# Table of Contents

Table of Contents ........................................................................................................................................... 3
Table of Figures .................................................................................................................................................. 4
Always Encrypted and Thales nShield HSMs ................................................................................................. 5
  Introduction to Always Encrypted .................................................................................................................. 5
  Always Encrypted Integration using Operator Cards and/or Module protection ........................................... 5
  Requirements ................................................................................................................................................ 6
  Using multiple on-premises client servers ....................................................................................................... 6
  Database Permissions .................................................................................................................................... 6
  Security Worlds and key protection ................................................................................................................ 7
  Application Key Tokens .................................................................................................................................. 7
  Configuring nShield Hardware Security Modules for use with Always Encrypted ........................................ 8
  Install and register the CNG provider ............................................................................................................. 8
Always Encrypted using SSMS .......................................................................................................................... 14
  Creating the Always Encrypted Column Master Key using the nCipher KSP ................................................ 14
  Enable Always Encrypted ............................................................................................................................ 19
  Removing column encryption ....................................................................................................................... 26
Always Encrypted using PowerShell: without Role Separation .................................................................... 29
  Install and Configure SqlServer PowerShell module ................................................................................... 29
  Install the Thales nCipher CNG provider ...................................................................................................... 29
  Creating the Always Encrypted Column Master Key using the nCipher KSP ................................................. 30
  Creating the Column Encryption Key ........................................................................................................... 30
  Encrypting Columns with the Column Encryption Key .................................................................................. 31
  Remove Always Encrypted Column Encryption .......................................................................................... 31
  Query the encrypted columns ....................................................................................................................... 32
Always Encrypted using PowerShell: with Role Separation ......................................................................... 33
  Install and Configure SqlServer PowerShell module and verify KSP ......................................................... 33
  Creating the Always Encrypted Column Master Key using the nCipher KSP ................................................. 34
  Encrypt Columns using SSMS ....................................................................................................................... 38
  Encrypt Columns using PowerShell (ISE) ...................................................................................................... 38
  Remove Always Encrypted Column Encryption .......................................................................................... 39
  Query the encrypted columns ....................................................................................................................... 40
Glossary of PowerShell SqlServer CMDlets .................................................................................................. 41
Feature Details .................................................................................................................................................. 42
Troubleshooting .............................................................................................................................................. 43
Table of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Figure 1</td>
<td>Always Encrypted example using HSM to protect CMK</td>
<td>7</td>
</tr>
<tr>
<td>Figure 2</td>
<td>Install and register nShield provider</td>
<td>8</td>
</tr>
<tr>
<td>Figure 3</td>
<td>CNG install Welcome screen</td>
<td>8</td>
</tr>
<tr>
<td>Figure 4</td>
<td>Select to enable / disable Pool Mode</td>
<td>9</td>
</tr>
<tr>
<td>Figure 5</td>
<td>Set Module States</td>
<td>9</td>
</tr>
<tr>
<td>Figure 6</td>
<td>Optional setting to enable module for remote shares</td>
<td>10</td>
</tr>
<tr>
<td>Figure 7</td>
<td>Present ACS card when prompted</td>
<td>10</td>
</tr>
<tr>
<td>Figure 8</td>
<td>Set Key Protection</td>
<td>11</td>
</tr>
<tr>
<td>Figure 9</td>
<td>Writing the Operator Card Set</td>
<td>12</td>
</tr>
<tr>
<td>Figure 10</td>
<td>Register CNG Providers</td>
<td>12</td>
</tr>
<tr>
<td>Figure 11</td>
<td>CNG provider registry Path</td>
<td>13</td>
</tr>
<tr>
<td>Figure 12</td>
<td>New Column Master Key</td>
<td>14</td>
</tr>
<tr>
<td>Figure 13</td>
<td>Generate new CMK</td>
<td>15</td>
</tr>
<tr>
<td>Figure 14</td>
<td>nCipher KSP - Create new key</td>
<td>15</td>
</tr>
<tr>
<td>Figure 15</td>
<td>Select module or OCS</td>
<td>16</td>
</tr>
<tr>
<td>Figure 16</td>
<td>Select card set by name</td>
<td>16</td>
</tr>
<tr>
<td>Figure 17</td>
<td>Enter passphrase</td>
<td>17</td>
</tr>
<tr>
<td>Figure 18</td>
<td>Card reading complete</td>
<td>17</td>
</tr>
<tr>
<td>Figure 19</td>
<td>New CMK generated</td>
<td>17</td>
</tr>
<tr>
<td>Figure 20</td>
<td>New CMK</td>
<td>18</td>
</tr>
<tr>
<td>Figure 21</td>
<td>Encrypt Columns</td>
<td>19</td>
</tr>
<tr>
<td>Figure 22</td>
<td>Column selection and encryption type</td>
<td>19</td>
</tr>
<tr>
<td>Figure 23</td>
<td>Select CMK to use</td>
<td>20</td>
</tr>
<tr>
<td>Figure 24</td>
<td>Run Settings</td>
<td>20</td>
</tr>
<tr>
<td>Figure 25</td>
<td>Verify Settings</td>
<td>21</td>
</tr>
<tr>
<td>Figure 26</td>
<td>Load CMK</td>
<td>21</td>
</tr>
<tr>
<td>Figure 27</td>
<td>Enter passphrase for OCS protecting the CMK</td>
<td>22</td>
</tr>
<tr>
<td>Figure 28</td>
<td>Confirmation of card reading</td>
<td>22</td>
</tr>
<tr>
<td>Figure 29</td>
<td>Load key</td>
<td>23</td>
</tr>
<tr>
<td>Figure 30</td>
<td>Enter passphrase</td>
<td>23</td>
</tr>
<tr>
<td>Figure 31</td>
<td>Card reading complete</td>
<td>24</td>
</tr>
<tr>
<td>Figure 32</td>
<td>CEK successfully encrypted column</td>
<td>24</td>
</tr>
<tr>
<td>Figure 33</td>
<td>Showing encrypted columns</td>
<td>25</td>
</tr>
<tr>
<td>Figure 34</td>
<td>Select Encrypt Columns</td>
<td>26</td>
</tr>
<tr>
<td>Figure 35</td>
<td>Choose the option - Plaintext</td>
<td>26</td>
</tr>
<tr>
<td>Figure 36</td>
<td>Confirm that database is off-line</td>
<td>27</td>
</tr>
<tr>
<td>Figure 37</td>
<td>Review column decryption state</td>
<td>27</td>
</tr>
<tr>
<td>Figure 38</td>
<td>Successfully removed Always Encrypted column encryption</td>
<td>28</td>
</tr>
<tr>
<td>Figure 39</td>
<td>SqlServer module installed</td>
<td>29</td>
</tr>
<tr>
<td>Figure 40</td>
<td>Parameterization for Always Encrypted</td>
<td>32</td>
</tr>
<tr>
<td>Figure 41</td>
<td>Adjust Connection Parameters</td>
<td>32</td>
</tr>
<tr>
<td>Figure 42</td>
<td>Verify Thales KSP is installed</td>
<td>33</td>
</tr>
<tr>
<td>Figure 43</td>
<td>CMK imported into Database</td>
<td>36</td>
</tr>
<tr>
<td>Figure 44</td>
<td>CEK creation</td>
<td>37</td>
</tr>
<tr>
<td>Figure 45</td>
<td>Permissions Status</td>
<td>38</td>
</tr>
<tr>
<td>Figure 46</td>
<td>Parameterization for Always Encrypted</td>
<td>40</td>
</tr>
<tr>
<td>Figure 47</td>
<td>Adjust Connection Parameters</td>
<td>40</td>
</tr>
</tbody>
</table>
Always Encrypted and Thales nShield HSMs

Introduction to Always Encrypted

Always Encrypted is a feature in Windows SQL Server 2016 designed to protect sensitive data both at rest and in flight between an on-premises 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 on-premises 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 / SQL server Databases. Always Encrypted was designed to be used in conjunction with Transparent Data Encryption however; TDE is NOT a requisite for implementing Always Encrypted.

Configuring Always Encrypted involves creating and provisioning cryptographic keys, specifically:

- A Column Master Key – The CMK, is an asymmetric RSA encryption key of size 2048 bits
- One or more Column Encryption Key(s) - A CEK, is a symmetric AES key of size 256 bits.

The CEK is responsible for encrypting the database column data while the CMK is protected by the HSM and is responsible for wrapping (encrypting) the CEK.

The table below shows current support for the different data operations.

<table>
<thead>
<tr>
<th>Task</th>
<th>SSMS</th>
<th>PowerShell</th>
<th>T-SQL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Provisioning column master keys, column encryption keys and encrypted column encryption keys with their corresponding column master keys</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Creating key metadata in the database</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Creating new tables with encrypted columns</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Encrypting existing data in selected database columns</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
</tr>
</tbody>
</table>

The Column Master Key is generated using the Thales nCipher CNG provider via the HSM and the key(s) stored in an encrypted state on the on-premises client application server in the %NFAST_KMDATA%\local folder.

Note: It is recommended that the server configured with Always Encrypted be located on a different server than that on which the database resides.

Always Encrypted supports two named types of encryption, Deterministic and Randomized. Selecting deterministic encryption means that the same encrypted value will be produced from the same plaintext value each time encryption occurs, this allows for point lookups, equality joins, grouping and indexing on encrypted columns. However, this has implications on the security of the data as it potentially allows an attacker to ‘guess’ the plaintext from the recurring cipher text through emerging patterns within the encrypted columns. Deterministic encryption should not really be used where a small set of values are presented, e.g. True / False, Yes / No etc. Randomized encryption is more secure, as it produces different cipher text values from the same plaintext every time the data is encrypted, eliminating the predictable aspects associated with deterministic encryption, however, this also removes the ability to perform any search operations on the encrypted data in situ.

Always Encrypted Integration using Operator Cards and/or Module protection.

In order to cover both methods of protection currently available using CNG and Thales HSMs, the guide performs Always Encrypted configuration using both Operator Card Sets and Module protection for the Column Master Key (CMK). Operator Card Set protection (OCS are physical tokens in the form of a quorum of smart cards) is covered in the “Always Encrypted using SSMS” whereas Module* protection is covered in the section “Always Encrypted using PowerShell”.

* Module protection utilises an AES 256 bit symmetric key with 128 bit security secured by the Security World Module key which is stored in the HSM hardware FIPS 140-2 level 3 boundary.
Requirements

This integration guide provides a step by step account detailing the configuration of the SQL server 2016 Always Encrypted feature. The guide covers the configuration of Always Encrypted by reference to both the Microsoft Always Encrypted wizard GUI accessed via the SSMS and Microsoft PowerShell ISE for provisioning the required cryptographic key(s). It is recommended that you read through the entirety of the guide before proceeding to implement Always Encrypted.

N.B. Always backup you database before performing any activity that may compromise database availability.

The integration was performed and tested using the following configuration:

- Microsoft Windows 2016 Hosting SQL Server 2016 Database
- Microsoft Windows 2016 for “On-Premise” client sever
- .NET Framework 4.6.1
- SQL Server Management Studio 17.x (SSMS)
- PowerShellGet 1.1.3.1
- PowerShell version 5.1 (Desktop)
- Thales nShield HSM with Security World software 12.40.xx
- Thales nShield Hardware Security module (nShield Solo +; nShield Connect +; nShield Solo XC)

You must have at least .NET Framework 4.6.1 on the on-premises client server before installing SQL Server Management Studio (SSMS). The download for .NET framework 4.6.1 can be obtained via the link below:

.NET Framework 4.6.1 download URL:


The Always Encrypted integration process requires administrator level access to both the “on-premises” client server and target database server for initial configuration of the Column Master Key and Column Encryption Key, thereafter Column Encryption is performed entirely via the Client Application server.

Using multiple on-premises client servers

In order for multiple on-premises client application servers to share and decrypt database columns encrypted with HSM assisted Always Encrypted, there is a requirement that:

- each client server wanting access to the contents of data encrypted with a given Column Encryption key protected by a specific Column Master Key that the server must have access to an HSM in the same Security World
- and have a copy of the Column Master Key Application key token stored on its local drive in “C:\Program Data\nCipher\Key Management Data\local” (%NFAST_LOCAL%).

By default "C:\Program Data" is a hidden folder. To view this folder open an explorer window go to the “View” tab and tick the check box named “Hidden items”.

For more information about:

- Configuring a Thales nShield HSM, see the Installation Guide for your HSM available on the DVD supplied with the HSM.
- Security World Configuration, see the appropriate User Guide for your HSM.

Database Permissions

There are four permissions required for Always Encrypted:

<table>
<thead>
<tr>
<th>Operation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALTER ANY COLUMN MASTER KEY</td>
<td>Required to create and delete a column master key</td>
</tr>
<tr>
<td>ALTER ANY COLUMN ENCRYPTION KEY</td>
<td>Required to create and delete a column encryption key</td>
</tr>
<tr>
<td>VIEW ANY COLUMN MASTER KEY</td>
<td>Required to access and read the metadata of the column master keys to manage keys or query encrypted columns</td>
</tr>
<tr>
<td>VIEW ANY COLUMN ENCRYPTION KEY</td>
<td>Required to access and read the metadata of the column encryption key to manage keys or query encrypted columns</td>
</tr>
</tbody>
</table>
The Thales Security World Software must be installed onto the “on-premises” client application server(s) utilizing the SQL Server 2016 Always Encrypted feature.

**Note:** If you are running TDE with nShield HSMs the same Security World can be used or if preferred an entirely different Security World can be implemented. If you prefer to use a different Security World you will need further HSMs as the nShield HSM can only host a single Security World instance at any one time.

### Security Worlds and key protection

This section covers the options for Security World when using Always Encrypted. Always Encrypted uses the nCipher CNG provider; there are certain restrictions on the use of this provider concerning methods of authentication and operations that are available. The table below shows the restrictions on HSM key protection methods available when using the Thales nCipher CNG provider.

<table>
<thead>
<tr>
<th>Security World Type</th>
<th>Protection type</th>
<th>Supported</th>
<th>Works in Pool mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>FIPS 140-2 level 2</td>
<td>Module</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>Softcard</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>Operator Card Set 1/ n</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>Operator Card Set k/ n</td>
<td>Yes</td>
<td>No</td>
</tr>
</tbody>
</table>

**Table 1: Supported key protection methods for nCipher CNG provider**

### Application Key Tokens

Application Key Tokens are an encrypted form of a Security World generated cryptographic key. These Key Tokens must not be mistaken for or regarded as being a “Key” in or of itself. The key is at all times obfuscated in this encrypted form and is only available for use as a cryptographic key when copied to the FIPS 140-2 level 3 security boundary of a correctly configured Thales Hardware Security Module.
Configuring nShield Hardware Security Modules for use with Always Encrypted

Ensure that the Thales Security World software is installed on the on-premises client server(s) utilizing the Always Encrypted feature.

Install and register the CNG provider

Once the Security World Software has been installed you must run the CNG install wizard to install and register the Thales Key Storage Provider (KSP). This can be performed via the CNG install wizard that can be found in the “Apps By name” screen of the Desktop.

Click the start button and then click on the to access all applications. Look for the recently installed nCipher utilities.

Double click the CNG configuration wizard. (If the User Access Control prompt pops up click “YES” to continue.)

Figure 2: Install and register nShield provider

Figure 3: CNG install Welcome screen
The following screen (Figure 4) prompts you to enable Pool Mode. Leave the default value with the check box unticked and click “Next”.

![Figure 4: Select to enable / disable Pool Mode](image)

If you already have a Security World that you intend to use for Always Encrypted the next screen will allow you to select to “Use the existing security world”. If you do not currently have a Security World or would like to create a new Security World then check the “Create a new Security World” radio button and click “Next” (for the purposes of this integration guide we have chosen to use an existing Security World).

**Note:** If you are creating a new Security World please refer to the Thales nShield documentation for details on creating and configuring a new Security World.

Ensure that the Set Module States show the available modules as:

- Mode = initialisation
- State = (pre-) initialisation

![Figure 5: Set Module States](image)

Click “Next”.
Leave the “Enable this module as a remote target.” unchecked (Figure 6). (Please be aware that this is not to be confused with the nShield Remote Administration* utility).

Click “Next”. If you are using an existing Security World you must have the “World” file in the %NFAST_KMDATA%\local folder. Be prepared to present the quorum of Administrator cards.

When the ACS quorum has been presented, and the Security World loaded / created, return the HSM to “Operational” mode and choose the appropriate protection method for the Column Master Key. (Figure 8: Set Key Protection)

* For details on Remote Administration setup and configuration please refer to the nShield Documentation on the DVD that came with your Thales HSM.
Proceed to create an Operator Card Set by selecting “Operator Card Set protection” and enter a name for your card set; ensure that the “Always use the wizard when creating or importing keys” is de-selected. Enter the card set name (this field is mandatory) then enter the required $K$ of $N$ value; (consult your security policy document for details on correct values to enter here). Carefully consider which of the optional values to set for the Operator Card Set. Please refer to the description in the table below for further details. Please note that by default the OCS is created as non-persistent.

Click “Next” to proceed to create the Operator Card Set.

<table>
<thead>
<tr>
<th>Term</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Card set name</td>
<td>Card set name must be supplied, unlike naming of individual cards which is optional.</td>
</tr>
<tr>
<td>Number of cards required</td>
<td>This relates to $K$ of $N$ where the value $[K]$ = the necessary number of cards required to complete authentication (the quorum) and $[N]$ = the total number of cards available. The value for $K$ should be less than $N$. We do not recommend creating card sets in which $K$ is equal to $N$ because an error on one card would render the whole card set unusable.</td>
</tr>
<tr>
<td>Card set has a time-out</td>
<td>This allows a specified period of time, in seconds, where keys protected by any given OCS remain loaded in the HSM for use by your application. Once the time period has expired, all keys loaded under the OCS will be forcibly removed from the HSM such that they are no longer available. Time-outs operate independently of OCS persistence</td>
</tr>
<tr>
<td>Persistent</td>
<td>Keys protected by a persistent card set can be used for as long as the application that loaded the OCS remains connected to the hardware security device (unless that application removes the keys). A key protected with a persistent OCS card does not need the card to be present in the slot once the key is loaded.</td>
</tr>
<tr>
<td>Non-persistent</td>
<td>Keys protected by a non-persistent card set can only be used while the last required card of the quorum remains loaded in the smart card reader of the Thales hardware security device. The keys protected by this card are removed from the memory of the device as soon as the card is removed from the smart card reader.</td>
</tr>
<tr>
<td>Usable remotely</td>
<td>The Remote Operator feature enables the contents of a smart card inserted into the slot of one module (the attended module, such as a client module) to be securely transmitted and loaded onto another module (an unattended module, such as the nShield Connect). This is useful when you need to load an OCS-protected key onto a machine to which you do not have physical access (because, for example, it is in a secure area). This feature is deprecated in favour of Remote Administration which was launched with version 12.00 of the Thales nShield Security World software.</td>
</tr>
<tr>
<td>Recoverable PP</td>
<td>The option allows the recovery of a lost or forgotten pass phrase. For further details on recovery operations and Security World settings please refer to the HSM documentation supplied on the Security World media disk.</td>
</tr>
</tbody>
</table>
If you wish to give a name to each card, do so here, select to enter a pass phrase if required, enter and confirm the pass phrase before clicking on “Next” to create the OCS.

**Note:** You must have the \( N \) value of cards (where \( N \) is the total number of cards in the Set) available for this operation before you commence. Insert a card into the attached HSM card reader or the TVD (Trusted Verification Device) if you are using the Remote Administration feature, when you are prompted to do so.

---

**Figure 9: Writing the Operator Card Set**

You do not have to give individual cards names, but if you wish, enter the name of the card in the appropriate field. Similarly, you do not have to give the cards a pass phase, but enter one if appropriate for your security policy. Click “Next”.

---

**Figure 10: Register CNG Providers**

**Note:** If you are using Remote Administration you may need to select the `<Back` button and then return via `Next >` to the ESN confirmation screen. Re-select “OK” on the TVD to continue creating the OCS.
The Thales nCipher CNG providers will now be installed and the key Storage Provider will be registered. To confirm that the KSP has been successfully registered open either a Command Line Interface or PowerShell (right click and “Run as Administrator”) and run the following command:

```bash
>cnglist.exe --list-providers
```

```
PS C:\WINDOWS\system32> cnglist.exe --list-providers
Microsoft Key Protection Provider
Microsoft Passport Key Storage Provider
Microsoft Platform Crypto Provider
Microsoft Primitive Provider
Microsoft Smart Card Key Storage Provider
Microsoft Software Key Storage Provider
Microsoft SSL Protocol Provider
Windows Client Key Protection Provider
nCipher Primitive Provider
nCipher Security World Key Storage Provider
```

You should see the “nCipher Security World key Storage Provider” listed (Circled in red, above). You will find the provider in the registry at this location:

HKEY_LOCAL_MACHINE\SYSTEM\ControlSet001\Control\Cryptography\Providers\nCipherSecurityWorldKeyStorageProvider

![Figure 11: CNG provider registry Path](image-url)
Always Encrypted using SSMS

Creating the Always Encrypted Column Master Key using the nCipher KSP
Once you have successfully installed the nCipher CNG Key Storage Provider you can begin to configure Always Encrypted.

From the “Apps by name” desktop environment, select the Microsoft SQL Server Management Studio and connect to the desired database. Once connected to the database the first thing you will need to do is create a Column Master Key. This key will encrypt all subsequent Column Encryption keys (CEKs).

Using Object Explorer, select the Security directory under the desired Database (In the example below this can be seen as “TestDatabase”). Click to expand “Always Encrypted Keys”.

Select: <Your_database> > Security > Always Encrypted Keys > Column Master Keys. Right click on “Column Master Keys” and select > New Column Master Key… the “New Column Master Key” dialogue box will open (Figure 13).

Figure 12: New Column Master Key
In the **Name** field, enter a meaningful name for the CMK, e.g. MyCMK.

From the drop down list select the "Key Storage Provider (CNG)" option. This will then present the option to "**Select a provider**". Choose the “nCipher Security World Key Storage Provider” from the drop down list and click Generate Key to create a new CMK using the nShield HSM and CNG KSP.

If the “nCipher Security World Key Storage Provider” is not visible you will need to ensure that you have correctly installed and registered the Thales Key Storage Provider.

![Image of Generate CMK dialogue]

**Figure 13: Generate new CMK**

The “nCipher Key Storage Provider – Create key” dialogue will open.

![Image of nCipher KSP - Create new key]

**Figure 14: nCipher KSP - Create new key**

Click “Next” to select key protection options.
The “Select a method to protect new key” dialogue box will open allowing you to select the appropriate method for your security policy. Select to use either Module or Operator Card Set protection. (In this example we are using Operator Card Set Protection).

![Select module or OCS](image)

**Figure 15: Select module or OCS**

The following screen will prompt you to select which Operator Card Set to use for the CMK. (If you have multiple OCS all currently available Operator Card sets will be listed) Operator Card sets will be listed in the left hand field (Figure 16).

![Select card set by name](image)

**Figure 16: Select card set by name**

Select the OCS that you want to use and click “Finish”. The next two screens will prompt you to enter the passphrase for the selected OCS, if one exists, and confirm that card reading completed successfully. (Ensure that you have the correct OCS available)
You will now have a Column Master Key called MyCMK protected by the card set, AESQL. The newly generated CMK will be visible in the Name field.
To confirm the key has been successfully created using the Thales nShield Key Storage Provider open either a CLI or PowerShell. (This must be done with elevated permissions; right click and select “Run as Administrator”). Run the Thales utility `nfkminfo.exe` with the `-k` argument. You should see something similar to the output, seen below.

```
nfkminfo.exe -k
Key list - 1 keys
AppName caping Ident s-1-5-21-1277476411-3880915791-1682396242-1002--
7cbbd9d5477b6d2ed4b6df83e3fa50ac3745b855
```

For further information about the key, including its name, and protection (i.e., Module or Operator Card Set) run `>nfkminfo` with the `<AppName>` and `<Ident>` as reported by `>nfkminfo -k`, above.

Example Key information:

```
nfkminfo.exe -k caping s-1-5-21-1277476411-3880915791-1682396242-1002--
7cbbd9d5477b6d2ed4b6df83e3fa50ac3745b855

Key AppName caping Ident s-1-5-21-1277476411-3880915791-1682396242-1002--
7cbbd9d5477b6d2ed4b6df83e3fa50ac3745b855
BlobKA length 1052
BlobPubKA length 444
BlobRecoveryKA length 1464
name "MyCMK"
hash 94f0bd3bcf6cc1f07a06086bb7918961c67747f0
recovery Enabled
protection CardSet
other flags PublicKey !SEEAppKey !NVMemBlob +0x0
gentime 2017-09-14 12:59:12
SEE integrity key NONE
```

Click OK, the database now has a Column Master Key protected by the Security World under OCS protection.

To view the new Column Master Key use the SQL Object Explorer. Navigate to the relevant database and expand by clicking the + sign. Expand the “Security” folder and then expand the “Always Encrypted Keys” Folder. You will find two folders, one for the Column Master Key(s) and one for the Column Encryption Key(s).

![Figure 20: New CMK](image-url)
**Enable Always Encrypted.**

To Enable Always Encrypted and generate a Column Encryption Key, right click on the required database, in this example we shall use TestDatabase, right click and in the “Tasks” tab select to “Encrypt Columns...” this will open the Always Encrypted wizard.

![Figure 21: Encrypt Columns](image)

If you don’t want the Introduction screen presented each time you run the wizard, check the “Do not show this page again” box. Click “Next”.

The Column Selection screen allows you to choose the type of Column Encryption Key and specify the columns you want to encrypt.

![Figure 22: Column selection and encryption type](image)

**Note:** The “Apply one key to all checked columns” is shaded out until such time as you have two or more CEKs available. You will then also have the option to select the CEK for any given column via the drop down list beneath the “Encryption Key” option.
Under “Encryption Type” click to select the column(s) to encrypt by checking the appropriate box to the left of the column name, you can then select the encryption method from the drop down box beneath “Choose Type” Encryption is either:

- Deterministic
- Randomized
- Plaintext (only available to revert encrypted columns to an unencrypted state)

Click “Next”.

On the Master key Configuration page, Make sure that you select the CMK that was generated using the nCipher Key Storage Provider and protected by the HSM and click next.

The process of encrypting your database records can take a considerable amount of time, depending on the size / quantity of data. To mitigate the possibility of data corruption occurring as records are encrypted whilst being updated, it is advisable to back up the database and to only perform this activity when the database is off-line.

In this case we will continue and run the encryption straight away. Select the radio button, “Proceed to finish now” this will begin the process of creating the CEK and using it to encrypt the specified column in the database. Click “Next” to view the Summary page.
N.B. **Run Settings**: It is recommended that maintenance downtime be scheduled for this activity.

This page allows you to verify your configuration choices and amend if necessary.

![Figure 25: Verify Settings](image)

The next operation requires the Operator Card Set quorum to be available.

Before you can create a CEK you must first load the CMK. The following screen (Figure 26) will prompt you to present the OCS protecting the Column Master Key. Present the OCS quorum and enter the passphrase, continue by clicking “Finish”.

![Figure 26: Load CMK](image)
You will be prompted for the Operator Card passphrase, enter the passphrase and click “Next”.

Click “Finish” to complete the loading of the CMK into the memory of the HSM this will allow it to securely encrypt the Column Encryption Key.
Next, the CEK shall be generated and protected by an OCS protected Column Master Key.

Click “Finish” to proceed.

Figure 29: Load key

Insert the quorum from the Operator Card set and enter the passphrase(s) when prompted.

Figure 30: Enter passphrase
The following screen reports on the status of the Operator Card reading operation.

![Card reading complete](image)

**Figure 31: Card reading complete**

Providing the Operator Card(s) where correctly read the CEK will have been created.

![CEK successfully encrypted column](image)

**Figure 32: CEK successfully encrypted column**

The Results page will report that the “CEK was generated and the requested / specified columns are now encrypted. You can now click “Close” to exit the Always Encrypted Column Encryption Key wizard.
If you now open the table by right clicking on the dbo.Table and selecting “Select Top 1000 Rows” you will see that the column that was chosen for encryption now appears as ciphertext (i.e. an encrypted value).

To show the encrypted columns in plaintext (i.e. Decrypted) you should disconnect from the database and reconnect with the given additional connection parameter. This is entered from the “Connect to Database Engine” logon screen. Select the required server name and click on “Options>>” Go to “additional Connection Parameters” and add the connection string “Column Encryption Setting = enabled” (without parenthesis “)”) and then click “Connect”.

When you now run the query on the table you will now see the original values decrypted by the Column Encryption Key.

Figure 33: Showing encrypted columns
Removing column encryption

If you want to remove the protection provided by Always Encrypted column encryption this can be done using the SQL Server Management Studio Object Explorer.

To remove Column Encryption from a specific or multiple data column(s):

Right click on the required database and in the “Tasks” menu select “Encrypt Columns”.

![Object Explorer](image)

**Figure 34: Select Encrypt Columns...**

From the Introduction screen, select “Next” to get to the Column Selection page. Click on the field “Encryption Type” to enter your preferred option for this value.

![Column Selection](image)

**Figure 35: Choose the option - Plaintext**

From the drop down list select “Plaintext” then click “Next”.

![Column Selection](image)
As there is no key to configure this time click “Next” to proceed straight to the **Run Settings** page. If the database is live at this point, you should first take it off-line before proceeding to remove the column encryption.

Select “Proceed to finish now” and click Next.

![Run Settings](alwaysencrypted.png)

**Figure 36: Confirm that database is off-line**

The following page will provide a review summary for the requested operations.

![Summary](alwaysencrypted.png)

**Figure 37: Review column decryption state**

Check to ensure that the correct Decrypt column(s) are listed and click “Finish”. The “Performing encryption operations” should show as “Passed”.

Microsoft SQL Server 2016 Always Encrypted
You have successfully removed Always Encrypted column encryption from your database. When you next log into the database, you can remove the **Column Encryption Setting = enabled** string from the “Additional Connection Parameters” field of the database login screen. When you now view your database table via, “Select Top 1000 Rows” you should see all columns in plaintext (i.e. an unencrypted state).
Always Encrypted using PowerShell: without Role Separation

Install and Configure SqlServer PowerShell module

All sessions must be executed in an Administrator Shell, to facilitate this open Powershell (or PowerShell ISE) by right click and select “Run as Administrator”.

Decide on the level of security required around the running of scripts and change the ExecutionPolicy setting accordingly. Before updating PowerShellGet or PackageManagement, install the latest Nuget provider.

Open a PowerShell session as Administrator and run:

```powershell
Install-PackageProvider Nuget -force -verbose
```

Next update PowerShellGet:

```powershell
Install-Module -name PowerShellGet -force -verbose
```

Then download and install the SqlServer module to configure Always Encrypted using PowerShell.

```powershell
Install-Module -name SqlServer -force -verbose -allowclobber
```

**Note:** The “-allowclobber” parameter allows you to import the specified commands if it exists in the current session.

(Once installed, If you are using PowerShell ISE refresh the Commands pane if you are using PowerShell open a new session. Confirm the install by running:

```powershell
Get-Module -list -name SqlServer
```

You should see something similar to the output below:

```
Directory: C:\Program Files\WindowsPowerShell\Modules
ModuleType Version    Name          ExportedCommands
---------- -------    ----          ----------------
Manifest   21.0.17152 SqlServer     {Add-SqlColumnEncryptionKeyValue, Complete-SqlColumnMasterKeyRotatio...
```

Figure 39: SqlServer module installed

Install the Thales nCipher CNG provider

Please refer to the section “Install and register the CNG provider” on page 8 of this guide for details on installing and registering the nCipher CNG provider.

**Note:** if using an existing Security World ensure that the “Use existing Security World” check box is ticked.

The provider location can be found at: 
HKEY_LOCAL_MACHINE\SYSTEM\ControlSet001\Control\Cryptography\Providers\nCipherSecurityWorldKeyStorageProvider
Creating the Always Encrypted Column Master Key using the nCipher KSP

Once you have successfully installed the nCipher CNG Key Storage Provider and registered for use with Module protection you can begin to configure Always Encrypted.

**Note:** Always ensure that you check and confirm all values are adjusted according to your environment; the values in this integration guide are example values only. The bespoke values have been highlighted throughout in orange.

Generate a CNG RSA key pair for use as a Column Master Key:

```powershell
$cngProviderName = "nCipher Security World Key Storage Provider"
$cngAlgorithmName = "RSA"
$cngKeySize = 2048 # Recommended key size for Always Encrypted column master keys
$cngKeyName = "AE CMK" # Name identifying your new key in the KSP
$cngKeyParams.provider = $cngProvider
$cngKeyParams.KeyCreationOptions = [System.Security.Cryptography.CngKeyCreationOptions]::OverwriteExistingKey
$cngKeyParams.Parameters.Add($keySizeProperty)
$cngKey = [System.Security.Cryptography.CngKey]::Create($cngAlgorithm, $cngKeyName, $cngKeyParams)
```

The above example will generate a 2048 bit RSA key pair with Name AECMK (highlighted orange). The resulting key is encrypted whilst in the HSM and then pushed to the requesting On-Premise Client server where it is stored as an Application Key Token in the %NFAST_KMDATA%\local folder (C:\ProgramData\nCipher\Key Management Data\local).

Next invoke the > New-SqlCngColumnMasterKeySettings cmdlet:

```powershell
## Specify the Column Master Key settings for importing into the database:
$CmkSettings = New-SqlCngColumnMasterKeySettings -CngProviderName "nCipher Security World Key Storage Provider" -KeyName "AECMK"
```

Then create the Column Master Key via:

```powershell
New-SqlColumnMasterKey "AECMK" -ColumnMasterKeySettings $CmkSettings -Path SQLSERVER:\SQL\server_name\DEFAULT\Databases\your_database
```

Creating the Column Encryption Key

Once the Column Master Key has been successfully generated create a Column Encryption Key using the example below:

```powershell
New-SqlColumnEncryptionKey -Name "CEK" -ColumnMasterKeyName "AECMK" -Path SQLSERVER:\SQL\server_name\DEFAULT\Databases\your_database
```

The resulting Column Encryption Key (CEK) is a 256 bit symmetric key protected by the Column Master Key (CMK) this is achieved by calling the EncryptColumnEncryptionKey method of the SqlColumnEncryptionCNGProvider provider class.
Encrypting Columns with the Column Encryption Key

Open a PowerShell (or PowerShell ISE) session with elevated permissions (right click and select “Run as Administrator”) and run the following to encrypt a given column in the specified database. Adjust the values highlighted in Orange to those suitable for your database name and data columns that you want to encrypt.

- EncryptionType values are one of either:
  
  - Deterministic
  - Randomized
  - Plaintext (only available to revert encrypted columns to an unencrypted state)

```sql
#Import ModuleSqlServer
Import-ModuleSqlServer
# Set up connection and database SMO objects
$sqlConnectionString = "Data Source=server_name;Initial Catalog=your_database;Integrated Security=True;MultipleActiveResultSets=False;Connect Timeout=30;Encrypt=False;TrustServerCertificate=False;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.NationalIdNumber -EncryptionType Randomized -EncryptionKey "CEK"
Set-SqlColumnEncryption -ColumnEncryptionSettings $encryptionChanges -InputObject $smoDatabase

Remove Always Encrypted Column Encryption

To remove column encryption from previously encrypted column data, replace -EncryptionType value (i.e. either Randomized or Deterministic) with the string Plaintext and execute.

N.B. If the database is live at this point, you should first take it off-line before proceeding to remove the column encryption.

```sql
#Import ModuleSqlServer
Import-ModuleSqlServer
# Set up connection and database SMO objects
$sqlConnectionString = "Data Source=server_name;Initial Catalog=your_database;Integrated Security=True;MultipleActiveResultSets=False;Connect Timeout=30;Encrypt=False;TrustServerCertificate=True;Packet Size=4096;Application Name=Microsoft SQL Server Management Studio";Column Encryption Setting=Enabled"
$smoDatabase = Get-SqlDatabase -ConnectionString $sqlConnectionString

# Change encryption schema
$encryptionChanges = @()

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

The Always Encrypted encrypted data will revert to plaintext. (If your database is protected by TDE then the data is still being encrypted whilst at rest). When you next log into the database you can remove the Column Encryption Setting = enabled string from the “Additional Connection Parameters” field of the database login screen. When you now view your database table via, “Select Top 1000 Rows” you should see all columns in plaintext (i.e. an unencrypted state).

Note: When removing Always Encryption from your database columns, ensure that all columns appear in plaintext. You must delete any Column Encryption Keys (CEK) before you can drop the Column Master Key(s) (CMK)
**Query the encrypted columns**

In order to successfully query the encrypted columns ensure that you connect to the database with the correct connection parameters. You may, upon first time connection, receive the “Parameterization for Always Encrypted” prompt. Click on “Enable” to proceed.

![Parameterization for Always Encrypted](image1.png)

*Figure 40: Parameterization for Always Encrypted*

When initiating a connection to the database, select “Options” -> Select “Additional Connection Parameters” and enter: *Column Encryption Setting = enabled* in the provided field.

![Adjust Connection Parameters](image2.png)

*Figure 41: Adjust Connection Parameters*
Always Encrypted using PowerShell: with Role Separation

Install and Configure SqlServer PowerShell module and verify KSP

All sessions must be executed in an Administrator Shell, to facilitate this open Powershell (or PowerShell ISE) by right click and select “Run as Administrator”. The following operations need to be run on all SQL Always Encrypted servers and the Database server(s) in order to align all with the latest SqlServer package.

Note: Always ensure that you check and confirm all values are adjusted according to your environment; the values in this integration guide are example values only. The bespoke values have been highlighted throughout in orange.

Before updating PowerShellGet or PackageManagement, install the latest Nuget provider.

Open a PowerShell session as Administrator and run:

```
Install-PackageProvider Nuget -Force -Verbose
```

Next update PowerShellGet:

```
Install-Module -Name PowerShellGet -Force -Verbose
```

Then download and install the SqlServer module to configure Always Encrypted using Power Shell.

```
Install-Module -Name SqlServer -Force -Verbose -AllowClobber
```

Note: The “-AllowClobber” parameter allows you to import the specified commands if it exists in the current session.

Once installed, If you are using PowerShell ISE refresh the Commands pane if you are using PowerShell open a new session. Confirm the install by running:

```
Get-Module -list -Name SqlServer
```

You should see the following reported:

```
Directory: C:\Program Files\WindowsPowerShell\Modules
ModuleType Version Name ExportedCommands
---------- ------- ------ ----------------
Manifest 21.0.17152 SqlServer {Add-SqlColumnEncryptionKeyvalue, Complete-SqlColumnMasterKeyRota...
```

Verify that the Thales CNG KSP is installed correctly:

```
cnglist.exe -list-providers
Microsoft Key Protection Provider
Microsoft Passport Key Storage Provider
Microsoft Platform Crypto Provider
...
ncipher Primitive Provider
ncipher Security World Key Storage Provider
```

Figure 42: Verify Thales KSP is installed
For the purpose of this guide when integrating with role separation, roles are defined as Duty Role. The table below shows the separation and function of these duty roles with reference to Security Administrator and Database Administrator.

<table>
<thead>
<tr>
<th>Process</th>
<th>Duty Role</th>
</tr>
</thead>
<tbody>
<tr>
<td>Generating the CMK</td>
<td>Security Administrator</td>
</tr>
<tr>
<td>Generating / encryption of CEK</td>
<td>Security Administrator</td>
</tr>
<tr>
<td>Defining the CMK and CEK in the database</td>
<td>Database Administrator</td>
</tr>
<tr>
<td>Encrypt database columns with CEK</td>
<td>Security Administrator</td>
</tr>
</tbody>
</table>

**Creating the Always Encrypted Column Master Key using the nCipher KSP**

Once you have successfully installed the nCipher CNG Key Storage Provider and registered for use with Module protection you can begin to configure Always Encrypted.

The Security Officer must have administrator rights on the Client Server being configured to use Always Encrypted. The following sections are divided between the Security Administrator and the Database Administrator. The DBA should not have access to the Client server.

**Duty Role: Security Administrator**

Confirm that the SqlServer module is present by running:

```
Get-Module -list -Name SqlServer
```

Directory: C:\Program Files\windowsPowerShell\Modules

```
Manifest Version    Name                                ExportedCommands
---------- ------- ----------------------------------------- -----------------------------------
Manifest 21.0.17178 SqlServer                           {Add-SqlColumnEncryptionKeyValue, Complete-SqlColumnMasterKeyRotation, Get-SqlColumnEncryptionKey, Get-SqlCo...
```

Generate a CNG RSA key pair for use as a Column Master Key: Choose a suitable name for the Column Master Key ($cngKeyName highlighted orange in the example, below) for your particular deployment.

```
$cngProviderName = "nCipher Security World Key Storage Provider"
$cngAlgorithmName = "RSA"
$cngKeySize = 2048 # Recommended key size for Always Encrypted column master keys
$cngKeyName = "AECMK" # Name identifying your new key in the KSP
$cngKeyParams.provider = $cngProvider
$cngKeyParams.KeyCreationOptions = [System.Security.Cryptography.CngKeyCreationOptions]::OverwriteExistingKey
[System.BitConverter]::GetBytes($cngKeySize),
[System.Security.Cryptography.CngPropertyOptions]::None);
$cngKeyParams.Parameters.Add($keySizeProperty)
$cngKey = [System.Security.Cryptography.CngKey]:Create($cngAlgorithm, $cngKeyName, $cngKeyParams)
```

The above will generate a 2048 bit RSA key pair. The resulting key is encrypted whilst in the HSM and then pushed to the requesting On-Premise Client server, where it is stored as an Application Key Token in the %NFAST_KMDATA%\local folder (C:\ProgramData\nCipher\Key Management Data\Token).
Confirm that the key was successfully generated and exists in the Thales %KMDATA_LOCAL% directory by running `nfkminfo -k` this will list the available keys associated with the Security World:

```
nfkminfo.exe -k
AppName caping Ident   s-1-5-21-1277476411-3880915791-1682396242-1002--
7cbb9d5477b6d2dd4b6df83e3fa50ac3745bb85
```

```
PS C:\WINDOWS\system32> nfkminfo.exe -k caping s-1-5-21-1277476411-3880915791-1682396242-1002--
7cbb9d5477b6d2dd4b6df83e3fa50ac3745bb85
Key AppName caping Ident   s-1-5-21-1277476411-3880915791-1682396242-1002--
7cbb9d5477b6d2dd4b6df83e3fa50ac3745bb85
name "AECMK"
hash 9f215d32a765a708e4fcb92678ce43d0f473955b
recovery Enabled
protection Module
```

The Security Administrator can now create the `SqlCngColumnMasterKeySettings` object which will reference the CMK application key token created using the nCipher KSP.

```
$CmkSettings = New-SqlCngColumnMasterKeySettings -CngProviderName "nCipher Security World Key Storage Provider" -KeyName AECMK
```

Finally invoke the `New-SqlColumnEncryptionKeyEncryptedValue` cmdlet. This will produce the encrypted value of the CMK which will need to be passed to the Database Administrator and is required for generating a Column Encryption Key.

```
New-SqlColumnEncryptionKeyEncryptedValue -TargetColumnMasterKeySettings $CMKSettings
```

```
0x0162000016E0063006900700068006500720020007300650063007500730074007900200077006F006C00640020006B00650079002000730074006F0072006100670065002000700072006F00760069006400650072002F006100650063006D006B00989CD11831C74395E380A8F16A25104D03E98390BD8AF12252062B53Ae80A00C7CFA1FC284914A38CB6A9796C55CF5981318DC1B8A862B032F1430B26F1BAC241EEB4810C8F9DDB6523E34E6F7EE0B36FA53822BEF1964C3E2A1E4EDB056700AA2B25BFB9B34679C9A84D78275F0945429E97639535EF79E8EB36593978578BB621C2D5FC322464FECF2F172A240872FC53EACBECFCF2D8B44554357C8EDB6F1CC8A3070BF0FA9F430589D738AA2CE17800E7210CA18437FA3F6333075A3F282353CA6E6B4C017F3ADA03D7FECB6CEC5F58F6B15646B5D3D6A6FCC4A4B4D36D9130D335825B642A6ED2B6FE1692010943E325253072730635822BDC2CAD8DC13F4B8478686B0ADDAD7208A6508A57D482C90A438EE60138CC26398D5CE589AB659FEADS579C28CF4BCEAA96F1B268F50F5B381C4B8763C27CD1E1307C9A6F612CF8BC8CF309A4173648FA78C5D9A8FEEFC877FCDC44CC06AFT5043A992C757890A9D9C3D0AEE78002AC3E6D6DCF1C18AEBE9B390518F59DCE4F77FE51453C8443C32DD62309CC92F29F32A80DA9883B363C2DFF36620318175B937636DF696D5ED9C37F51655E181207165A986AAEF60271B705B8838D96BEC648745A65EFE9DF7F48AF54FFE187FE0598AC47CA306B232018898989DFAA574688CA185816FEEAFA9F9A87072CE0E853675E2369CE3ED912F6
``
Duty Role: Data Base Administrator
Open PowerShell ISE as Administrator and run the following in order to create a SqlcngcolumnMasterKeySettings object that contains information about the location of your column master key. Ensure that the Sql Server module has been imported.

New-SqlCngColumnMasterKeySettings -CngProviderName "nCipher Security World Key Storage Provider" -KeyName AECMK

KeyStoreProviderName KeyPath
MSSQL_CNG_STORE nCipher Security World Key Storage Provider/AECMK

$CmkSettings = New-SqlCngColumnMasterKeySettings -CngProviderName "nCipher Security World Key Storage Provider" -KeyName AECMK

New-SqlColumnMasterKey "AECMK" -ColumnMasterKeySettings $CmkSettings -Path SQLSERVER:\SQL\server_name\DEFAULT\Databases\your_database

Name
AECMK

You can confirm the presence of the newly imported CMK using Object Explorer in SSMS.

The New-SqCngColumnEncryptionKey cmdlet creates a column encryption key object in the form of an “Encrypted Value” in the database. A Column Encryption Key object encapsulates an encrypted value of a symmetric cryptographic key, i.e. the Column Encryption Key; this can be subsequently used to encrypt database columns using the Always Encrypted feature. When attempting to copy the encrypted value, ensure that the entire value is written to a single line, appended with the path to the database.
The screenshot above shows the output confirming the creation of a new CEK, called “AECEK1” this was generated and then encrypted using the encapsulated CMK metadata provided to the DBA by the Security Administrator. 

Note: When adding the –Path you should append this after the encrypted value:

```
....C04234151 -Path SQLSERVER:\SQL\server_name\DEFAULT\Databases\your_database
```

You can confirm the presence of the CEK using SSMS Object Explorer to view the new Always Encrypted key.

This concludes role separation activities performed by the Data Base Administrator. The Security Administrator can now connect to the Database and use the defined keys to encrypt the desired columns using Always Encrypted.
Duty Role: Security Administrator
The Security Administrator can now connect to the database either through the SQL Server Management Studio or PowerShell and use the provisioned key Metadata to encrypt the required columns.

Encrypt Columns using SSMS
To encrypt a column using SSMS please refer to the chapter on “Always Encrypted using SSMS” this can be found on page 14 of this document.

To encrypt columns using PowerShell please refer to the section below.

Encrypt Columns using PowerShell (ISE)

Note: In order to encrypt a column using a CEK the user of the database must have the following “effective” permissions:

ALTER ANY COLUMN MASTER KEY
ALTER ANY COLUMN ENCRYPTION KEY
VIEW ANY COLUMN MASTER KEY DEFINITION
VIEW ANY COLUMN ENCRYPTION KEY DEFINITION

These permissions can be found by right clicking on the relevant database and selecting “Properties” then select “permissions” (The example below has been edited for convenience of illustration).

![Permissions Status](image)

Figure 45: Permissions Status

To encrypt a column using the pre-defined Column Encryption Key (given in the example as AECEK1) use one of the following for EncryptionType value.
Available Options are:

- Deterministic
- Randomized
- Plaintext (only available to revert encrypted columns to an unencrypted state)

Run the below with the relevant server name and database inserted appropriately and your chosen table / column intended for encryption, specify also the Column Encryption Key to use.

```powershell
# Import Module SqlServer
Import-Module SqlServer
# Set up connection and database SMO objects
$sqlConnectionString = "Data Source=$server_name;Initial Catalog=$your_database;Integrated Security=True;MultipleActiveResultSets=False;Connect Timeout=30;Encrypt=False;TrustServerCertificate=False;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.NationalIdNumber -EncryptionType Deterministic -EncryptionKey "AECEK1"
Set-SqlColumnEncryption -ColumnEncryptionSettings $encryptionChanges -InputObject $smoDatabase
```

**Note:** As the Column Master Key Application Key Token only exists in the NFAST-KMDATA%\local directory of a participating Always Encrypted client server, the facility to decrypt a column is dependent on the presence of both the CMK Application Key Token in the NFAST-KMDATA%\local directory and the availability of a correctly configured HSM, thereby rendering all encrypted data available only to those computers configured with and with access too, these requisite resources.

### Remove Always Encrypted Column Encryption

To remove column encryption from previously encrypted column data, replace `-EncryptionType` value (i.e. either Randomized or Deterministic) with the value `Plaintext` and execute.

**N.B.** If the database is live at this point, you should first take it off-line before proceeding to remove the column encryption.

```powershell
# Import Module SqlServer
Import-Module SqlServer
# Set up connection and database SMO objects
$sqlConnectionString = "Data Source=$server_name;Initial Catalog=$your_database;Integrated Security=True;MultipleActiveResultSets=False;Connect Timeout=30;Encrypt=False;TrustServerCertificate=False;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.NationalIdNumber -EncryptionType Plaintext
Set-SqlColumnEncryption -ColumnEncryptionSettings $encryptionChanges -InputObject $smoDatabase
```

The *Always Encrypted* encrypted data will revert to plaintext. (If your database is protected by TDE then the data is still being encrypted whilst at rest). When you next log into the database you can remove the Column Encryption Setting = enabled value from the "Additional Connection Parameters" field of the database login screen. When you now view your database table via, "Select Top 1000 Rows" you should see all columns in plaintext (i.e. an unencrypted state).
**Note:** When removing Always Encryption from your database columns, ensure that all columns appear in plaintext. You must delete any Column Encryption Keys (CEK) before you can drop the Column Master Key(s) (CMK)

**Query the encrypted columns**

In order to successfully query the encrypted columns ensure that you connect to the database with the correct connection parameters. You may, upon first time connection, receive the “Parameterization for Always Encrypted” prompt. Click on Enable to proceed.

**Figure 46: Parameterization for Always Encrypted**

When initiating a connection to the database, select “Options” -> Select “Additional Connection Parameters” and enter: *Column Encryption Setting = enabled* in the provided field.

**Figure 47: Adjust Connection Parameters**
## Glossary of PowerShell SqlServer CMDlets

The following list shows the currently available Always Encrypted PowerShell cmdlets:

<table>
<thead>
<tr>
<th>PowerShell CMDlet</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Add-SqlColumnEncryptionKeyValue</td>
<td>Adds a new encrypted value for an existing column encryption key object in the database.</td>
</tr>
<tr>
<td>Complete-SqlColumnMasterKeyRotation</td>
<td>Completes the rotation of a column master key.</td>
</tr>
<tr>
<td>Get-SqlColumnEncryptionKey</td>
<td>Returns all column encryption key objects defined in the database, or returns one column encryption key object with the specified name.</td>
</tr>
<tr>
<td>Get-SqlColumnMasterKey</td>
<td>Returns the column master key objects defined in the database, or returns one column master key object with the specified name.</td>
</tr>
<tr>
<td>Invoke-SqlColumnMasterKeyRotation</td>
<td>Initiates the rotation of a column master key.</td>
</tr>
<tr>
<td>New-SqlAzureKeyVaultColumnMasterKeySettings</td>
<td>Creates a SqlColumnMasterKeySettings object describing an asymmetric key stored in Azure Key Vault.</td>
</tr>
<tr>
<td>New-SqlCngColumnMasterKeySettings</td>
<td>Creates a SqlColumnMasterKeySettings object describing an asymmetric key stored in a key store supporting the Cryptography Next Generation (CNG) API.</td>
</tr>
<tr>
<td>New-SqlColumnEncryptionKey</td>
<td>Creates a new column encryption key object in the database.</td>
</tr>
<tr>
<td>New-SqlColumnEncryptionKeyEncryptedValue</td>
<td>Produces an encrypted value of a column encryption key.</td>
</tr>
<tr>
<td>New-SqlColumnEncryptionSettings</td>
<td>Creates a new SqlColumnEncryptionSettings object that encapsulates information about a single column’s encryption, including CEK and encryption type.</td>
</tr>
<tr>
<td>New-SqlColumnMasterKey</td>
<td>Creates a new column master key object in the database.</td>
</tr>
<tr>
<td>New-SqlCspColumnMasterKeySettings</td>
<td>Creates a SqlColumnMasterKeySettings object describing an asymmetric key stored in a key store with a Cryptography Service Provider (CSP) supporting Cryptography API (CAPI).</td>
</tr>
<tr>
<td>Remove-SqlColumnEncryptionKey</td>
<td>Removes the column encryption key object from the database.</td>
</tr>
<tr>
<td>Remove-SqlColumnEncryptionKeyValue</td>
<td>Removes an encrypted value from an existing column encryption key object in the database.</td>
</tr>
<tr>
<td>Remove-SqlColumnMasterKey</td>
<td>Removes the column master key object from the database.</td>
</tr>
<tr>
<td>Set-SqlColumnEncryption</td>
<td>Encrypts, decrypts or re-encrypts specified columns in the database.</td>
</tr>
</tbody>
</table>

Alternatively, the full list, along with any additions to the SqlServer module released after this guides publication, can be found via the link below:

https://blogs.technet.microsoft.com/dataplatforminsider/2016/06/30/sql-powershell-july-2016-update/
Feature Details

The information below was taken from the Microsoft website on relational databases;


For further information on Always Encrypted operational capabilities please consult the website via the above URL.

- Queries can perform equality comparison on columns encrypted using deterministic encryption, but no other operations (e.g. greater/less than, pattern matching using the LIKE operator, or arithmetical operations).
- Queries on columns encrypted by using randomized encryption cannot perform operations on any of those columns. Indexing columns encrypted using randomized encryption is not supported.
- A column encryption key can have up to two different encrypted values, each encrypted with a different column master key. This facilitates column master key rotation.
- Deterministic encryption requires a column to have one of the binary2 collations.
- After changing the definition of an encrypted object, execute sp_refresh_parameter_encryption to update the Always Encrypted metadata for the object.

Always Encrypted is not supported for the columns with the below characteristics (e.g. the Encrypted WITH clause cannot be used in CREATE TABLE/ALTER TABLE for a column, if any of the following conditions apply to the column):

- Columns using one of the following datatypes: xml, timestamp/rowversion, image, ntext, text, sql_variant, hierarchyid, geography, geometry, alias, user defined-types.
- FILESTREAM columns
- Columns with the IDENTITY property
- Columns with ROWGUIDCOL property
- String (varchar, char, etc.) columns with non-bin2 collations
- Columns that are keys for nonclustered indices using a randomized encrypted column as a key column (deterministic encrypted columns are fine)
- Columns that are keys for clustered indices using a randomized encrypted column as a key column (deterministic encrypted columns are fine)
- Columns that are keys for fulltext indices containing encrypted columns both randomized and deterministic
- Columns referenced by computed columns (when the expression does unsupported operations for Always Encrypted)
- Sparse column set
- Columns that are referenced by statistics
- Columns using alias type
- Partitioning columns
- Columns with default constraints
- Columns referenced by unique constraints when using randomized encryption (deterministic encryption is supported)
- Primary key columns when using randomized encryption (deterministic encryption is supported)
- Referencing columns in foreign key constraints when using randomized encryption or when using deterministic encryption, if the referenced and referencing columns use different keys or algorithms
- Columns referenced by check constraints
- Columns in tables that use change data capture
- Primary key columns on tables that have change tracking
- Columns that are masked (using Dynamic Data Masking)
- Columns in Stretch Database tables. (Tables with columns encrypted with Always Encrypted can be enabled for Stretch.)
- Columns in external (PolyBase) tables (note: using external tables and tables with encrypted columns in the same query is supported)
- Table-valued parameters targeting encrypted columns are not supported.

The following clauses cannot be used for encrypted columns:

- FOR XML
- FOR JSON PATH
## Troubleshooting

### Common returned errors when encrypting columns:

<table>
<thead>
<tr>
<th>Error</th>
<th>Resolution</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>Set-SqlColumnEncryption : Object reference not set to an instance of an object. At line:16 char:1 + Set-SqlColumnEncryption -ColumnEncryptionSettings $encryptionChanges ...</code></td>
<td>Check the correct CEK is referenced in the call and re-submit.</td>
</tr>
<tr>
<td><code>Get-SqlDatabase : Failed to connect to server &lt;server_name&gt;. At line:7 char:16 + $smoDatabase = Get-SqlDatabase -ConnectionString $sqlConnectionString</code></td>
<td>Check the Database server name is correct and available to the network.</td>
</tr>
<tr>
<td><code>Set-SqlColumnEncryption : Cannot validate argument on parameter 'InputObject'. The argument is null or empty. Provide an argument that is not null or empty, and then try the command again. At line:17 char:83 + .... ColumnEncryptionSettings $encryptionChanges -InputObject $smoDatabase</code></td>
<td>Check database name is correct.</td>
</tr>
</tbody>
</table>
About Thales e-Security

Thales e-Security is the leader in advanced data security solutions and services that deliver trust wherever information is created, shared or stored. We ensure that the data belonging to companies and government entities is both secure and trusted in any environment – on-premise, in the cloud, in data centers or big data environments – without sacrificing business agility. Security doesn’t just reduce risk, it’s an enabler of the digital initiatives that now permeate our daily lives – digital money, e-identities, healthcare, connected cars and with the internet of things (IoT) even household devices. Thales provides everything an organization needs to protect and manage its data, identities and intellectual property and meet regulatory compliance – through encryption, advanced key management, tokenization, privileged user control and high assurance solutions. Security professionals around the globe rely on Thales to confidently accelerate their organization’s digital transformation. Thales e-Security is part of Thales Group.