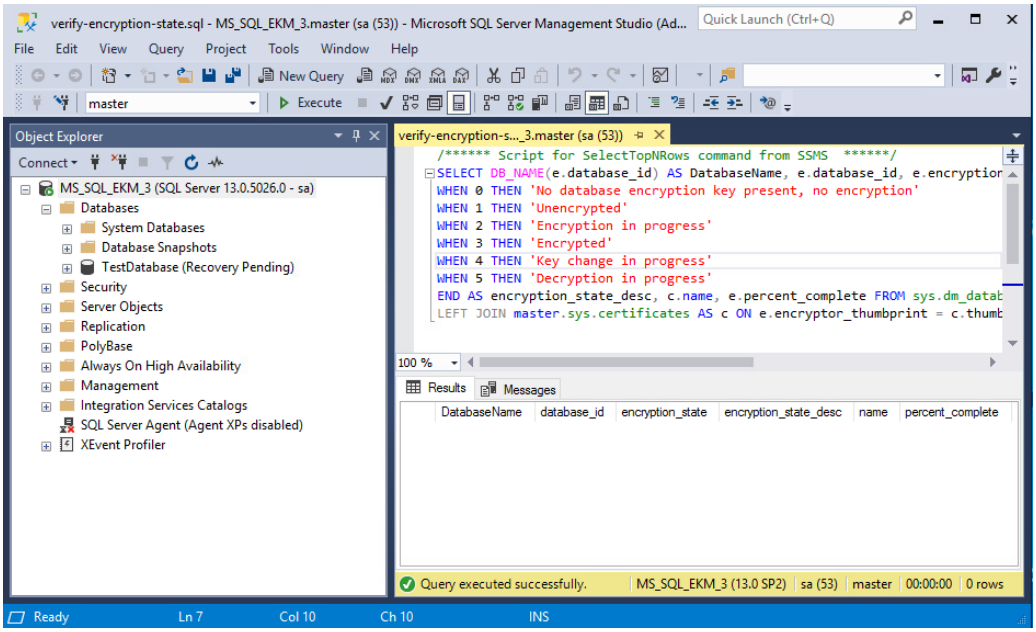




5. Restart the SQL Server from the Windows MSSMS or services.



6. Wait for 60 seconds after the restart. Then check the database status. Notice **Recovery Pending** next to **TestDatabase**.



7. Un-install nDSOP v1.01 EKM provider using the Windows **Control Panel > Programs > Programs and Features**.
8. Install nDSOP v2.1 EKM provider by mounting the `.iso` file and double-clicking `setup`.
9. Insert the OCS in the HSM slot or TVD. No action is needed if you are using Softcard protection.

10. Retarget the keys by running the `sqlekm_retarget_keys` command:

```
C:\Users\Administrator>nfkminfo -k

Key list - 2 keys
 AppName pkcs11               Ident uc79dfaf7c3311d22d240a7257e5e760ede89fbc70-
56ac051fb249f91e641b065dc12fec8a9fea2419
 AppName pkcs11               Ident uc79dfaf7c3311d22d240a7257e5e760ede89fbc70-
c88b06f02bdca29f2a98b9c9352daf9191fc8afd

C:\Users\Administrator>sqlekm_retarget_keys --all
Found 2 keys to retarget
Retargetted: key_pkcs11_uc79dfaf7c3311d22d240a7257e5e760ede89fbc70-c88b06f02bdca29f2a98b9c9352daf9191fc8afd
Retargetted: key_pkcs11_uc79dfaf7c3311d22d240a7257e5e760ede89fbc70-56ac051fb249f91e641b065dc12fec8a9fea2419

C:\Users\Administrator>nfkminfo -k

Key list - 4 keys
 AppName pkcs11               Ident uc79dfaf7c3311d22d240a7257e5e760ede89fbc70-
56ac051fb249f91e641b065dc12fec8a9fea2419
 AppName pkcs11               Ident uc79dfaf7c3311d22d240a7257e5e760ede89fbc70-
c88b06f02bdca29f2a98b9c9352daf9191fc8afd
 AppName simple               Ident sqlekm-79dfaf7c3311d22d240a7257e5e760ede89fbc70-
b1844c5bb4eadbdb1166dcdb64f4c5d59e4e408c
 AppName simple               Ident sqlekm-79dfaf7c3311d22d240a7257e5e760ede89fbc70-
fa9380a3e111df122b0e02dd37c1233da89b8e16
```

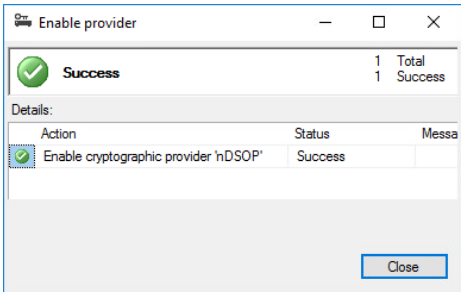
11. Open the `C:\ProgramData\nCipher\Key Management Data\local` folder. Move all `pkcs11` keys to another folder. Leave the `simple` keys in the current folder.

```
C:\ProgramData\nCipher\Key Management Data>mkdir local_pcks11_keys

C:\ProgramData\nCipher\Key Management Data>move local\key_pkcs11* local_pcks11_keys\.
C:\ProgramData\nCipher\Key Management Data\local\key_pkcs11_uc79dfaf7c3311d22d240a7257e5e760ede89fbc70-
56ac051fb249f91e641b065dc12fec8a9fea2419
C:\ProgramData\nCipher\Key Management Data\local\key_pkcs11_uc79dfaf7c3311d22d240a7257e5e760ede89fbc70-
c88b06f02bdca29f2a98b9c9352daf9191fc8afd
        2 file(s) moved.

C:\ProgramData\nCipher\Key Management Data>nfkminfo -k

Key list - 2 keys
 AppName simple               Ident sqlekm-79dfaf7c3311d22d240a7257e5e760ede89fbc70-
b1844c5bb4eadbdb1166dcdb64f4c5d59e4e408c
 AppName simple               Ident sqlekm-79dfaf7c3311d22d240a7257e5e760ede89fbc70-
fa9380a3e111df122b0e02dd37c1233da89b8e16
```

12. Set the new provider by running the following query:

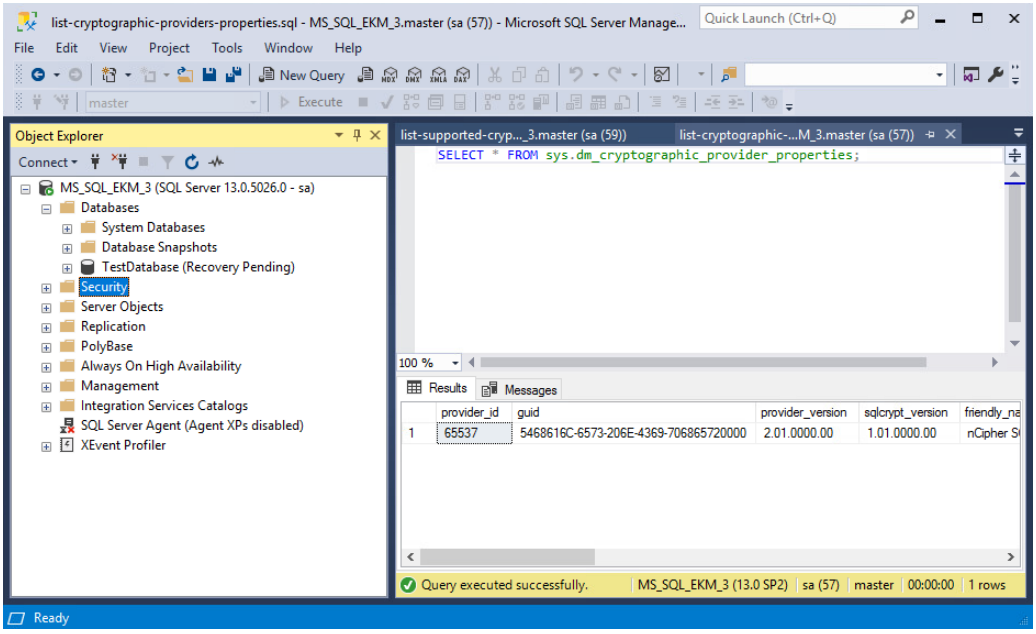
```
--ChangeToNewProvider.sql

ALTER CRYPTOGRAPHIC PROVIDER nDSOP
FROM FILE = 'C:\Program Files\nCipher\nfast\bin\ncsqlekm.dll';
GO
```

13. Enable the EKM provider. Select **Security > Cryptographic Providers**. Right-click the provider and select **Enable**.



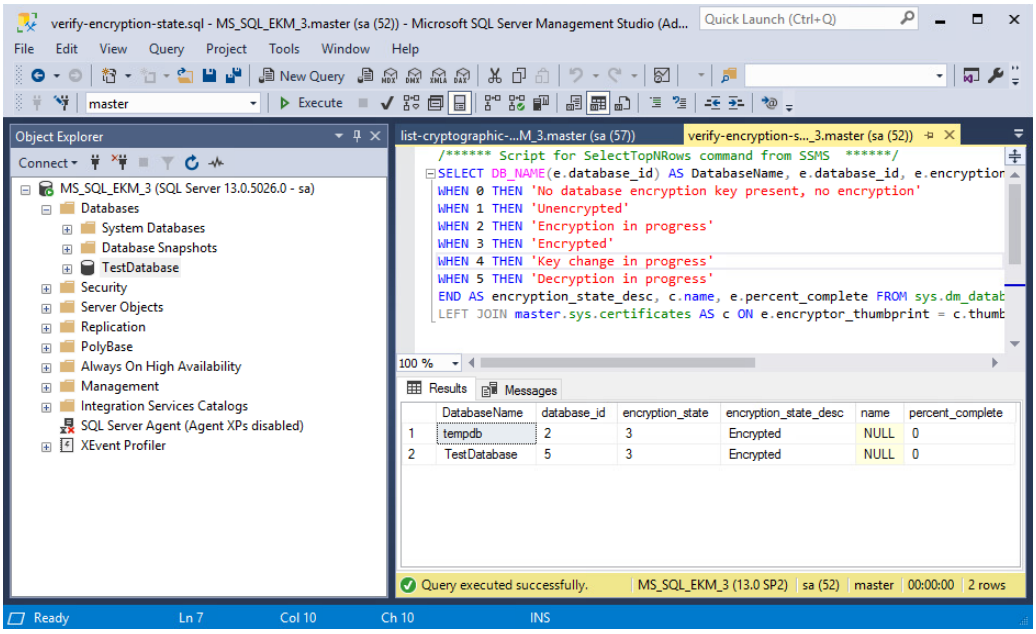
14. Verify the new EKM provider version by running the following query. Notice the **provider_version**.

```
SELECT * FROM sys.dm_cryptographic_provider_properties;
```



15. Restart the SQL Server from the Windows MSSMS or services. Wait for 60 seconds after the restart.
16. Check and refresh database status. Notice the **Recovery Pending** message next to the TestDatabase goes away.
17. Verify the encryption state by running the following query. Notice the **encryption_state_desc** shown as **Encrypted**.

```
/****** Script for SelectTopNRows command from SSMS  ******/
SELECT DB_NAME(e.database_id) AS DatabaseName, e.database_id, e.encryption_state, CASE e.encryption_state
WHEN 0 THEN 'No database encryption key present, no encryption'
WHEN 1 THEN 'Unencrypted'
WHEN 2 THEN 'Encryption in progress'
WHEN 3 THEN 'Encrypted'
WHEN 4 THEN 'Key change in progress'
WHEN 5 THEN 'Decryption in progress'
END AS encryption_state_desc, c.name, e.percent_complete FROM sys.dm_database_encryption_keys AS e
LEFT JOIN master.sys.certificates AS c ON e.encryptor_thumbprint = c.thumbprint
```


verify-encryption-state.sql - MS_SQL_EKM_3.master (sa (52)) - Microsoft SQL Server Management Studio (Ad...
Quick Launch (Ctrl+Q)
File Edit View Query Project Tools Window Help
New Query
master
Execute
Object Explorer
Connect
MS_SQL_EKM_3 (SQL Server 13.0.5026.0 - sa)
Databases
System Databases
Database Snapshots
TestDatabase
Security
Server Objects
Replication
PolyBase
Always On High Availability
Management
Integration Services Catalogs
SQL Server Agent (Agent XPs disabled)
XEvent Profiler
list-cryptographic-...M_3.master (sa (57))
verify-encryption-s..._3.master (sa (52))
/****** Script for SelectTopNRows command from SSMS ******/
SELECT DB_NAME(e.database_id) AS DatabaseName, e.database_id, e.encryptior
WHEN 0 THEN 'No database encryption key present, no encryption'
WHEN 1 THEN 'Unencrypted'
WHEN 2 THEN 'Encryption in progress'
WHEN 3 THEN 'Encrypted'
WHEN 4 THEN 'Key change in progress'
WHEN 5 THEN 'Decryption in progress'
END AS encryption_state_desc, c.name, e.percent_complete FROM sys.dm_datab
LEFT JOIN master.sys.certificates AS c ON e.encryptor_thumbprint = c.thumb
100 %
Results
Messages
DatabaseName database_id encryption_state encryption_state_desc name percent_complete
1 tempdb 2 3 Encrypted NULL 0
2 TestDatabase 5 3 Encrypted NULL 0
Query executed successfully.
MS_SQL_EKM_3 (13.0 SP2) sa (52) master 00:00:00 2 rows
Ready Ln 7 Col 10 Ch 10 INS

Chapter 7. Troubleshoot

7.1. Microsoft SQL Server, Error: 15209 while rotating the TDEKEK

1. Restart the database.
2. Try again to rotate the TDEKEK.