THALES

Microsoft SQL Server: Integration Guide

THALES LUNA HSM AND DPOD LUNA CLOUD HSM

Document Information

Document Part Number	007-011108-001
Revision	AR
Release Date	13 January 2021

Trademarks, Copyrights, and Third-Party Software

CONTENTS

Overview

This document contains instructions for integrating Microsoft SQL Server with Luna HSM devices or Luna Cloud HSM services. Microsoft SQL Server uses the Extensible Key Management (EKM) feature to enable the use of HSM devices for key storage and cryptographic operations such as key creation, deletion, encryption, and decryption. Luna HSM provides access to Luna EKM, including EKM provider library. To use the HSM devices with SQL server, you must configure the EKM provider option. You must have the basic knowledge of SQL Server and HSM concepts to make full use of the instructions in this document. Using Luna HSMs with Microsoft SQL Server provides the following benefits:

> Secure generation, storage and protection of the Identity signing private key on FIPS 140-2 level 3 validated hardware.

> Full life cycle management of the keys.

> HSM audit trail*.

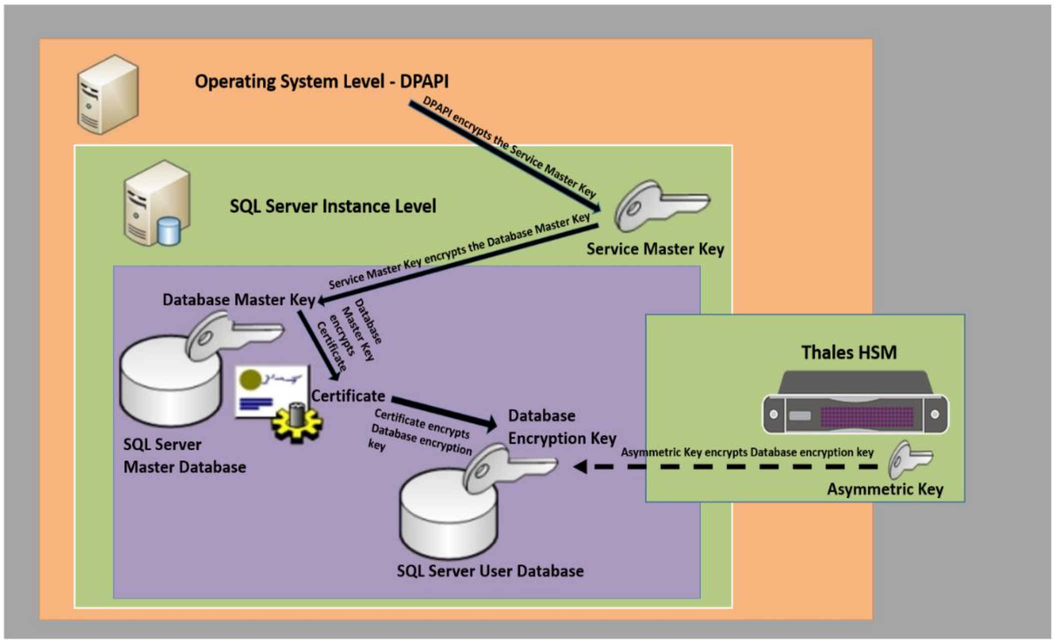
> Significant performance improvements by off-loading cryptographic operations from application servers

*Luna Cloud HSM service does not have access to the secure audit trail

Third Party Application Details

This integration uses the following third party applications:

Microsoft SQL Server

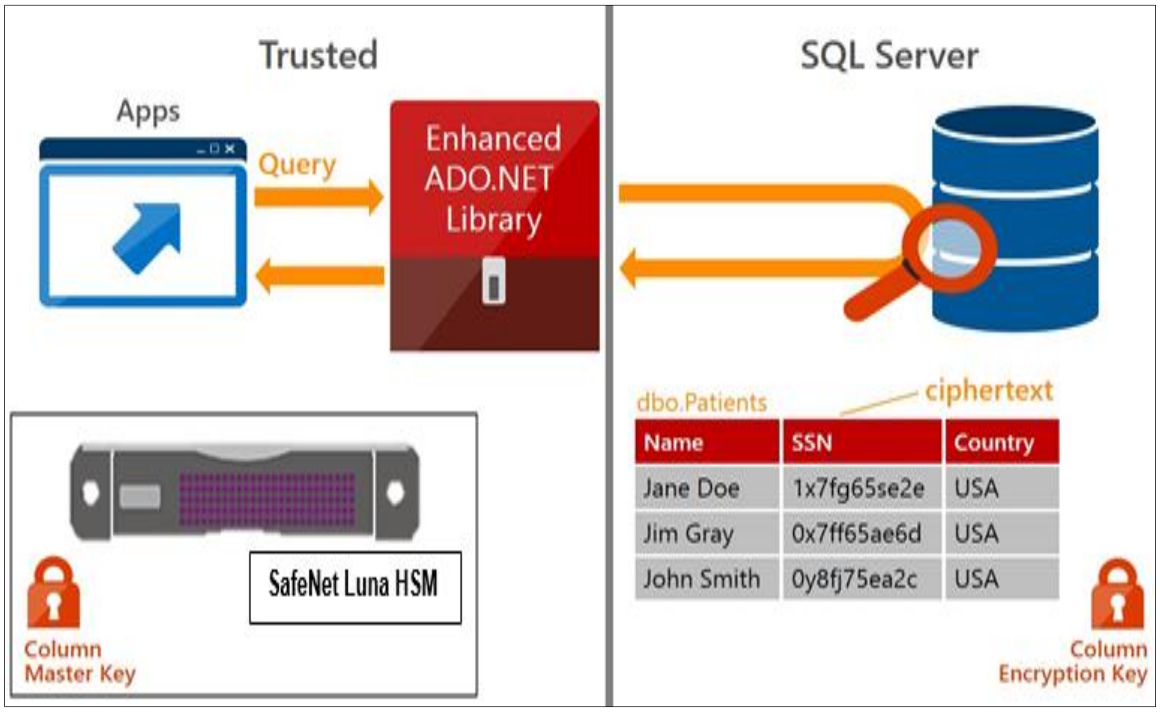

Operating System Level - DPAPI
DPAPI encrypts the Service Master Key
SQL Server Instance Level
Service Master Key
Service Master Key encrypts the Database Master Key
Database Master Key
Database Master Key encrypts Certificate
Certificate
Certificate encrypts Database encryption key
Database Encryption Key
SQL Server Master Database
Thales HSM
Asymmetric Key encrypts Database encryption key
Asymmetric Key
SQL Server User Database

Microsoft SQL Server is a database platform for large-scale online transaction processing (OLTP), data warehousing, and e-commerce applications. It is also a business intelligence platform for data integration, analysis, and reporting solutions. The above diagram shows the relationships between the database master key and the HSM in a Microsoft SQL configuration.

Microsoft SQL Server High Availability (Always On)

The Always On Availability group feature is a high-availability and disaster recovery solution that provides an enterprise-level alternative to database mirroring. An availability group supports a failover environment for a discrete set of user databases, known as availability databases that fail over together. An availability group supports a set of read-write primary databases and one to eight sets of corresponding secondary databases. Optionally, secondary databases can be made available for read-only access and/or some backup operations.

Microsoft SQL Server Always Encrypted



dbo.Patients

Name	SSN	Country
Jane Doe	1x7fg65se2e	USA
Jim Gray	0x7ff65ae6d	USA
John Smith	0y8fj75ea2c	USA

The new feature, called Always Encrypted, is available from SQL Server 2016's first public preview. Always Encrypted adds an extra measure of security while the data is being used, when it is most susceptible to attack. The new security layer addresses that vulnerability by keeping the data encrypted during transactions and computations, and by only giving decryption keys to the client. If anyone else, including a database or system administrator, tries to access that client's database, sensitive data will not appear in plaintext. In order to use SQL Always Encrypted, the following keys are created:

> Column master key

> Column encryption key

A column encryption key is used to encrypt data in an encrypted column. A column master key is a key-protecting key that encrypts one or more column encryption keys. The Database Engine stores encryption configuration for each column in database metadata. However, the Database Engine never stores or uses the keys of either type in plaintext. It only stores encrypted values of column encryption keys and the information about the location of column master keys, which are stored in external trusted key stores, such as a Hardware Security Module (HSM).

Certified Platforms

This integration is certified on the following platforms:

Luna HSM: Luna HSM appliances are purposefully designed to provide a balance of security, high performance, and usability that makes them an ideal choice for enterprise, financial, and government organizations. Luna HSMs physically and logically secure cryptographic keys and accelerate cryptographic processing. The Luna HSM on premise offerings include the Luna Network HSM, Luna PCIe HSM, and Luna USB HSMs. Luna HSMs are also available for access as an offering from cloud service providers such as IBM cloud HSM and AWS cloud HSM classic.

Platforms Tested	EKM Software Version	Microsoft SQL Server
Windows Server 2019+CU2 (KB4536075)	EKM v1.5 EKM v1.4	Microsoft SQL Server 2019
Windows Server 2016 Windows Server 2012 R2	EKM v1.4	Microsoft SQL Server 2017 Microsoft SQL Server 2016

NOTE: This integration is tested in both HA and FIPS mode.

Luna Cloud HSM: Luna Cloud HSM platform provides on-demand, cloud-based HSM and Key Management services through a simple graphical user interface. With Luna Cloud HSM, security is simple, cost effective and easy to manage because there is no hardware to buy, deploy and maintain. As an Application Owner, you click and deploy services, generate usage reports and maintain just the services you need.

Platforms Tested	EKM Software Version	Microsoft SQL Server
Windows Server 2019+CU2 (KB4536075)	EKM v1.5	Microsoft SQL Server 2019
Windows Server 2016	EKM v1.4	Microsoft SQL Server 2017 Microsoft SQL Server 2016

NOTE: SQL Server uses CKM_RSA_PKCS mechanism for encryption which is not allowed by Luna Cloud HSM in FIPS mode. Therefore, SQL Server integration with Luna Cloud HSM works in Non-FIPS mode only.

Prerequisites

Before you proceed with the integration, complete the following tasks:

Configuring Luna HSM

If you are using Luna HSM:

1. Verify the HSM is set up, initialized, provisioned and ready for deployment.

2. Create a partition that will be later used by SQL Server.
3. If using a Luna Network HSM, register a client for the system and assign the client to the partition to create an NTLS connection. Initialize the Crypto Officer and Crypto User roles for the registered partition.
4. Ensure that the partition is successfully registered and configured. The command to see the registered partitions is:

```
C:\Program Files\SafeNet\LunaClient>lunacm.exe

lunacm.exe (64-bit) v10.3.0-273. Copyright (c) 2020 SafeNet. All rights
reserved.

Available HSMs:

   Slot Id ->                0

   Label ->                  SQL

   Serial Number ->          1238696044952

   Model ->                  LunaSA 7.4.0

   Firmware Version ->       7.4.0

   Configuration ->          Luna User Partition With SO (PW) Key Export With
   Cloning Mode

   Slot Description ->       Net Token Slot

   FM HW Status ->           Non-FM
```

5. For PED-authenticated HSM, enable partition policies 22 and 23 to allow activation and auto-activation.

> **NOTE:** Refer to Luna HSM documentation for detailed steps on creating NTLS connection, initializing the partitions, and assigning various user roles.

Set up Luna HSM High-Availability

Refer to the Luna HSM documentation for HA steps and details regarding configuring and setting up two or more HSM boxes on host systems. You must enable the HAOnly setting in HA for failover to work so that if the primary goes down due to any reason all calls automatically route to the secondary until the primary recovers and starts up.

Set up Luna HSM in FIPS Mode

> **NOTE:** This setting is not required for Universal Client. This setting is applicable only for Luna Client 7.x.

Under FIPS 186-3/4, the RSA methods permitted for generating keys are 186-3 with primes and 186-3 with aux primes. This means that RSA PKCS and X9.31 key generation is no longer approved for operation in a FIPS-compliant HSM. If you are using the Luna HSM in FIPS mode, you have to make the following change in the configuration file:

```
[Misc]
RSAKeyGenMechRemap=1
```

The above setting redirects the older calling mechanism to a new approved mechanism when Luna HSM is in FIPS mode.

Configure Luna Cloud HSM Service

You can configure Luna Cloud HSM Service in the following ways:

> Standalone Cloud HSM service using minimum client package

> Standalone Cloud HSM service using full Luna client package

> Luna HSM and Luna Cloud HSM service in hybrid mode

NOTE: Luna Client v10.x or higher is required for configuring Luna HSM device and Luna Cloud HSM service in hybrid mode.

Standalone Cloud HSM service using minimum client package

To configure Luna Cloud HSM service using minimum client package:

1. Transfer the downloaded .zip file to your Client workstation using pscp, scp, or other secure means.
2. Extract the .zip file into a directory on your client workstation.
3. Extract or untar the appropriate client package for your operating system. Do not extract to a new subdirectory; place the files in the client install directory.

   ```
   [Windows]
   
   cvclient-min.zip
   ```

4. Run the **setenv** script to create a new configuration file containing information required by the Luna Cloud HSM service.

   ```
   [Windows]
   
   Right-click setenv.cmd and select Run as Administrator.
   ```

5. Run the **LunaCM** utility and verify the Cloud HSM service is listed.

Standalone Cloud HSM service using full Luna client package

To configure Luna Cloud HSM service using full Luna client package:

1. Transfer the downloaded .zip file to your Client workstation using pscp, scp, or other secure means.
2. Extract the .zip file into a directory on your client workstation.
3. Extract or untar the appropriate client package for your operating system. Do not extract to a new subdirectory; place the files in the client install directory.

   ```
   [Windows]
   
   cvclient-min.zip
   ```

4. Run the setenv script to create a new configuration file containing information required by the Luna Cloud HSM service.

   ```
   [Windows]
   
   Right-click setenv.cmd and select Run as Administrator.
   ```

5. Copy the server and partition certificates from the Cloud HSM service client directory to Luna client certificates directory:

NOTE: Skip this step, if using Luna Client v10.2 or higher.

Cloud HSM Certificates:

`server-certificate.pem`

`partition-ca-certificate.pem`

`partition-certificate.pem`

LunaClient Certificate Directory:

`[Windows default location for Luna Client]`

`C:\Program Files\Safenet\Lunaclient\cert\`

6. Open the configuration file from the Cloud HSM service client directory and copy the XTC and REST section.

   `[Windows]`

   `crystoki.ini`

7. Edit the Luna Client configuration file and add the XTC and REST sections copied from Cloud HSM service client configuration file.

8. Change server and partition certificates path from step 5 in XTC and REST sections. Do not change any other entries provided in these sections.

   NOTE: Skip this step, if using Luna Client v10.2 or higher.

   [XTC]

```
. . .
PartitionCAPath=<LunaClient_cert_directory>\partition-ca-certificate.pem
PartitionCertPath00=<LunaClient_cert_directory>\partition-certificate.pem
. . .

[REST]
. . .
SSLClientSideVerifyFile=<LunaClient_cert_directory>\server-certificate.pem
. . .
```

9. Edit the following entry from the Misc section and update the correct path for the plugins directory:

```
Misc]
PluginModuleDir=<LunaClient_plugins_directory>
```

`[Windows Default]`

```
C:\Program Files\Safenet\Lunaclient\plugins\
```

Save the configuration file. If you wish, you can now safely delete the extracted Cloud HSM service client directory.

10. Reset the ChrystokiConfigurationPath environment variable and point back to the location of the Luna Client configuration file.

[Windows]

In the Control Panel, search for "environment" and select Edit the system environment variables. Click Environment Variables. In both list boxes for the current user and system variables, edit ChrystokiConfigurationPath and point to the crystoki.ini file in the Luna client install directory.

11. Run the LunaCM utility and verify that the Cloud HSM service is listed.

> **NOTE:** Refer to the Luna Cloud HSM documentation for detailed steps for creating service, client, and initializing various user roles.

Luna HSM and Luna Cloud HSM service in hybrid mode

To configure Luna HSM and Luna Cloud HSM service in hybrid mode, follow the steps mentioned under the Standalone Cloud HSM service using full Luna client package section above. In hybrid mode, both Luna and Cloud HSM service will be listed.

> **NOTE:** Luna Client v10.x or higher is required for configuring Luna HSM device and Luna Cloud HSM service in hybrid mode.

Luna Cloud HSM Service in FIPS mode

Luna Cloud HSM service operates in both FIPS and non-FIPS mode. If your organization requires non-FIPS algorithms for your operations, ensure you enable the Allow non-FIPS approved algorithms check box when configuring your Cloud HSM service. The FIPS mode is enabled by default. Refer to the Mechanism List in the SDK Reference Guide for more information about available FIPS and non-FIPS algorithms.

Set up Luna EKM

Install Luna EKM on the host system. A Windows-based installation program setup is provided to assist with the EKM installation interactively.

> **NOTE:** Luna EKM setup available for download from Thales support portal.
>
> DOC ID for EKM 1.5 is KB0023244.
>
> DOC ID for EKM 1.4 is KB0016274.
>
> DOC ID for EKM 1.3 is KB0014957.
>
> **NOTE:** In Luna EKM 1.5, there is added support for crypto-user of the partition. Crypto-user can perform read-only operations such as encryption or decryption
>
> **NOTE:** Luna EKM 1.5 support the silent installation, execute the command below to install the Luna EKM silently on your host system:
>
> ```
> msiexec.exe /i LunaEKM.msi /qn INSTALLLEVEL=101 /l* EKM.txt
> ```

Luna EKM includes a command line configuration utility `LunaEKMConfig` that is used to register the Luna EKM. This command line utility is available in the Luna EKM installation folder. `LunaEKMconfig` provides command to register slots, view slots, and to configure log settings. Run the following commands, available in `LunaEKMConfig`.

1. Register the slot to Luna EKM.

   ```
   RegisterSlot
   ```

2. View List of the Slots/HSM configured with this client.

   ```
   ViewSlots
   ```

3. Configure log settings for Luna EKM.

   ```
   LogSettings
   LogLevel (NONE=0,INFO=1,DEBUG=2): <LogLevel>
   LogFile name: <Name and location of LogFile>
   ```

Set up SQL Server

Install SQL Server on the target machine to complete the integration process. If you are configuring a high availability (Always ON) SQL server group, the SQL Server must be installed on all nodes and all nodes must have access to WFCS. Refer to the *Microsoft SQL Server Documentation* for detailed installation procedures.

Integrating Luna HSM with SQL Server

This document contains detailed instructions and procedures to integrate Microsoft SQL Server with a Luna HSM or Luna Cloud HSM service. This integration contains the following topics:

> Enable EKM Provider option

> Create and register Luna EKM Provider

> Set up CREDENTIAL for Luna EKM Provider

> Use Luna EKM Provider Option

> Enable Transparent Database Encryption using Asymmetric key on Luna HSM

> Rotate Keys for Transparent Database Encryption

> Migrate TDE from SQL EKM to Luna EKM

> Use Extensible Key Management on a SQL Server Failover Cluster

Enable EKM Provider option

To enable the EKM Provider option:

1. Open the SQL Server Management Studio.
2. Connect to the SQL Server.
3. Open a new query window, and execute the following query:

   NOTE: The sp_configure command is supported on Enterprise, Developer, and Evaluation editions of SQL server. If you execute the command on an alternative version, you will receive an error.

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

```
GO
RECONFIGURE
GO
```

Create and register Luna EKM Provider

To create and register the Luna EKM Provider:

1. Open the SQL Server Management Studio.
2. Connect to the SQL Server.
3. Open a new query window, and execute the following command:

   ```
   CREATE CRYPTOGRAPHIC PROVIDER <Name of Cryptographic Provider>
   FROM FILE = '<Location of Luna EKM Provider Library>'
   ```

   Where `<Name of Cryptographic Provider>` can be any user defined unique name.
4. Verify the list of EKM providers:

   ```
   SELECT [provider_id]
   [name]
   ,[guid]
   ,[version]
   ,[dll_path]
   ,[is_enabled]
   FROM [model].[sys].[cryptographic_providers]
   ```
5. Verify the provider properties:

   ```
   SELECT [provider_id],[guid],[provider_version]
   ,[sqlcrypt_version]
   ,[friendly_name]
   ,[authentication_type]
   ,[symmetric_key_support]
   ,[symmetric_key_persistance]
   ,[symmetric_key_export]
   ,[symmetric_key_import]
   ,[asymmetric_key_support]
   ,[asymmetric_key_persistance]
   ,[asymmetric_key_export]
   ,[asymmetric_key_import]
   FROM [master].[sys].[dm_cryptographic_provider_properties]
   ```

Set up CREDENTIAL for Luna EKM Provider

The next step is to create a CREDENTIAL for the Luna EKM Provider. These CREDENTIAL must map to the SQL Service Account or logged in user to use the Luna EKM Provider.

NOTE: In Luna EKM 1.5, there is added support for crypto-user of the partition. The IDENTITY value must use the prefix "CU_" for crypto user. If no prefix is specified then crypto-officer role will be used by default for login to HSM partition.

To setup the CREDENTIAL for Luna EKM Provider

1. Open a query window, and execute the following command:

   ```
   CREATE CREDENTIAL <Name of credential>
   WITH IDENTITY='<Name of EKM User>', SECRET='<HSM partition password>'
   FOR CRYPTOGRAPHIC PROVIDER LunaEKMProvider
   ```

   Where `CREDENTIAL` and `IDENTITY` can be any user defined unique name

   And `IDENTITY` must use prefix `"CU_"` for Crypto User.

   > **NOTE:** You cannot create/delete/rotate the keys on HSM using Credential associated with Crypto User. Only crypto operations can be performed by login mapped with Crypto User Credential.

   Create credential for crypto-officer (CO).

   ```
   CREATE CREDENTIAL EKMCredential
   WITH IDENTITY='EKMUser', SECRET='userpin1'
   FOR CRYPTOGRAPHIC PROVIDER LunaEKMProvider
   ```

   Create credential for crypto-user (CU).

   ```
   CREATE CREDENTIAL EKMCredential
   WITH IDENTITY='CU_EKMUser', SECRET='userpin2'
   FOR CRYPTOGRAPHIC PROVIDER LunaEKMProvider
   ```

2. Map the Credential with SQL Service Account or Login:

   ```
   ALTER LOGIN [Domain\Login Name]
   ADD CREDENTIAL <Name of Credential created>
   ```

   > **NOTE:** The EKM session must be reopened in case the user changes the credentials or HSM service, or the client machine is deleted from the service, or the machine suffers a network disconnection.

Use Luna EKM Provider Option

The Luna EKM provider is now ready to use, it can be used to create/drop symmetric and asymmetric keys to/from the Luna partition and can perform encryption/decryption using these keys. The following types of symmetric key can be created on Luna HSM from the SQL Server:

> AES_128

> AES_192

> AES_256

Follow below steps to exercise the cryptographic capabilities of the Luna HSM from the SQL Server:

Create Symmetric Keys on Luna HSM

The following examples uses AES algorithms for the symmetric key operation. To test other algorithms, substitute `AES_256` with an alternate algorithm tag, as mentioned above.

To create the symmetric key using the Luna EKM Provider

1. Execute the following command from the SQL query window:

```
CREATE SYMMETRIC KEY SQL_EKM_AES_256_Key
FROM Provider LunaEKMProvider
WITH ALGORITHM = AES_256,
PROVIDER_KEY_NAME = 'EKM_AES_256_Key',
CREATION_DISPOSITION=CREATE_NEW
```

> **NOTE:** Once a key is created on the Luna HSM, it can be used or referred to by its name from the SQL Server. For example in the above test case, `SQL_EKM_ AES_256_Key` is the unique name of the key in the SQL Server. This key name can be used on the HSM for encrypt and decrypt operations.

To view symmetric key using the Luna EKM Provider

1. Execute the following command from the SQL query window:

```
SELECT * FROM [master].[sys].[symmetric_keys]
```

To encrypt a database table with symmetric keys using the Luna EKM Provider

1. Create a test Table in the MASTER database with fields.

```
Create Table test(

id numeric(10),

name varchar (50),

data varchar (max),)
```

2. Execute the following command from the SQL query window:

```
INSERT INTO dbo.test

values( 1,'some text',

EncryptByKey(Key_GUID('SQL_EKM_AES_256_Key'), 'text to be encrypted'))
```

To decrypt a database table with symmetric keys using the Luna EKM Provider

1. Execute the following command from the SQL query window:

```
SELECT id,name,CONVERT(varchar(MAX),

DecryptByKey(data))

FROM dbo.test where id =1
```

To drop symmetric keys using the Luna EKM Provider

1. Execute the following command from the SQL query window:

```
DROP SYMMETRIC KEY SQL_EKM_AES_256_Key REMOVE PROVIDER KEY
```

Create Asymmetric Keys on Luna HSM

The following types of asymmetric key can be created on Luna HSM from the SQL Server:

> RSA_2048
> RSA_3072
> RSA_4096

The following examples use `RSA_2048` algorithms for asymmetric key operation. To test other algorithms, substitute `RSA_2048` with an alternate algorithm tag, as mentioned above.

To create the asymmetric key using the Luna EKM Provider

1. Execute the following command from the SQL query window:

```
CREATE ASYMMETRIC KEY SQL_EKM_RSA_2048_Key
FROM Provider LunaEKMProvider
WITH ALGORITHM = RSA_2048,
PROVIDER_KEY_NAME = 'EKM_RSA_2048_Key',
CREATION_DISPOSITION=CREATE_NEW
```

NOTE: SQL Server does not implement FIPS 186-4. As a result, RSA key generation by SQL Server is not supported when HSM in FIPS Mode. The procedure to generate RSA Key when HSM in FIPS mode is given below.

When using HSM in FIPS mode, open the command prompt and generate the key using CMU utility provided with HSM Client:

```
cmu generatekeypair -label EKM_RSA_2048_Key -modulusBits=2048 -publicExp=65537
-sign=T -verify=T -encrypt=T -decrypt=T
```

Map the key in SQL Server:

```
CREATE ASYMMETRIC KEY SQL_EKM_RSA_2048_Key
FROM Provider LunaEKMProvider
WITH PROVIDER_KEY_NAME = 'EKM_RSA_2048_Key',
CREATION_DISPOSITION=OPEN_EXISTING
```

NOTE: Once a key is created on the Luna HSM, it can be used or referred by its name from the SQL Server, for example in the above test case, `SQL_EKM_ RSA_2048_Key` is the unique name of the key in the SQL Server. This key name can be used on the HSM for encrypt and decrypt operations.

To view asymmetric keys using the Luna EKM Provider

1. Execute the following command:

```
SELECT * FROM [master].[sys].[asymmetric_keys]
```

To encrypt a database table with asymmetric keys using the Luna EKM Provider

1. Create a test Table in the MASTER database with fields:

```
Create Table test(
id numeric(10),
name varchar (50),
data varchar (max),)
```

2. Execute the following command from the SQL query window:

```
INSERT INTO dbo.test
values ( 1,'some text',
EncryptByAsymKey (AsymKey_Id ('SQL_EKM_RSA_2048_Key'), 'text to be encrypted'))
```

To decrypt a database table with asymmetric keys using the Luna EKM Provider

1. Execute the following command from the SQL query window:

```
SELECT id, name, CONVERT (varchar (MAX),

DecryptByAsymKey (AsymKey_Id ('SQL_EKM_RSA_2048_Key'), data))

FROM dbo.test where id =1
```

To drop asymmetric keys using the Luna EKM Provider

1. Execute the following command from the SQL query window:

```
DROP ASYMMETRIC KEY SQL_EKM_RSA_2048_Key REMOVE PROVIDER KEY
```

Create Symmetric Key Encrypted by Asymmetric Key on Luna HSM

You can encrypt a symmetric keys using an asymmetric key. This increases the security of the symmetric key.

To create a symmetric key encrypted by an asymmetric key

1. Execute the following command from SQL query window:

```
Create SYMMETRIC KEY key1
WITH ALGORITHM = AES_256
ENCRYPTION BY Asymmetric Key SQL_EKM_RSA_2048_Key;
```

> **NOTE:** `SQL_EKM_RSA_2048_Key` is an existing asymmetric key on the Luna HSM. For more information about generating an asymmetric key, see Creating Asymmetric Keys on Luna HSM.

2. Before using the key, you need to open the key. Execute the following command to open the symmetric key:

```
OPEN SYMMETRIC KEY key1 DECRYPTION BY Asymmetric Key SQL_EKM_RSA_2048_Key;
```

> **NOTE:** For Microsoft SQL Server 2017, apply the patch as described in the Troubleshooting Problem – 3.

3. Create a test Table in the MASTER database with fields:

```
Create Table test(
id numeric(10),
name varchar (50),
data varchar (max),)
```

4. Encrypt the table data using the symmetric key.

```
INSERT INTO dbo.test
values ( 1,'some text',
Encryptbykey(KEY_GUID('key1'),'text to be encrypted'))
```

5. Decrypt the data using the symmetric key.

```
SELECT id,name,CONVERT(varchar(MAX),
DecryptByKey(data))
FROM dbo.test where id =1
```

6. Close the symmetric key.

```
CLOSE SYMMETRIC KEY key1
```

Enable Transparent Database Encryption using Asymmetric key on Luna HSM

You can enable Transparent Data Encryption (TDE) using an asymmetric key stored on a Luna HSM.

> **NOTE:** We have included support for creating higher length asymmetric keys: RSA_3072 and RSA_4096 from Luna EKM v1.3 onwards. However during our integration testing, we identified an issue in TDE when encrypting the DEK using RSA_4096 key. This issue has been reported to Microsoft technical support and we are awaiting a resolution. At this time, we recommend to use a maximum key length of RSA_3072 for the TDE. We will retest and update the integration guide when Microsoft resolves this issue.
>
> **NOTE:** Database encryption operations cannot be executed on 'master', 'model', 'tempdb', 'msdb', or 'resource' databases.

To enable TDE using asymmetric key on Luna HSM

1. Create an asymmetric key using Luna EKM Provider.

```
Use master;
CREATE ASYMMETRIC KEY SQL_EKM_RSA_2048_Key_TDE
FROM Provider LunaEKMProvider
WITH ALGORITHM = RSA_2048,
PROVIDER_KEY_NAME = 'EKM_RSA_2048_Key_TDE',
CREATION_DISPOSITION=CREATE_NEW
```

> **NOTE:** SQL Server does not implement FIPS 186-4. As a result, RSA key generation by SQL Server is not supported when HSM in FIPS Mode. The procedure to generate RSA Key when HSM in FIPS mode is given below.

When using HSM in FIPS mode, open the command prompt and generate the key using CMU utility provided with HSM Client:

```
cmu generatekeypair -label EKM_RSA_2048_Key_TDE -modulusBits=2048 -
publicExp=65537 -sign=T -verify=T -encrypt=T -decrypt=T
```

Map the key in SQL Server:

```
CREATE ASYMMETRIC KEY SQL_EKM_RSA_2048_Key_TDE
FROM Provider LunaEKMProvider
WITH PROVIDER_KEY_NAME = 'EKM_RSA_2048_Key_TDE',
CREATION_DISPOSITION=OPEN_EXISTING
```

2. Create a CREDENTIAL for Luna EKM Provider.

```
CREATE CREDENTIAL <Name of credential>
WITH IDENTITY='<Name of EKM User>', SECRET='<HSM partition password>'
FOR CRYPTOGRAPHIC PROVIDER LunaEKMProvider
```

3. Create a login based on the recently created asymmetric key.

```
CREATE LOGIN <Name of login>
FROM ASYMMETRIC KEY SQL_EKM_RSA_2048_Key_TDE;
```

4. Map the credential created above to the recently created login.

```
ALTER LOGIN <Name of Login>
ADD CREDENTIAL <Name of credential>;
```

5. Create a Database Encryption Key.

```
CREATE DATABASE TDE;
Use tde;
CREATE DATABASE ENCRYPTION KEY
WITH ALGORITHM = AES_256
ENCRYPTION BY SERVER ASYMMETRIC KEY SQL_EKM_RSA_2048_Key_TDE;
```

6. Enable Transparent Database Encryption.

```
ALTER DATABASE TDE
SET ENCRYPTION ON;
```

7. Query the status of database encryption and the completion percentage.

```
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.asymmetric_keys AS c
ON e.encryptor_thumbprint = c.thumbprint
```

Rotate Keys for Transparent Database Encryption

Microsoft recommends updating your TDE security keys regularly by rotating the available symmetric and asymmetric encryption keys. Execute the following command to rotate keys for TDE:

1. Generate an asymmetric key using the Luna EKM Provider.

```
Use master;
CREATE ASYMMETRIC KEY SQL_EKM_RSA_2048_Key_TDE_Rot
FROM Provider LunaEKMProvider
WITH ALGORITHM = RSA_2048,
PROVIDER_KEY_NAME = 'EKM_RSA_2048_Key_TDE_Rot',
CREATION_DISPOSITION=CREATE_NEW
```

> **NOTE:** SQL Server does not implement FIPS 186-4. As a result, RSA key generation by SQL Server is not supported when HSM in FIPS Mode. The procedure to generate RSA Key when HSM in FIPS mode is given below.

When using HSM in FIPS mode, open the command prompt and generate the key using CMU utility provided with HSM Client and then map the key in SQL Server:

```
cmu generatekeypair -label EKM_RSA_2048_Key_TDE_Rot -modulusBits=2048 -
publicExp=65537 -sign=T -verify=T -encrypt=T -decrypt=T
```

Map the key in SQL Server:

```
CREATE ASYMMETRIC KEY SQL_EKM_RSA_2048_Key_TDE_Rot
FROM Provider LunaEKMProvider
WITH PROVIDER_KEY_NAME = 'EKM_RSA_2048_Key_TDE_Rot',
CREATION_DISPOSITION=OPEN_EXISTING
```

2. Create a CREDENTIAL for Luna EKM Provider.

```
CREATE CREDENTIAL <Name of credential>
WITH IDENTITY='<Name of EKM User>', SECRET='<HSM partition password>'
FOR CRYPTOGRAPHIC PROVIDER LunaEKMProvider
```

3. Create a login based on the recently created asymmetric key.

```
CREATE LOGIN <Name of login>
FROM ASYMMETRIC KEY SQL_EKM_RSA_2048_Key_TDE_Rot;
```

4. Map the credential to the recently created login.

```
ALTER LOGIN <Name of Login>
ADD CREDENTIAL <Name of credential>;
```

5. Enable Transparent Database Encryption Key Rotation.

   To rotate the database encryption key

```
Use tde;
ALTER DATABASE ENCRYPTION KEY
REGENERATE
WITH ALGORITHM = AES_128
```

   To rotate the asymmetric key used to encrypt the DEK.

```
ALTER DATABASE ENCRYPTION KEY
ENCRYPTION BY SERVER ASYMMETRIC KEY SQL_EKM_RSA_2048_Key_TDE_Rot;
go
SELECT * FROM sys.dm_database_encryption_keys
go
```

6. Execute the following command to query the status of database encryption, the status of TDE key change, and the tablespace encryptions percentage of completion.

```
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.asymmetric_keys AS c
ON e.encryptor_thumbprint = c.thumbprint
```

Migrate TDE from SQL EKM to Luna EKM

Previously, the database master key was generated in SQL and encrypted using a certificate or asymmetric key. Now you can do the following to migrate to Luna EKM.

> Rotate the database encryption key (DEK) and migrate to key encryption key (KEK) generated on a Luna HSM.

> Migrate to key encryption key (KEK) generated on a Luna HSM without rotating database encryption key (DEK).

The former will decrypt the database using existing DEK and then re-encrypt the database with new DEK while later will only encrypt the DEK with KEK created on Luna HSM without decrypting and re-encrypting the whole database.

This example uses the database name <AdventureWorks>. To migrate TDE from SQL EKM to Luna EKM:

1. Create an asymmetric key.

   ```
   Use master;
   CREATE ASYMMETRIC KEY SQL_EKM_RSA_2048_Key_AW
   FROM Provider LunaEKMProvider
   WITH ALGORITHM = RSA_2048,
   PROVIDER_KEY_NAME = 'EKM_RSA_2048_Key_AW',
   CREATION_DISPOSITION=CREATE_NEW
   ```

   NOTE: SQL Server does not implement FIPS 186-4. As a result, RSA key generation by SQL Server is not supported when HSM in FIPS Mode. The procedure to generate RSA Key when HSM in FIPS mode is given below.

   When using HSM in FIPS mode, open the command prompt and generate the key using CMU utility provided with HSM Client and then map the key in SQL Server:

   ```
   cmu generatekeypair -label EKM_RSA_2048_Key_AW -modulusBits=2048 -
   publicExp=65537 -sign=T -verify=T -encrypt=T -decrypt=T
   ```

   Map the key in SQL Server:

   ```
   CREATE ASYMMETRIC KEY SQL_EKM_RSA_2048_Key_AW
   FROM Provider LunaEKMProvider
   WITH PROVIDER_KEY_NAME = 'EKM_RSA_2048_Key_AW',
   CREATION_DISPOSITION=OPEN_EXISTING
   ```

2. Create a CREDENTIAL for Luna EKM Provider.

   ```
   CREATE CREDENTIAL <Name of credential>
   WITH IDENTITY='<Name of EKM User>', SECRET='<HSM partition password>'
   FOR CRYPTOGRAPHIC PROVIDER LunaEKMProvider
   ```

3. Create a login based on the recently created asymmetric key.

   ```
   CREATE LOGIN <Name of login>
   FROM ASYMMETRIC KEY SQL_EKM_RSA_2048_Key_AW;
   ```

4. Map the CREDENTIAL to the recently created login.

   ```
   ALTER LOGIN <Name of Login>
   ADD CREDENTIAL <Name of credential>;
   ```

5. Migrate Transparent Database Encryption from SQL to Luna EKM.

To rotate the Database Encryption Key (DEK) and migrate to KEK created on Luna HSM.

i. Back up the database and transaction logs. When the backup completes, restart the SQL database.

ii. Rotate the DEK

```
USE AdventureWorks;

ALTER DATABASE ENCRYPTION KEY

REGENERATE

WITH ALGORITHM = AES_256
```

iii. Migrate to KEK.

```
USE AdventureWorks;

ALTER DATABASE ENCRYPTION KEY

ENCRYPTION BY SERVER ASYMMETRIC KEY SQL_EKM_RSA_2048_Key_AW

Go

SELECT * FROM sys.dm_database_encryption_keys

Go
```

To migrate to KEK created on Luna HSM without rotating the Database Encryption Key (DEK)

i. Back up the database and transaction logs. When the backup completes, restart the SQL database.

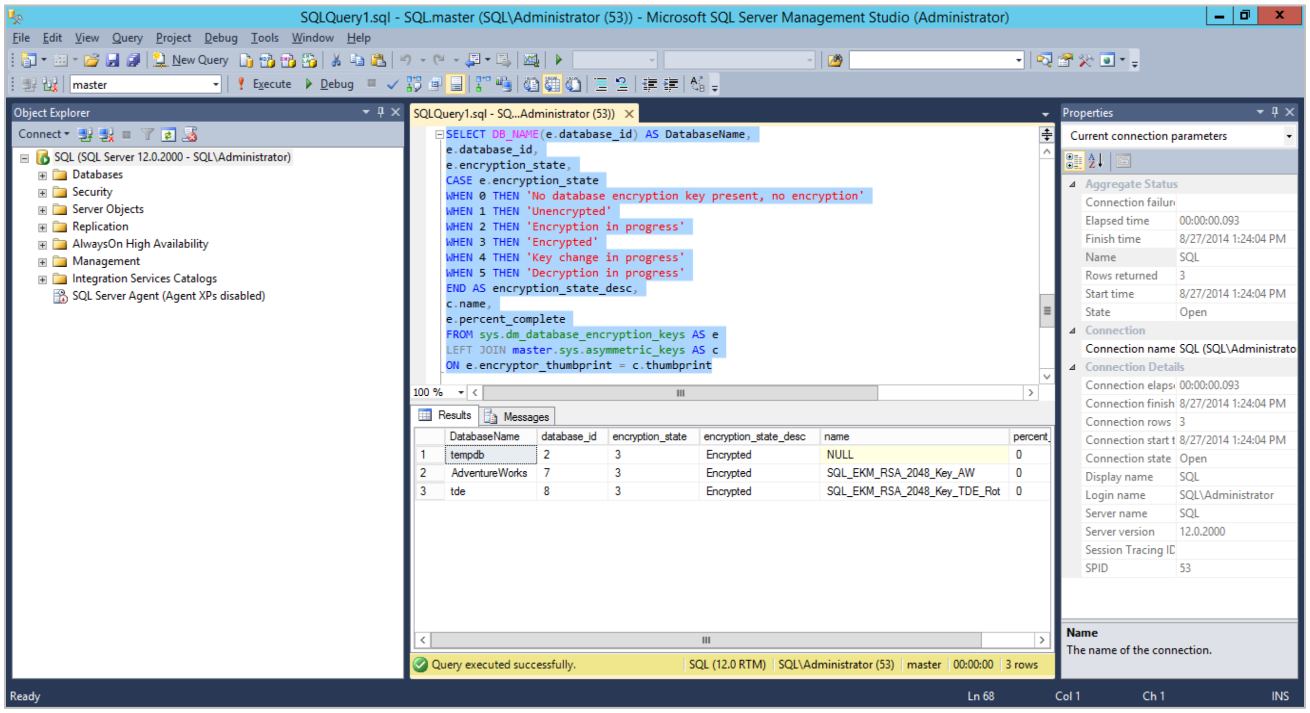
ii. Migrate to KEK.

```
USE AdventureWorks;

ALTER DATABASE ENCRYPTION KEY

ENCRYPTION BY SERVER ASYMMETRIC KEY SQL_EKM_RSA_2048_Key_AW

Go

SELECT * FROM sys.dm_database_encryption_keys

Go
```

6. Check the status of Transparent Database Encryption after migration.

```
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.asymmetric_keys AS c
ON e.encryptor_thumbprint = c.thumbprint
```

If the Query is executed successfully, status of database encryption and its completion percentage will display.


SQLQuery1.sql - SQL.master (SQL\Administrator (53)) - Microsoft SQL Server Management Studio (Administrator)
File Edit View Query Project Debug Tools Window Help
New Query
master
Execute
Debug
Object Explorer
Connect
SQL (SQL Server 12.0.2000 - SQL\Administrator)
Databases
Security
Server Objects
Replication
AlwaysOn High Availability
Management
Integration Services Catalogs
SQL Server Agent (Agent XPs disabled)
SQLQuery1.sql - SQ...Administrator (53))
100 %
Results
Messages
Properties
Ready
Ln 68
Col 1
Ch 1
INS

```
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.asymmetric_keys AS c
ON e.encryptor_thumbprint = c.thumbprint
```

	DatabaseName	database_id	encryption_state	encryption_state_desc	name	percent_
1	tempdb	2	3	Encrypted	NULL	0
2	AdventureWorks	7	3	Encrypted	SQL_EKM_RSA_2048_Key_AW	0
3	tde	8	3	Encrypted	SQL_EKM_RSA_2048_Key_TDE_Rot	0

Query executed successfully. | SQL (12.0 RTM) | SQL\Administrator (53) | master | 00:00:00 | 3 rows

Current connection parameters

Aggregate Status	
Connection failur	
Elapsed time	00:00:00.093
Finish time	8/27/2014 1:24:04 PM
Name	SQL
Rows returned	3
Start time	8/27/2014 1:24:04 PM
State	Open
Connection	
Connection name	SQL (SQL\Administrato
Connection Details	
Connection elaps	00:00:00.093
Connection finish	8/27/2014 1:24:04 PM
Connection rows	3
Connection start t	8/27/2014 1:24:04 PM
Connection state	Open
Display name	SQL
Login name	SQL\Administrator
Server name	SQL
Server version	12.0.2000
Session Tracing IE	
SPID	53

Name
The name of the connection.

The migration from SQL server to Luna HSM or Luna Cloud HSM service is completed.

Use Extensible Key Management on a SQL Server Failover Cluster

This section focuses on the preparation of the environment for a 2-node SQL Server Cluster in Windows Server.

1. Refer to the SQL Server documentation to install a failover cluster.

   To set up a shared storage disk for SQL Server Cluster, refer to the configuration procedures that apply for shared storage solution. Plan the size of the shared storage, depending on the number of certificates that are required to be enrolled.
2. Once the cluster is up and running, install the Luna Network HSM client or Luna Cloud HSM service client on both the nodes.
3. Configure and set up the HSM on both the nodes and register the same partition or Cloud HSM service on both nodes in the SQL Server Cluster.
4. Install the Luna EKM client on both the nodes.
5. Configure the Luna EKM provider on both the nodes.
6. Open the SQL Server management studio to register the Luna EKM provider on the first node.
7. Set up the credential on the first node.
8. Create encryption keys using the Luna EKM provider on the first node.
9. Create a table and encrypt a column with the Luna EKM key on the first node.
10. Shut down the first node.
11. Log in to the second node and decrypt the data encrypted on the first node.
12. If the data decrypts successfully, Extensible Key Management (EKM) using Luna EKM is operating correctly on the SQL Server cluster.

Integrating Luna HSM with SQL Server High Availability Group

To integrate Luna HSM with SQL Server, set up and configure the Luna EKM Provider and enable the EKM provider in the SQL server. The EKM feature is available on the Enterprise, Developer, and Evaluation editions of the SQL server. EKM is disabled by default. You can set up SQL server in a High Availability configuration for failover support. Luna Client and Luna EKM must be set up on all SQL Server cluster nodes to be added to the “Always On” availability group. All nodes must be registered with the same partition of Luna HSM or the same service client on Cloud HSM service. This integration involves the following steps:

> Enable EKM Provider Option

> Create and Register Luna EKM Provider

> Set up CREDENTIAL for Luna EKM Provider

> Create Always On Availability Group

> Create Encryption Keys for Availability Group Database

> Enable Transparent Database Encryption using Asymmetric key on Luna HSM

> Add Encrypted Database to Availability Group

> Rotate Keys for Transparent Database Encryption

Enable EKM Provider Option

Use the `sp_configure` command to enable the EKM Provider option on all nodes in the high availability configuration. To enable the Extensible Key Management option:

1. Open the SQL Server Management Studio.
2. Connect to the SQL Server.
3. Open a query window, and execute the following:

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

> **NOTE:** The `sp_configure` command is supported on Enterprise, Developer, and Evaluation editions of SQL server. If you execute the command on an alternative version, you will receive an error.

Create and Register Luna EKM Provider

Set up the Luna EKM provider. Install the Luna EKM Software and register it for use with SQL Server on all nodes in the high availability configuration. To create and register the Luna EKM Provider:

1. Open the SQL Server Management Studio.
2. Connect to the SQL Server.
3. Open a query window, and execute the following:

```
CREATE CRYPTOGRAPHIC PROVIDER <Name of Cryptographic Provider>
FROM FILE = '<Location of Luna EKM Provider Library>'
```

   where `CRYPTOGRAPHIC PROVIDER` can be any user defined unique name.

4. To view the list of EKM providers:

```
SELECT [provider_id]
,[name]
,[guid]
,[version]
,[dll_path]
,[is_enabled]
FROM [model].[sys].[cryptographic_providers]
```

5. View the provider properties:

```
SELECT [provider_id],[guid],[provider_version]
,[sqlcrypt_version]
```

```
,[friendly_name]
,[authentication_type]
,[symmetric_key_support]
,[symmetric_key_persistance]
,[symmetric_key_export]
,[symmetric_key_import]
,[asymmetric_key_support]
,[asymmetric_key_persistance]
,[asymmetric_key_export]
,[asymmetric_key_import]
FROM [master].[sys].[dm_cryptographic_provider_properties]
```

Set up CREDENTIAL for Luna EKM Provider

Create a CREDENTIAL for the Luna EKM Provider and map the CREDENTIAL to the SQL Service Account or Login to use the Luna EKM Provider on all nodes in the High Availability configuration. To setup the CREDENTIAL for Luna EKM Provider:

1. Open a query window, and execute the following command:

   ```
   CREATE CREDENTIAL <Name of credential>
   WITH IDENTITY='<Name of EKM User>', SECRET='<HSM partition password>'
   FOR CRYPTOGRAPHIC PROVIDER LunaEKMProvider
   ```

   Where `CREDENTIAL` and `IDENTITY` can be any user defined unique name.

2. Map the Credential to the SQL Service Account or Login:

   ```
   ALTER LOGIN [Domain\Login Name]
   ADD CREDENTIAL <Name of Credential created>
   ```

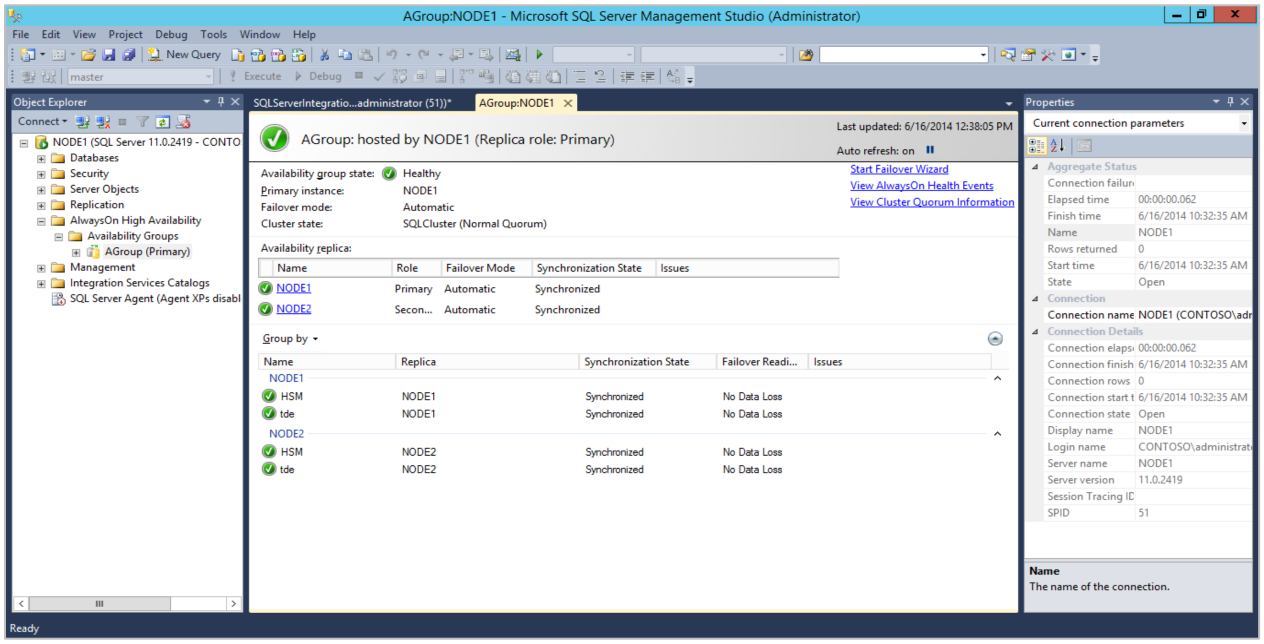
   > **NOTE:** We recommend using a domain user on all SQL Server nodes. The EKM session must be reopened if the user changes the HSM service, the client machine is deleted from the service, or the machine suffers a network disconnection.

Create Always On Availability Group

Create the Always On Availability group and configure the nodes in the cluster to communicate with each other. For detailed installation procedures, refer to the Microsoft Documentation for creating the Always on Availability group. To create the Always On Availability Group:

1. Open the Microsoft SQL Server management Studio on the primary node.
2. Create a database.
3. Back up the database to a shared network location that is accessible by all of the SQL Server nodes.

4. Open the Always On Availability Group Creation wizard and create an Always On Availability group for the cluster configuration. After the successful creation of the group, the dashboard displays all the participating nodes. An example of a dashboard is shown below. For demonstration purposes, two nodes were added: primary and secondary.


AGroup:NODE1 - Microsoft SQL Server Management Studio (Administrator)
File Edit View Project Debug Tools Window Help
New Query
Execute
Debug
master
Object Explorer
Connect
SQLServerIntegratio...administrator (51))*
AGroup:NODE1
Properties
Ready

Object Explorer:

- NODE1 (SQL Server 11.0.2419 - CONTO
  - Databases
  - Security
  - Server Objects
  - Replication
  - AlwaysOn High Availability
    - Availability Groups
      - AGroup (Primary)
  - Management
  - Integration Services Catalogs
  - SQL Server Agent (Agent XPs disabl

AGroup: hosted by NODE1 (Replica role: Primary)

Last updated: 6/16/2014 12:38:05 PM
Auto refresh: on

Availability group state:	Healthy
Primary instance:	NODE1
Failover mode:	Automatic
Cluster state:	SQLCluster (Normal Quorum)

Start Failover Wizard
View AlwaysOn Health Events
View Cluster Quorum Information

Availability replica:

Name	Role	Failover Mode	Synchronization State	Issues
NODE1	Primary	Automatic	Synchronized	
NODE2	Secon...	Automatic	Synchronized	

Group by

Name	Replica	Synchronization State	Failover Readi...	Issues
NODE1				
HSM	NODE1	Synchronized	No Data Loss	
tde	NODE1	Synchronized	No Data Loss	
NODE2				
HSM	NODE2	Synchronized	No Data Loss	
tde	NODE2	Synchronized	No Data Loss	

Properties: Current connection parameters

Aggregate Status	
Connection failur	
Elapsed time	00:00:00.062
Finish time	6/16/2014 10:32:35 AM
Name	NODE1
Rows returned	0
Start time	6/16/2014 10:32:35 AM
State	Open
Connection	
Connection name	NODE1 (CONTOSO\adr
Connection Details	
Connection elaps	00:00:00.062
Connection finish	6/16/2014 10:32:35 AM
Connection rows	0
Connection start t	6/16/2014 10:32:35 AM
Connection state	Open
Display name	NODE1
Login name	CONTOSO\administrat
Server name	NODE1
Server version	11.0.2419
Session Tracing IC	
SPID	51

Name
The name of the connection.

Create Encryption Keys for Availability Group Database

You can use the Luna EKM provider to create/drop symmetric and asymmetric keys to/from the HSM and can perform encryption/decryption using these keys.

To create the symmetric key using the Luna EKM Provider

1. Open the SSMS on the primary node.
2. Execute the following command from the SQL query window:

```
USE HSMDB;
```

3. Execute the following command from the SQL query window:

```
CREATE SYMMETRIC KEY SQL_EKM_AES_256_Key
FROM Provider LunaEKMProvider
WITH ALGORITHM = AES_256,
PROVIDER_KEY_NAME = 'EKM_AES_256_Key',
CREATION_DISPOSITION=CREATE_NEW
```

NOTE: Once a key is created on the Luna HSM, it can be used or referred by its name from the SQL Server. For example in the above said test case, `SQL_EKM_ AES_256_Key` is the unique name of the key in the SQL Server. Using this key name will use the key on the HSM for encrypt and decrypt operations.

To view symmetric keys using the Luna EKM Provider

1. Execute the following command from the SQL query window:

   ```
   SELECT * FROM [hsmdb].[sys].[symmetric_keys]
   ```

To encrypt a database table with symmetric keys using the Luna EKM Provider

1. Create a test Table in the HSMDB database with fields.

   ```
   Create Table test(
   id numeric(10),
   name varchar (50),
   data varchar (max),)
   ```

2. Execute the following command from the SQL query window:

   ```
   INSERT INTO dbo.test
   values( 1,'some text',
   EncryptByKey(Key_GUID('SQL_EKM_AES_256_Key'), 'text to be encrypted'))
   ```

To decrypt a database table with symmetric keys using the Lune EKM Provider

1. Execute the following command from the SQL query window:

   ```
   SELECT id,name,CONVERT(varchar(MAX),
   DecryptByKey(data))
    FROM dbo.test where id =1
   ```

2. Now execute the above command on secondary replica and verify that the output is same as primary replica.

Create Asymmetric Keys on Luna HSM

To create the asymmetric key using the Lune EKM Provider:

1. Execute the following command from the SQL query window:

   ```
   CREATE ASYMMETRIC KEY SQL_EKM_RSA_2048_Key
   FROM Provider LunaEKMProvider
   WITH ALGORITHM = RSA_2048,
   PROVIDER_KEY_NAME = 'EKM_RSA_2048_Key',
   CREATION_DISPOSITION=CREATE_NEW
   ```

   > **NOTE:** SQL Server does not implement FIPS 186-4. As a result, RSA key generation by SQL Server is not supported when HSM in FIPS Mode. The procedure to generate RSA Key when HSM in FIPS mode is given below.

   When using HSM in FIPS mode, open the command prompt and generate the key using CMU utility provided with HSM Client:

   ```
   cmu generatekeypair -label EKM_RSA_2048_Key -modulusBits=2048 -publicExp=65537
   -sign=T -verify=T -encrypt=T -decrypt=T
   ```

   Map the key in SQL Server:

   ```
   CREATE ASYMMETRIC KEY SQL_EKM_RSA_2048_Key
   ```

```
FROM Provider LunaEKMProvider
WITH PROVIDER_KEY_NAME = 'EKM_RSA_2048_Key',
CREATION_DISPOSITION=OPEN_EXISTING
```

NOTE: Once a key is created on the Luna HSM, it can be used or referred by its name from the SQL Server, for example in the above test case, `SQL_EKM_ RSA_2048_Key` is the unique name of the key in the SQL Server. This key name can be used on the HSM for encrypt and decrypt operations.

To view asymmetric keys using the Luna EKM Provider

1. Execute the following command from the SQL query window:

```
SELECT * FROM [hsmdb].[sys].[asymmetric_keys]
```

To encrypt a database table with asymmetric keys using the Luna EKM Provider

1. Execute the following command from the SQL query window:

```
INSERT INTO dbo.test
values ( 2,'some text',
EncryptByAsymKey (AsymKey_Id ('SQL_EKM_RSA_2048_Key'), 'text to be encrypted'))
```

To decrypt a database table with asymmetric keys using the Luna EKM Provider

1. Execute the following command from the SQL query window:

```
SELECT id, name, CONVERT (varchar (MAX),

DecryptByAsymKey (AsymKey_Id ('SQL_EKM_RSA_2048_Key'), data))

FROM dbo.test where id =2
```

2. Now execute the above command on secondary replica and verify that the output is same as primary replica.

Create Symmetric Key Encrypted by Asymmetric Key on Luna HSM

You can encrypt the symmetric keys using an asymmetric key. This increases the security of the symmetric key. To create a symmetric key encrypted by an asymmetric key:

1. Open the SSMS on the primary node.
2. Execute the following command from SQL query window:

```
Create SYMMETRIC KEY key1
WITH ALGORITHM = AES_256
ENCRYPTION BY Asymmetric Key SQL_EKM_RSA_2048_Key;
```

   Where `"SQL_EKM_RSA_2048_Key"` is an existing asymmetric key.

3. Before using the key you need to open the key. Execute the following command to open the symmetric key:

```
OPEN SYMMETRIC KEY key1 DECRYPTION BY Asymmetric Key SQL_EKM_RSA_2048_Key;
```

NOTE: For Microsoft SQL Server 2017, apply the patch as described in the Troubleshooting Problem – 3

4. Encrypt the data using the key1.

   ```
   INSERT INTO dbo.test
   values ( 3,'some text',
   Encryptbykey(KEY_GUID('Key1'), 'text to be encrypted'))
   ```

5. Decrypt the data using the key1.

   ```
   SELECT id,name,CONVERT(varchar(MAX),
   DecryptByKey(data))
   FROM dbo.test where id =3
   ```

6. Close the symmetric key.

   ```
   CLOSE SYMMETRIC KEY key1
   ```

7. Now execute the above steps (3-6) on secondary replica and verify that the output is the same as primary replica.

Enable Transparent Database Encryption using Asymmetric key on Luna HSM

NOTE: We have included support for creating higher length asymmetric keys: RSA_3072 and RSA_4096 from Luna EKM v1.3 onwards. However during our integration testing, we identified an issue in TDE when encrypting the DEK using RSA_4096 key. This issue has been reported to Microsoft technical support and we are awaiting a resolution. At this time, we recommend to use a maximum key length of RSA_3072 for the TDE. We will retest and update the integration guide when Microsoft resolves this issue.

NOTE: Database encryption operations cannot be executed on 'master', 'model', 'tempdb', 'msdb', or 'resource' databases.

You can enable Transparent Data Encryption (TDE) using an asymmetric key stored on a Luna HSM. To enable TDE using asymmetric key on Luna HSM:

1. Create an asymmetric key using Luna EKM Provider on primary replica.

   ```
   Use master;
   CREATE ASYMMETRIC KEY SQL_EKM_RSA_2048_Key_TDE
   FROM Provider LunaEKMProvider
   WITH ALGORITHM = RSA_2048,
   PROVIDER_KEY_NAME = 'EKM_RSA_2048_Key_TDE',
   CREATION_DISPOSITION=CREATE_NEW
   ```

   NOTE: SQL Server does not implement FIPS 186-4. As a result, RSA key generation by SQL Server is not supported when HSM in FIPS Mode. The procedure to generate RSA Key when HSM in FIPS mode is given below.

   When using HSM in FIPS mode, open the command prompt and generate the key using CMU utility provided with HSM Client and then map the key in SQL Server:

   ```
   cmu generatekeypair -label EKM_RSA_2048_Key_TDE -modulusBits=2048 -
   publicExp=65537 -sign=T -verify=T -encrypt=T -decrypt=T
   ```

Map the key in SQL Server

```
CREATE ASYMMETRIC KEY SQL_EKM_RSA_2048_Key_TDE
FROM Provider LunaEKMProvider
WITH PROVIDER_KEY_NAME = 'EKM_RSA_2048_Key_TDE',
CREATION_DISPOSITION=OPEN_EXISTING
```

2. Create the same asymmetric key using Luna EKM Provider on secondary replica.

```
Use master;
CREATE ASYMMETRIC KEY SQL_EKM_RSA_2048_Key_TDE
FROM Provider LunaEKMProvider
WITH PROVIDER_KEY_NAME = 'EKM_RSA_2048_Key_TDE',
CREATION_DISPOSITION=OPEN_EXISTING
```

3. Create a CREDENTIAL for Luna EKM Provider.

```
CREATE CREDENTIAL <Name of credential>
WITH IDENTITY='<Name of EKM User>', SECRET='<HSM partition password>'
FOR CRYPTOGRAPHIC PROVIDER LunaEKMProvider
```

4. Create a login based on the recently created asymmetric key.

```
CREATE LOGIN <Name of login>
FROM ASYMMETRIC KEY SQL_EKM_RSA_2048_Key_TDE;
```

5. Map the CREDENTIAL to the recently created Login.

```
ALTER LOGIN <Name of Login>
ADD CREDENTIAL <Name of credential>;
```

6. Execute the above steps (2-5) for all secondary nodes.

> **NOTE:** Repeating the procedure is required for all nodes in the database because the TDE encryption key, CREDENTIAL, and Login, are objects in the master database and are not replicated by including the node in the **Availability Groups**.

7. Create a Database Encryption Key on the primary node.

```
CREATE DATABASE TDE;
Use tde;
CREATE DATABASE ENCRYPTION KEY
WITH ALGORITHM = AES_256
ENCRYPTION BY SERVER ASYMMETRIC KEY SQL_EKM_RSA_2048_Key_TDE;
```

8. Enable Transparent Database Encryption:

```
ALTER DATABASE TDE
SET ENCRYPTION ON;
```

9. Query the status of database encryption and its completion percentage.

```
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.asymmetric_keys AS c
ON e.encryptor_thumbprint = c.thumbprint
```

Add Encrypted Database to Availability Group

Before adding the already encrypted database into the availability group, back up the encrypted database to a network location that is accessible by all secondary nodes. To add the encrypted database to the availability group:

1. Open the SMS on the primary node.
2. Add the database (e.g. TDE) into the availability group (e.g. AGroup).

   ```
   use master;
   ALTER AVAILABILITY GROUP AGroup ADD DATABASE tde;
   GO
   ```

   This command adds the database to the Availability Group, but it is not yet available on the secondary node. To access the encrypted database from the secondary node you need to synchronize the databases by restoring the database on the second node.
3. Restore the database on the secondary node. Restore the database from the location where you have stored the encrypted database with the “RESTORE WITH NORECOVERY” parameter.
4. Add the database on the secondary node using the following SQL command:

   ```
   use master;
   ALTER DATABASE tde SET HADR AVAILABILITY GROUP = AGroup;
   ```
5. Query the status of database encryption and its completion percentage on the secondary node.

   ```
   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.asymmetric_keys AS c
   ```

```
ON e.encryptor_thumbprint = c.thumbprint
```

Rotate Keys for Transparent Database Encryption

We recommend updating your TDE security keys regularly by rotating the available symmetric and asymmetric encryption keys. To rotate keys for TDE:

1. Create an asymmetric key using the Luna EKM Provider on the primary node.

   ```
   Use master;
   CREATE ASYMMETRIC KEY SQL_EKM_RSA_2048_Key_TDE_Rot
   FROM Provider LunaEKMProvider
   WITH ALGORITHM = RSA_2048,
   PROVIDER_KEY_NAME = 'EKM_RSA_2048_Key_TDE_Rot',
   CREATION_DISPOSITION=CREATE_NEW
   ```

   > **NOTE:** SQL Server does not implement FIPS 186-4. As a result, RSA key generation by SQL Server is not supported when HSM in FIPS Mode. The procedure to generate RSA Key when HSM in FIPS mode is given below.

   When using HSM in FIPS mode, open the command prompt and generate the key using CMU utility provided with HSM Client and then map the key in SQL Server:

   ```
   cmu generatekeypair -label EKM_RSA_2048_Key_TDE_Rot -modulusBits=2048 -
   publicExp=65537 -sign=T -verify=T -encrypt=T -decrypt=T
   ```

   Map the key in SQL Server:

   ```
   CREATE ASYMMETRIC KEY SQL_EKM_RSA_2048_Key_TDE_Rot
   FROM Provider LunaEKMProvider
   WITH PROVIDER_KEY_NAME = 'EKM_RSA_2048_Key_TDE_Rot',
   CREATION_DISPOSITION=OPEN_EXISTING
   ```

2. Create the same asymmetric key using the Luna EKM Provider on a secondary node.

   ```
   Use master;
   CREATE ASYMMETRIC KEY SQL_EKM_RSA_2048_Key_TDE_Rot
   FROM Provider LunaEKMProvider
   WITH PROVIDER_KEY_NAME = 'EKM_RSA_2048_Key_TDE_Rot',
   CREATION_DISPOSITION=OPEN_EXISTING
   ```

3. Create a CREDENTIAL for Luna EKM Provider.

   ```
   CREATE CREDENTIAL <Name of credential>
   WITH IDENTITY='<Name of EKM User>', SECRET='<HSM partition password>'
   FOR CRYPTOGRAPHIC PROVIDER LunaEKMProvider
   ```

4. Create a login based on the recently created asymmetric key.

   ```
   CREATE LOGIN <Name of login>
   FROM ASYMMETRIC KEY SQL_EKM_RSA_2048_Key_TDE_Rot;
   ```

5. Map the CREDENTIAL to the recently created Login.

   ```
   ALTER LOGIN <Name of Login>
   ADD CREDENTIAL <Name of credential>;
   ```

6. Execute steps 2-5 for all secondary nodes.

> **NOTE:** Repeating the procedure is required for all nodes in the database because the TDE encryption key, CREDENTIAL, and Login, are objects in the master database and are not replicated by including the node in the Availability Groups.

7. Enable TDE Key Rotation on the primary replica.

   ```
   Use tde;
   ALTER DATABASE ENCRYPTION KEY
   REGENERATE
   WITH ALGORITHM = AES_128

   ALTER DATABASE ENCRYPTION KEY
   ENCRYPTION BY SERVER ASYMMETRIC KEY SQL_EKM_RSA_2048_Key_TDE_Rot;
   go
   SELECT * FROM sys.dm_database_encryption_keys
   go
   ```

8. Query the status of database encryption, TDE key change and its completion percentage.

   ```
   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.asymmetric_keys AS c
   ON e.encryptor_thumbprint = c.thumbprint
   ```

This completes the integration of Microsoft SQL Server High Availability with a Luna Network HSM or Luna Cloud HSM service.

Integrating Luna HSM with SQL Server Always Encrypted

This section contains detailed instructions to integrate Microsoft SQL Server Always Encrypted with a Luna HSM. It contains the following topics:

Configure SafeNet KSP

Register the SafeNet Key Storage Provider (KSP) on the target machine to generate the column master key and encryption key on the Luna HSM. To configure the SafeNet KSP:

1. Navigate to the **32-bit SafeNet KSP** directory.

   `<Luna Client Installation Directory>\win32\KSP`

   For Example: `cd "C:\Program Files\SafeNet\LunaClient\win32\KSP"`

2. Run the KSPConfig.exe (KSP configuration wizard) utility to register the SafeNet KSP through a GUI. The general form of command is:

   `<Luna Client Installation Directory>\win32\KSP> KspConfig.exe`

   For Example: `C:\Program Files\SafeNet\LunaClient\win32\KSP>KspConfig.exe`

3. Double-click **Register or View Security Library** on the left side of the pane.

4. Browse the library `<Luna Client installation Directory>\win32\cryptoki.dll` library and click **Register**. On successful registration, you will see the following message: **"Success registering the security library"**.

5. Double-click **Register HSM Slots** on the left side of the pane.

   Open the **Register for User** drop-down menu and select the **User**. Open the **Domain** drop-down and select your domain.

   Open the **Available Slots** drop-down and select the partition.

   Enter the partition password in **Slot Password** field.

   Click **Register Slot** to register the slot for Domain\User. On successful registration, a message **"The slot was successfully and securely registered"** displays.

6. Double-click **Register HSM Slots** on the left side of the pane.

   a. Open the **Register for User** drop-down menu and select **NT_AUTHORITY**. Open the **Domain** drop-down and select **Domain**.

Open the **Available Slots** drop-down and select the partition.

Enter the partition password in **Slot Password** field.

Click **Register Slot** to register the slot for NT_AUTHORITY\SYSTEM. On successful registration, a message "**The slot was successfully and securely registered**" displays.

> **NOTE:** The partition has been registered for both users, despite only one entry appearing for the <slot_label> in the **Registered Slots** section of the KSP interface.

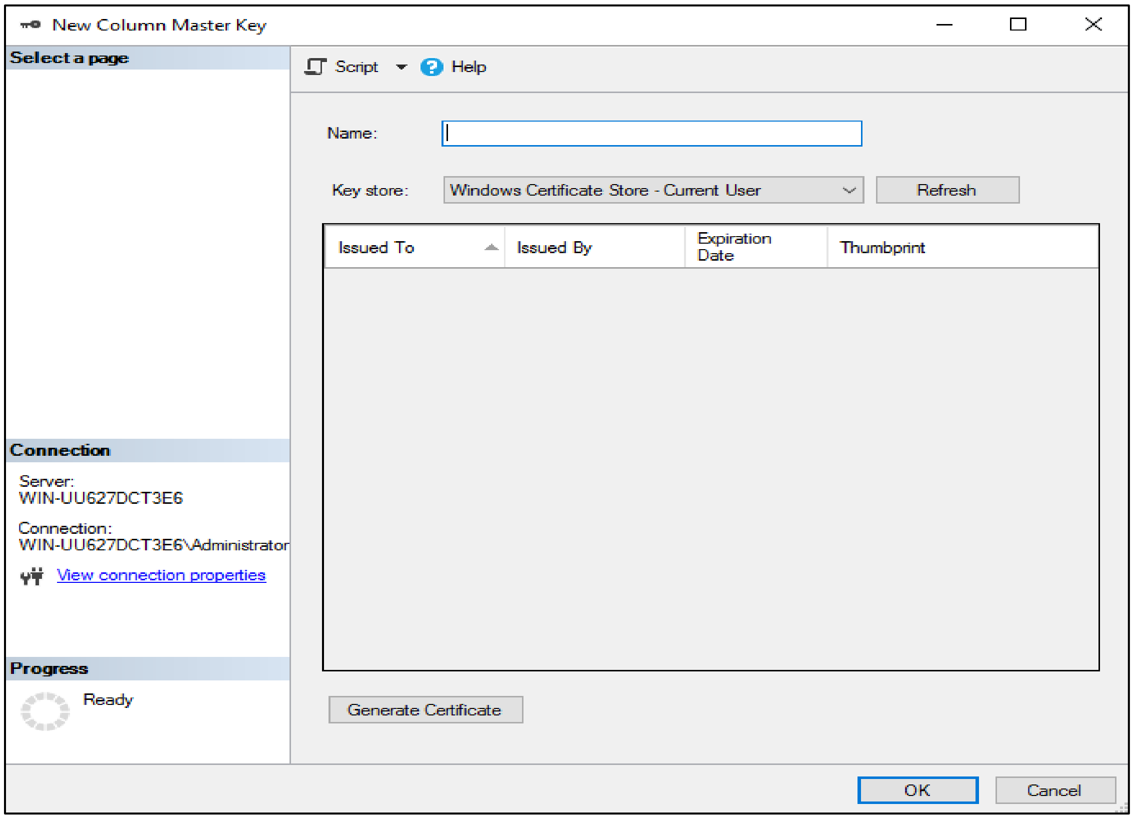
Generate Column Master Key

You require a Column Master Key to configure Always Encrypted. To generate the column master key:

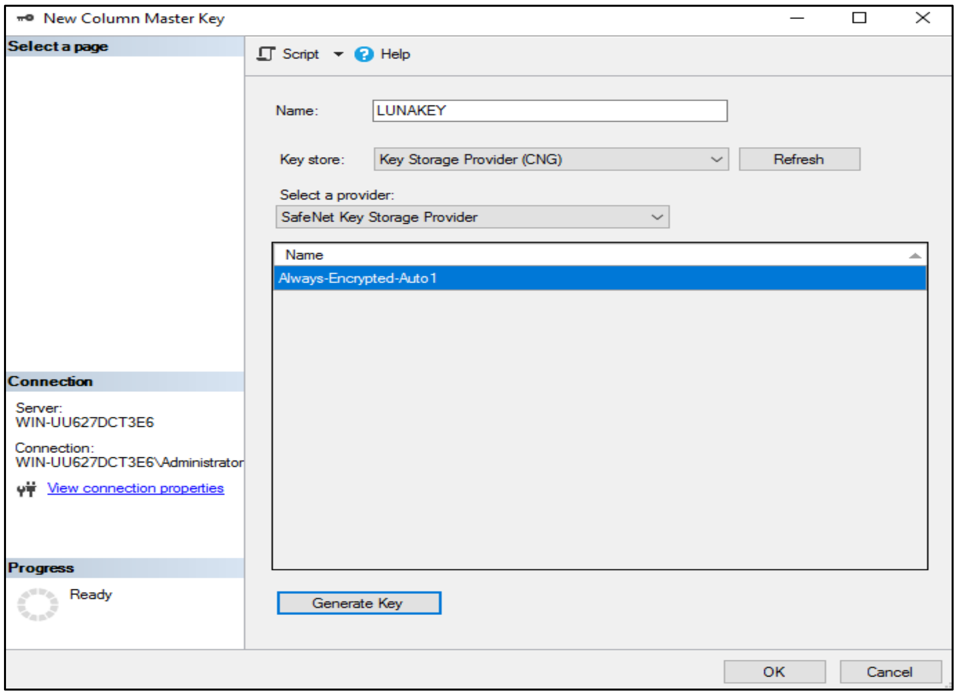
1. Connect to the database using **SQL Server Management Studio** from a client machine.
2. In Object Explorer, navigate to **Databases** > **Test** > **Security.**

> **NOTE:** Test is the sample database created for demonstration purpose.

3. Expand the **Always Encrypted Keys** folder to display its two subfolders:
   - Column Master Keys
   - Column Encryption Key
   - Right-click on **Column Master Keys** and select **New Column Master Key.** The **New Column Master Key** wizard appears on the screen.



4. Enter a name for the Column Master Key Pair in the **Name** field.
5. Open the **Key Store** drop-down menu and select **Key Storage Provider (CNG)**.
6. Open the **Select a provider** drop-down menu and select **SafeNet Key Storage Provider**. Click **Generate Key**.



Name: LUNAKEY

Key store: Key Storage Provider (CNG)

Select a provider: SafeNet Key Storage Provider

Name
Always-Encrypted-Auto1

Connection

Server:
WIN-UU627DCT3E6

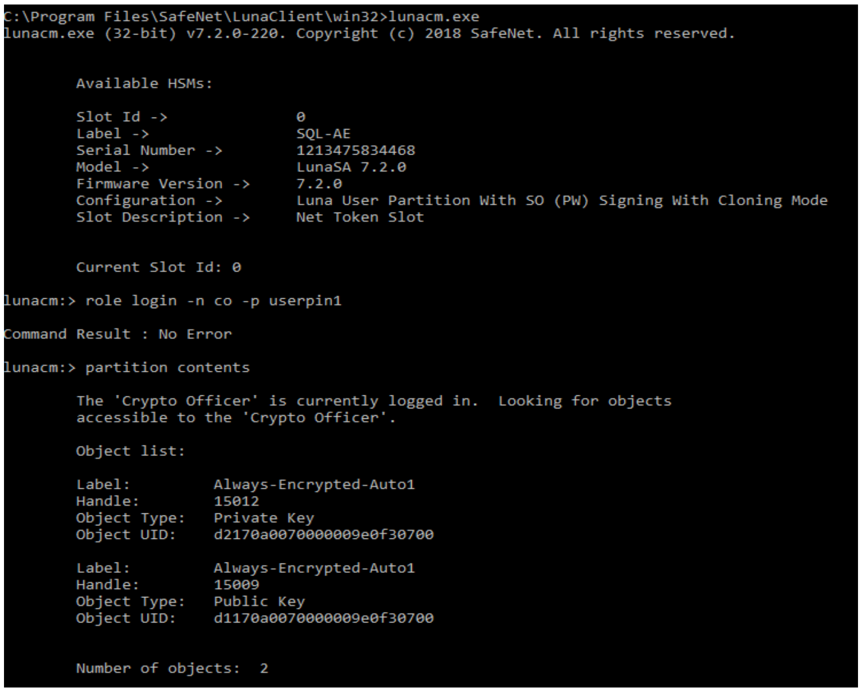
Connection:
WIN-UU627DCT3E6\Administrator

View connection properties

Progress

Ready

7. The key pair (column master key) gets generated on the Luna HSM partition. Execute **partition content** in lunacm to verify that the keys exist.



```
C:\Program Files\SafeNet\LunaClient\win32>lunacm.exe
lunacm.exe (32-bit) v7.2.0-220. Copyright (c) 2018 SafeNet. All rights reserved.


        Available HSMs:

        Slot Id ->              0
        Label ->                SQL-AE
        Serial Number ->        1213475834468
        Model ->                LunaSA 7.2.0
        Firmware Version ->     7.2.0
        Configuration ->        Luna User Partition With SO (PW) Signing With Cloning Mode
        Slot Description ->     Net Token Slot


        Current Slot Id: 0

lunacm:> role login -n co -p userpin1

Command Result : No Error

lunacm:> partition contents

        The 'Crypto Officer' is currently logged in.  Looking for objects
        accessible to the 'Crypto Officer'.

        Object list:

        Label:          Always-Encrypted-Auto1
        Handle:         15012
        Object Type:    Private Key
        Object UID:     d2170a0070000009e0f30700

        Label:          Always-Encrypted-Auto1
        Handle:         15009
        Object Type:    Public Key
        Object UID:     d1170a0070000009e0f30700


        Number of objects:  2
```

Generate Column Encryption Key

You require a Column Encryption Key to configure Always Encrypted. To generate the column encryption key:

1. In the Object Explorer, navigate to **Databases** > **Test** > **Security.**
2. Expand the **Always Encrypted Keys** folder to display its two subfolders :
   - Column Master Keys
   - Column Encryption Keys
3. Right-click on **Column Encryption Keys** and select **New Column Encryption Key.** The **New Column Encryption Key** wizard displays.
4. Enter a name for the Column Encryption Key in the **Name** field. Click the **Column Master Key** drop-down menu and select **LUNAKEY**.
5. Click **OK**.
6. The key is generated and is stored in the SQL Server Instance where Always Encrypted is implemented.

Implement Always Encrypted using SSMS

When you have configured the Column Master Key and Column Encryption Key, you can implement Always Encrypted on the SQL server.

NOTE: For demonstration purposes, further instructions will use Employee table as an example.

To implement Always Encrypted:

1. Create the table **Employee** with fields in database **Test**.

```
use Test;
Create Table Employee(
id numeric(10),
name varchar (50),
data varchar (max),);
```

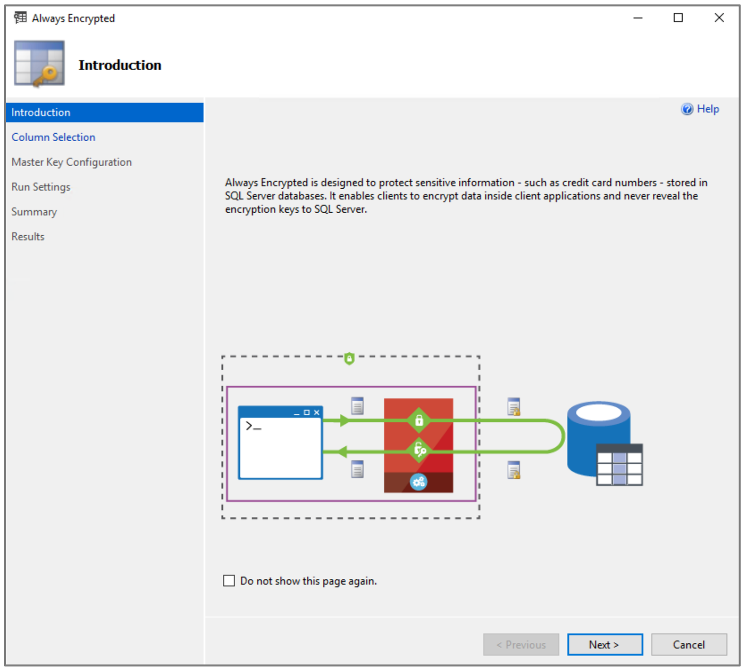
2. Insert some values into the table.

```
INSERT INTO dbo.Employee
values( 101,'Emp1','ConfidentialData'),(102,'Emp2','PrivateData');
```

3. View the table contents in plaintext.

```
Select * from dbo.Employee;
```

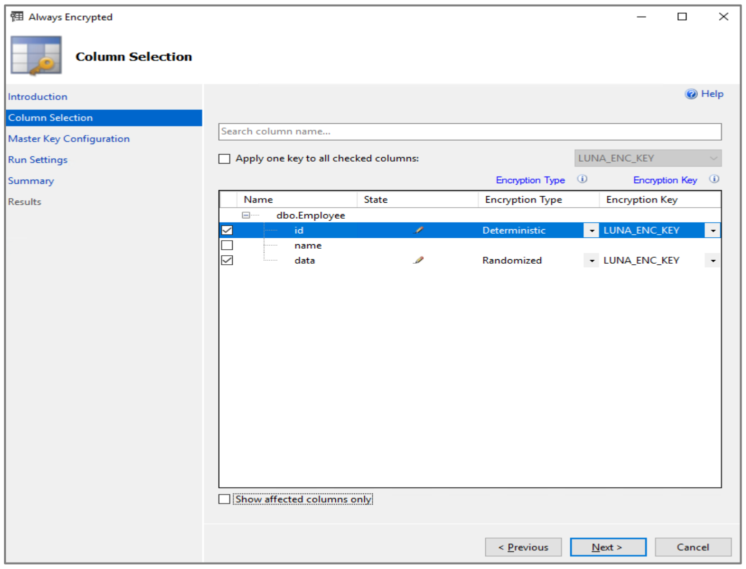
4. Encrypt the Employee details. Right-click the Employee table and select Encrypt Columns. The **Always Encrypted** wizard will appear on the screen.



Always Encrypted is designed to protect sensitive information - such as credit card numbers - stored in SQL Server databases. It enables clients to encrypt data inside client applications and never reveal the encryption keys to SQL Server.

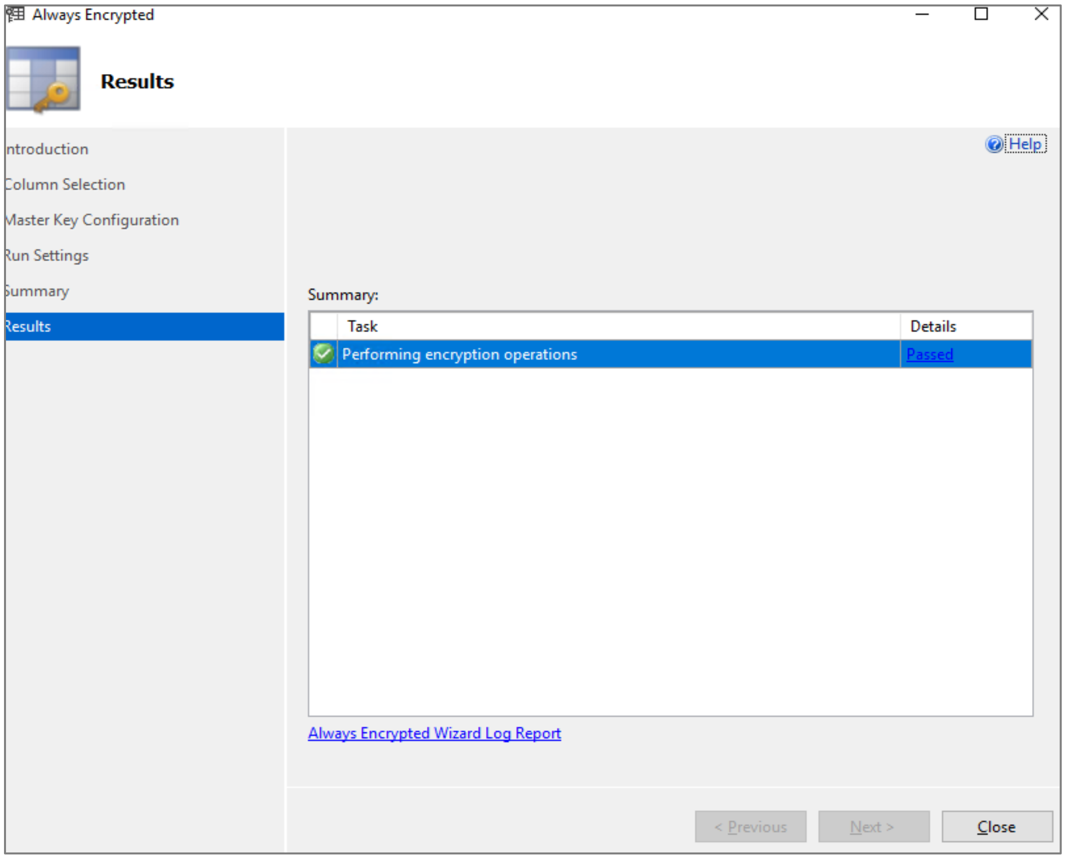
5. The introduction page displays. Click **Next**.
6. The **Column Selection** page allows you to select the columns to encrypt and the encryption type:
   - Randomized
   - Deterministic

   For the Employee table, set the Data column to be **Randomized**, and the id column to be **Deterministic**. Click **Next**.



	Name	State	Encryption Type	Encryption Key
	dbo.Employee			
☑	id		Deterministic	LUNA_ENC_KEY
☐	name			
☑	data		Randomized	LUNA_ENC_KEY

7. On the **Master Key Configuration** page, confirm the "No additional configuration is necessary because you are using existing keys" message. Click **Next**.
8. On the Run Settings page, select Proceed to finish now. Click **Next**.
9. Verify the details on **Summary** page. Click **Finish** to complete the encryption process.
10. Once the process gets completed, click **Close**.



Summary:

Task	Details
✓ Performing encryption operations	Passed

11. View the table contents.

    ```
    Select * from dbo.Employee;
    ```

    The Columns data and id now appear in encrypted form.

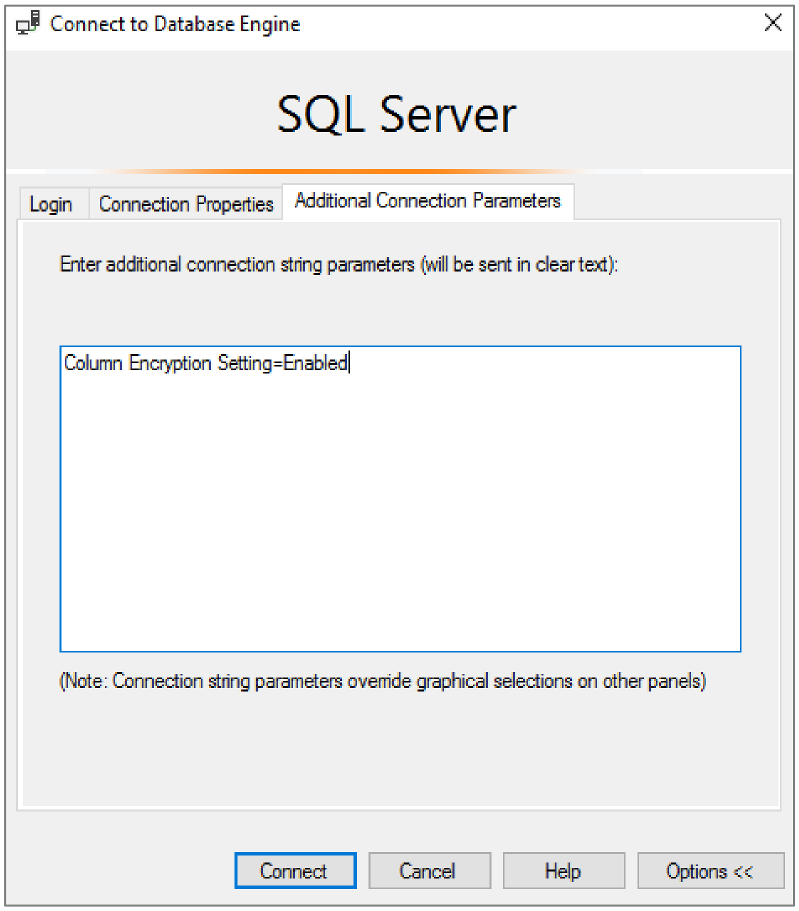
View Always Encrypted Data

Once you have configured SQL Server Always Encrypted you may need to access the encrypted data. To view Always Encrypted Data:

1. Select the **Query** Menu tab in SSMS. Point to **Connection**.
2. Click **Change Connection**. The **Connect to Database Engine** dialog appears on the screen.

3. Click **Options** and select the **Additional Connection Parameters** tab. On this tab, enter the following value:

```
Column Encryption Setting=Enabled
```



Enter additional connection string parameters (will be sent in clear text):

Column Encryption Setting=Enabled

(Note: Connection string parameters override graphical selections on other panels)

4. Click **Connect**.
5. Run the following query:

```
use Test;
Select * from dbo.Employee;
```

6. A pop-up screen prompts for **Parameterization for Always Encrypted**. Click **Enable**.

   This will display the encrypted data of the table.

Implement Always Encrypted using PowerShell: Without Role Separation

All commands must be executed using Administrator in the PowerShell. Open the PowerShell by right clicking and selecting **Run as Administrator**.

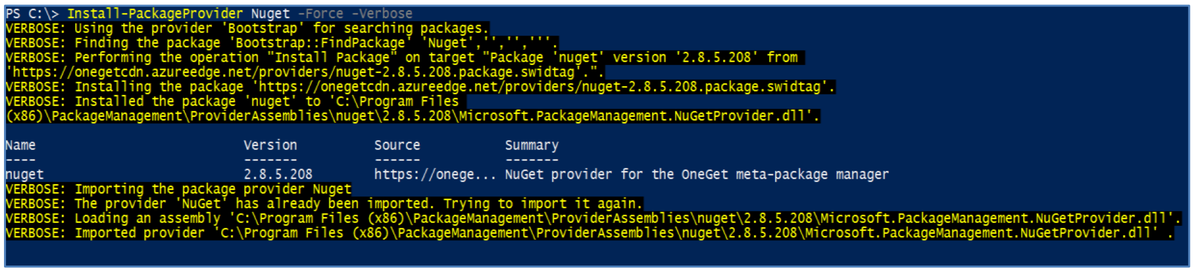
> **NOTE:** Ensure that all values are adjusted according to your environment. The values used in this integration guide are for example purpose only.

Install and configure SQL Server PowerShell module

Before proceeding, ensure that you have the required PowerShell modules installed.

1. Install Nuget provider using the command below:

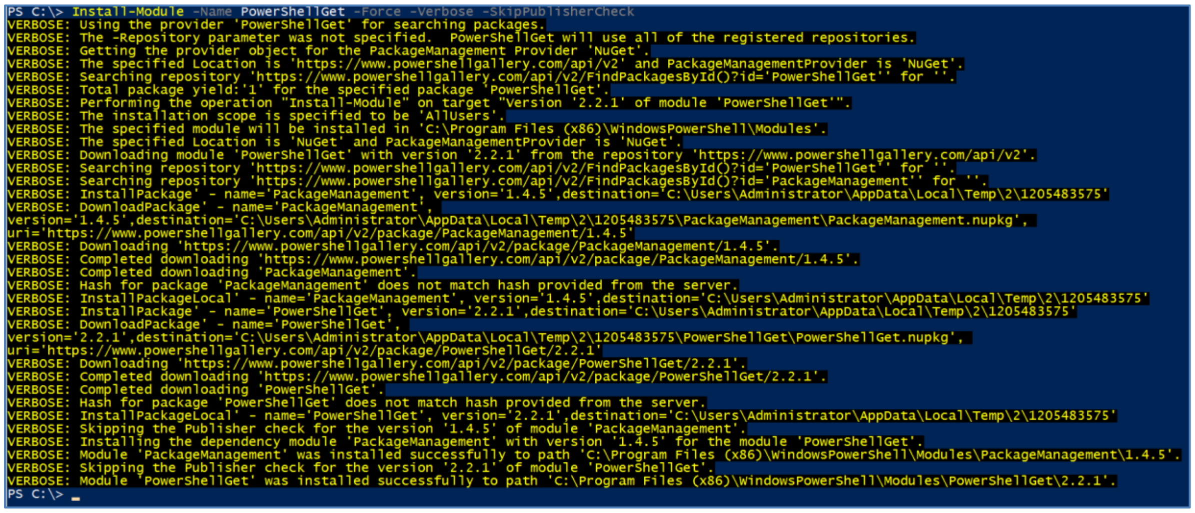
   ```
   Install-PackageProvider Nuget -Force -Verbose
   ```



```
PS C:\> Install-PackageProvider Nuget -Force -Verbose
VERBOSE: Using the provider 'Bootstrap' for searching packages.
VERBOSE: Finding the package 'Bootstrap::FindPackage' 'Nuget','','',''.
VERBOSE: Performing the operation "Install Package" on target "Package 'nuget' version '2.8.5.208' from
'https://onegetcdn.azureedge.net/providers/nuget-2.8.5.208.package.swidtag'.".
VERBOSE: Installing the package 'https://onegetcdn.azureedge.net/providers/nuget-2.8.5.208.package.swidtag'.
VERBOSE: Installed the package 'nuget' to 'C:\Program Files
(x86)\PackageManagement\ProviderAssemblies\nuget\2.8.5.208\Microsoft.PackageManagement.NuGetProvider.dll'.

Name                           Version          Source           Summary
----                           -------          ------           -------
nuget                          2.8.5.208        https://onege... NuGet provider for the OneGet meta-package manager
VERBOSE: Importing the package provider Nuget
VERBOSE: The provider 'NuGet' has already been imported. Trying to import it again.
VERBOSE: Loading an assembly 'C:\Program Files (x86)\PackageManagement\ProviderAssemblies\nuget\2.8.5.208\Microsoft.PackageManagement.NuGetProvider.dll'.
VERBOSE: Imported provider 'C:\Program Files (x86)\PackageManagement\ProviderAssemblies\nuget\2.8.5.208\Microsoft.PackageManagement.NuGetProvider.dll' .
```

2. Install PowerShellGet module using the command below:

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

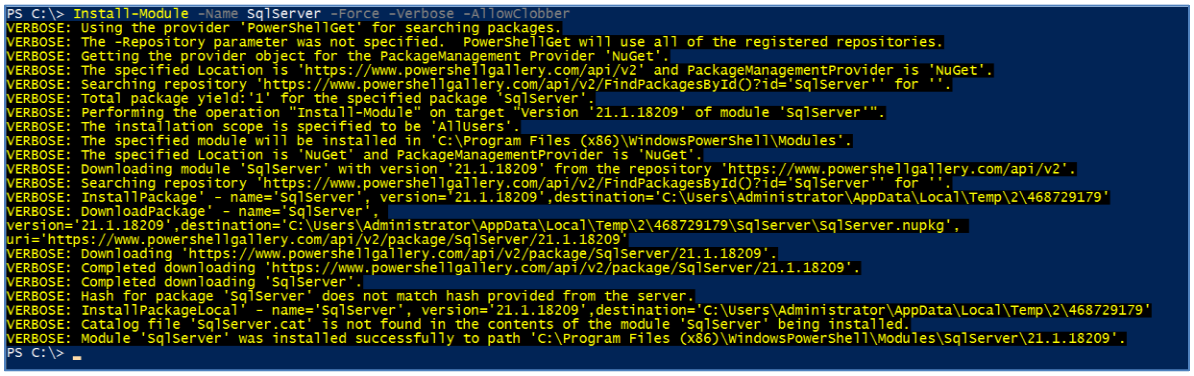


```
PS C:\> Install-Module -Name PowerShellGet -Force -Verbose -SkipPublisherCheck
VERBOSE: Using the provider 'PowerShellGet' for searching packages.
VERBOSE: The -Repository parameter was not specified.  PowerShellGet will use all of the registered repositories.
VERBOSE: Getting the provider object for the PackageManagement Provider 'NuGet'.
VERBOSE: The specified Location is 'https://www.powershellgallery.com/api/v2' and PackageManagementProvider is 'NuGet'.
VERBOSE: Searching repository 'https://www.powershellgallery.com/api/v2/FindPackagesById()?id='PowerShellGet'' for ''.
VERBOSE: Total package yield:'1' for the specified package 'PowerShellGet'.
VERBOSE: Performing the operation "Install-Module" on target "Version '2.2.1' of module 'PowerShellGet'".
VERBOSE: The installation scope is specified to be 'AllUsers'.
VERBOSE: The specified module will be installed in 'C:\Program Files (x86)\WindowsPowerShell\Modules'.
VERBOSE: The specified Location is 'NuGet' and PackageManagementProvider is 'NuGet'.
VERBOSE: Downloading module 'PowerShellGet' with version '2.2.1' from the repository 'https://www.powershellgallery.com/api/v2'.
VERBOSE: Searching repository 'https://www.powershellgallery.com/api/v2/FindPackagesById()?id='PowerShellGet'' for ''.
VERBOSE: Searching repository 'https://www.powershellgallery.com/api/v2/FindPackagesById()?id='PackageManagement'' for ''.
VERBOSE: InstallPackage' - name='PackageManagement', version='1.4.5',destination='C:\Users\Administrator\AppData\Local\Temp\2\1205483575'
VERBOSE: DownloadPackage' - name='PackageManagement',
version='1.4.5',destination='C:\Users\Administrator\AppData\Local\Temp\2\1205483575\PackageManagement\PackageManagement.nupkg',
uri='https://www.powershellgallery.com/api/v2/package/PackageManagement/1.4.5'
VERBOSE: Downloading 'https://www.powershellgallery.com/api/v2/package/PackageManagement/1.4.5'.
VERBOSE: Completed downloading 'https://www.powershellgallery.com/api/v2/package/PackageManagement/1.4.5'.
VERBOSE: Completed downloading 'PackageManagement'.
VERBOSE: Hash for package 'PackageManagement' does not match hash provided from the server.
VERBOSE: InstallPackageLocal' - name='PackageManagement', version='1.4.5',destination='C:\Users\Administrator\AppData\Local\Temp\2\1205483575'
VERBOSE: InstallPackage' - name='PowerShellGet', version='2.2.1',destination='C:\Users\Administrator\AppData\Local\Temp\2\1205483575'
VERBOSE: DownloadPackage' - name='PowerShellGet',
version='2.2.1',destination='C:\Users\Administrator\AppData\Local\Temp\2\1205483575\PowerShellGet\PowerShellGet.nupkg',
uri='https://www.powershellgallery.com/api/v2/package/PowerShellGet/2.2.1'
VERBOSE: Downloading 'https://www.powershellgallery.com/api/v2/package/PowerShellGet/2.2.1'.
VERBOSE: Completed downloading 'https://www.powershellgallery.com/api/v2/package/PowerShellGet/2.2.1'.
VERBOSE: Completed downloading 'PowerShellGet'.
VERBOSE: Hash for package 'PowerShellGet' does not match hash provided from the server.
VERBOSE: InstallPackageLocal' - name='PowerShellGet', version='2.2.1',destination='C:\Users\Administrator\AppData\Local\Temp\2\1205483575'
VERBOSE: Skipping the Publisher check for the version '1.4.5' of module 'PackageManagement'.
VERBOSE: Installing the dependency module 'PackageManagement' with version '1.4.5' for the module 'PowerShellGet'.
VERBOSE: Module 'PackageManagement' was installed successfully to path 'C:\Program Files (x86)\WindowsPowerShell\Modules\PackageManagement\1.4.5'.
VERBOSE: Skipping the Publisher check for the version '2.2.1' of module 'PowerShellGet'.
VERBOSE: Module 'PowerShellGet' was installed successfully to path 'C:\Program Files (x86)\WindowsPowerShell\Modules\PowerShellGet\2.2.1'.
PS C:\> _
```

NOTE: If any validation error occurs, use the –SkipPublisherCheck flag with the command.

3. Install SqlServer module using the following command:

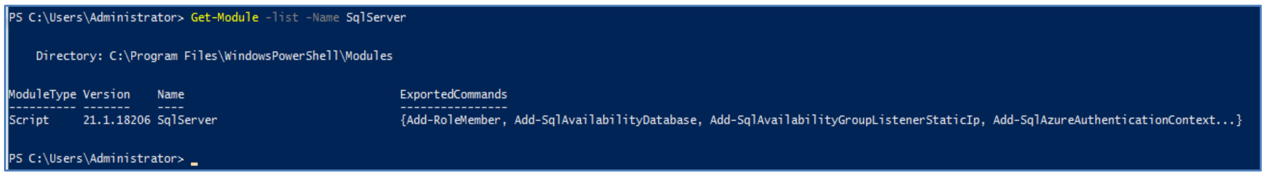
   ```
   Install-Module -Name SqlServer -Force -Verbose –AllowClobber
   ```



```
PS C:\> Install-Module -Name SqlServer -Force -Verbose -AllowClobber
VERBOSE: Using the provider 'PowerShellGet' for searching packages.
VERBOSE: The -Repository parameter was not specified.  PowerShellGet will use all of the registered repositories.
VERBOSE: Getting the provider object for the PackageManagement Provider 'NuGet'.
VERBOSE: The specified Location is 'https://www.powershellgallery.com/api/v2' and PackageManagementProvider is 'NuGet'.
VERBOSE: Searching repository 'https://www.powershellgallery.com/api/v2/FindPackagesById()?id='SqlServer'' for ''.
VERBOSE: Total package yield:'1' for the specified package 'SqlServer'.
VERBOSE: Performing the operation "Install-Module" on target "Version '21.1.18209' of module 'SqlServer'".
VERBOSE: The installation scope is specified to be 'AllUsers'.
VERBOSE: The specified module will be installed in 'C:\Program Files (x86)\WindowsPowerShell\Modules'.
VERBOSE: The specified Location is 'NuGet' and PackageManagementProvider is 'NuGet'.
VERBOSE: Downloading module 'SqlServer' with version '21.1.18209' from the repository 'https://www.powershellgallery.com/api/v2'.
VERBOSE: Searching repository 'https://www.powershellgallery.com/api/v2/FindPackagesById()?id='SqlServer'' for ''.
VERBOSE: InstallPackage' - name='SqlServer', version='21.1.18209',destination='C:\Users\Administrator\AppData\Local\Temp\2\468729179'
VERBOSE: DownloadPackage' - name='SqlServer',
version='21.1.18209',destination='C:\Users\Administrator\AppData\Local\Temp\2\468729179\SqlServer\SqlServer.nupkg',
uri='https://www.powershellgallery.com/api/v2/package/SqlServer/21.1.18209'
VERBOSE: Downloading 'https://www.powershellgallery.com/api/v2/package/SqlServer/21.1.18209'.
VERBOSE: Completed downloading 'https://www.powershellgallery.com/api/v2/package/SqlServer/21.1.18209'.
VERBOSE: Completed downloading 'SqlServer'.
VERBOSE: Hash for package 'SqlServer' does not match hash provided from the server.
VERBOSE: InstallPackageLocal' - name='SqlServer', version='21.1.18209',destination='C:\Users\Administrator\AppData\Local\Temp\2\468729179'
VERBOSE: Catalog file 'SqlServer.cat' is not found in the contents of the module 'SqlServer' being installed.
VERBOSE: Module 'SqlServer' was installed successfully to path 'C:\Program Files (x86)\WindowsPowerShell\Modules\SqlServer\21.1.18209'.
PS C:\> _
```

4. Once the SqlServer module is installed, confirm the install using the following command in a new PowerShell session:

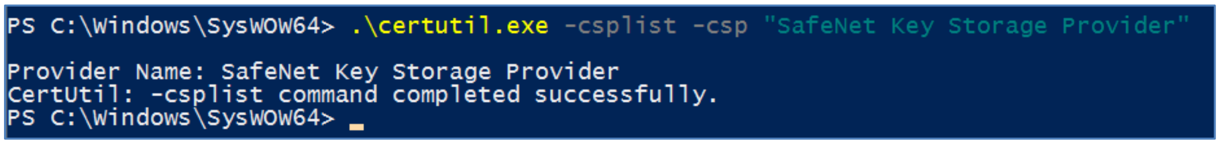
   ```
   Get-Module -list -Name SqlServer
   ```



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

    Directory: C:\Program Files\WindowsPowerShell\Modules

ModuleType Version    Name                                ExportedCommands
---------- -------    ----                                ----------------
Script     21.1.18206 SqlServer                           {Add-RoleMember, Add-SqlAvailabilityDatabase, Add-SqlAvailabilityGroupListenerStaticIp, Add-SqlAzureAuthenticationContext...}

PS C:\Users\Administrator> _
```

Configure SafeNet KSP

Please refer to the Configuring the SafeNet KSP section of this guide for details on installing and registering the SafeNet KSP. Verify that the SafeNet KSP is installed correctly.



```
PS C:\Windows\SysWOW64> .\certutil.exe -csplist -csp "SafeNet Key Storage Provider"

Provider Name: SafeNet Key Storage Provider
CertUtil: -csplist command completed successfully.
PS C:\Windows\SysWOW64> _
```

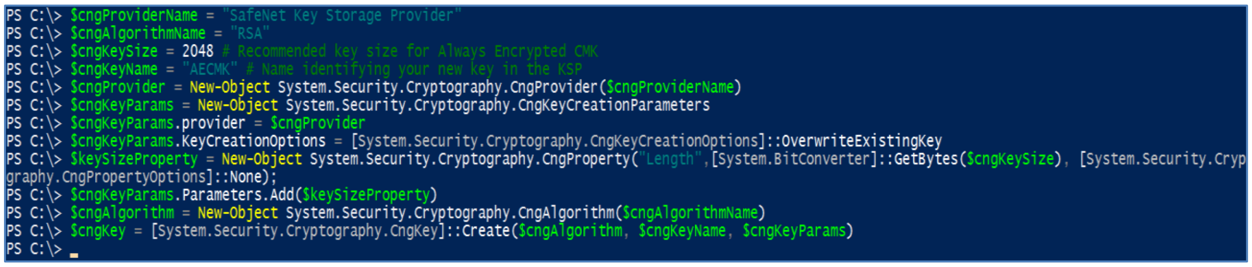
Create the Always Encrypted Column Master Key using the SafeNet KSP

Once you have successfully installed the SafeNet Key Storage Provider and registered it for use with Luna HSM partition, you can begin to configure Always Encrypted.

1. Generate an RSA key pair to use as a Column Master Key using the script below:

```
-------------------------------------------------------------------------------

$cngProviderName = "SafeNet Key Storage Provider"

$cngAlgorithmName = "RSA"

$cngKeySize = 2048        # Recommended key size for Always Encrypted CMK

$cngKeyName = "AECMK"     # Name identifying your new key in the KSP

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

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

$cngKeyParams.provider = $cngProvider

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

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

$cngKeyParams.Parameters.Add($keySizeProperty)

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

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



```
PS C:\> $cngProviderName = "SafeNet Key Storage Provider"
PS C:\> $cngAlgorithmName = "RSA"
PS C:\> $cngKeySize = 2048 # Recommended key size for Always Encrypted CMK
PS C:\> $cngKeyName = "AECMK" # Name identifying your new key in the KSP
PS C:\> $cngProvider = New-Object System.Security.Cryptography.CngProvider($cngProviderName)
PS C:\> $cngKeyParams = New-Object System.Security.Cryptography.CngKeyCreationParameters
PS C:\> $cngKeyParams.provider = $cngProvider
PS C:\> $cngKeyParams.KeyCreationOptions = [System.Security.Cryptography.CngKeyCreationOptions]::OverwriteExistingKey
PS C:\> $keySizeProperty = New-Object System.Security.Cryptography.CngProperty("Length",[System.BitConverter]::GetBytes($cngKeySize), [System.Security.Cryp
graphy.CngPropertyOptions]::None);
PS C:\> $cngKeyParams.Parameters.Add($keySizeProperty)
PS C:\> $cngAlgorithm = New-Object System.Security.Cryptography.CngAlgorithm($cngAlgorithmName)
PS C:\> $cngKey = [System.Security.Cryptography.CngKey]::Create($cngAlgorithm, $cngKeyName, $cngKeyParams)
PS C:\> _
```

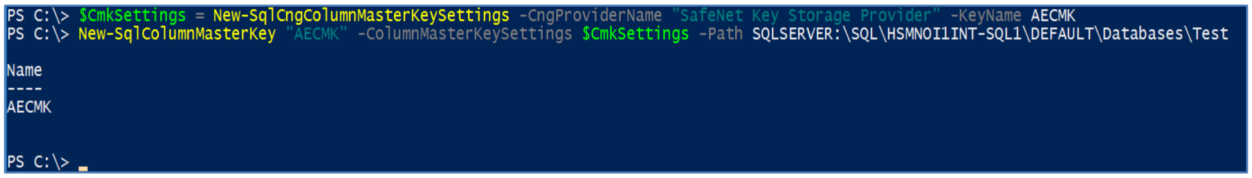
The above script, when executed successfully, will generate a 2048 bit RSA key pair with the Name **AECMK**.

2. Specify the Column Master Key settings for importing into the database using the following command:

```
$CmkSettings = New-SqlCngColumnMasterKeySettings -CngProviderName "SafeNet Key
Storage Provider" -KeyName "AECMK"
```

3. Finally, execute the command below to generate Column Master Key in database.

```
New-SqlColumnMasterKey "AECMK" -ColumnMasterKeySettings $CmkSettings -Path
SQLSERVER:\SQL\<your_server_name>\DEFAULT\Databases\<your_database_name>
```



```
PS C:\> $CmkSettings = New-SqlCngColumnMasterKeySettings -CngProviderName "SafeNet Key Storage Provider" -KeyName AECMK
PS C:\> New-SqlColumnMasterKey "AECMK" -ColumnMasterKeySettings $CmkSettings -Path SQLSERVER:\SQL\HSMNOI1INT-SQL1\DEFAULT\Databases\Test

Name
----
AECMK


PS C:\> _
```

NOTE: Replace <server_name> and <database_name> with actual values in your environment.

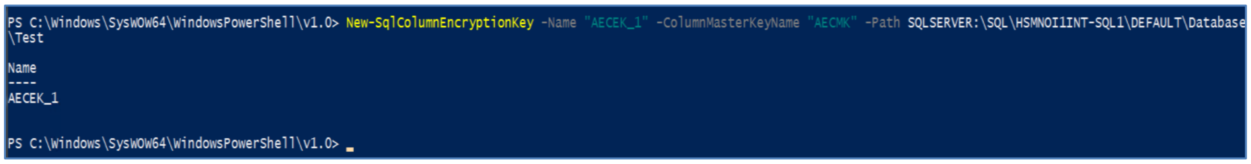
Create Column Encryption Key

1. After Column Master Key has been successfully generated, create a Column Encryption Key using the following command:

```
SqlColumnEncryptionKey -Name "AECEK_1" -ColumnMasterKeyName "AECMK" –Path
SQLSERVER:\SQL\<your_server_name>\DEFAULT\Databases\<your_database_name>
```

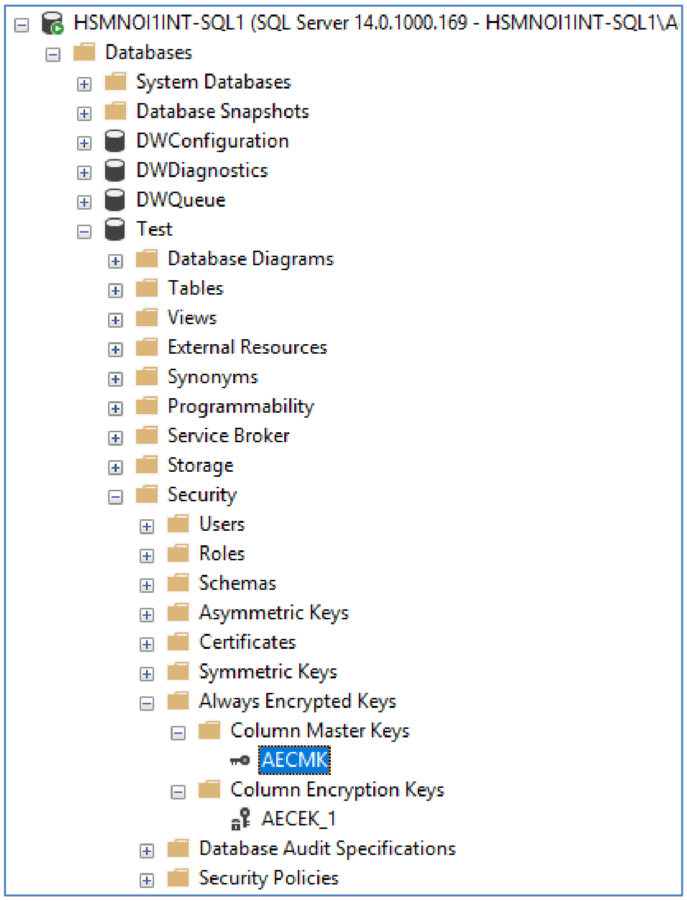
Where **AECEK_1** is the column encryption key and **AECMK** is the column master key. The resulting Column Encryption Key (CEK) is a 256 bit symmetric key protected by the Column Master Key (CMK)

NOTE: Replace the <server_name> and <database_name> with actual values in your environment.



```
PS C:\Windows\SysWOW64\WindowsPowerShell\v1.0> New-SqlColumnEncryptionKey -Name "AECEK_1" -ColumnMasterKeyName "AECMK" -Path SQLSERVER:\SQL\HSMNOI1INT-SQL1\DEFAULT\Database
\Test

Name
----
AECEK_1


PS C:\Windows\SysWOW64\WindowsPowerShell\v1.0> _
```

2. You can confirm the generated CMK and CEK using Object Explorer in SSMS.



```
⊟ HSMNOI1INT-SQL1 (SQL Server 14.0.1000.169 - HSMNOI1INT-SQL1\A
  ⊟ Databases
    ⊞ System Databases
    ⊞ Database Snapshots
    ⊞ DWConfiguration
    ⊞ DWDiagnostics
    ⊞ DWQueue
    ⊟ Test
      ⊞ Database Diagrams
      ⊞ Tables
      ⊞ Views
      ⊞ External Resources
      ⊞ Synonyms
      ⊞ Programmability
      ⊞ Service Broker
      ⊞ Storage
      ⊟ Security
        ⊞ Users
        ⊞ Roles
        ⊞ Schemas
        ⊞ Asymmetric Keys
        ⊞ Certificates
        ⊞ Symmetric Keys
        ⊟ Always Encrypted Keys
          ⊟ Column Master Keys
              AECMK
          ⊟ Column Encryption Keys
              AECEK_1
        ⊞ Database Audit Specifications
        ⊞ Security Policies
```

Encrypt Columns with Column Encryption Key

Create a table in the database to implement the Always Encrypted, if not created already. To implement Always Encrypted:

1. Create the table **Employee** with some fields in database **Test**.

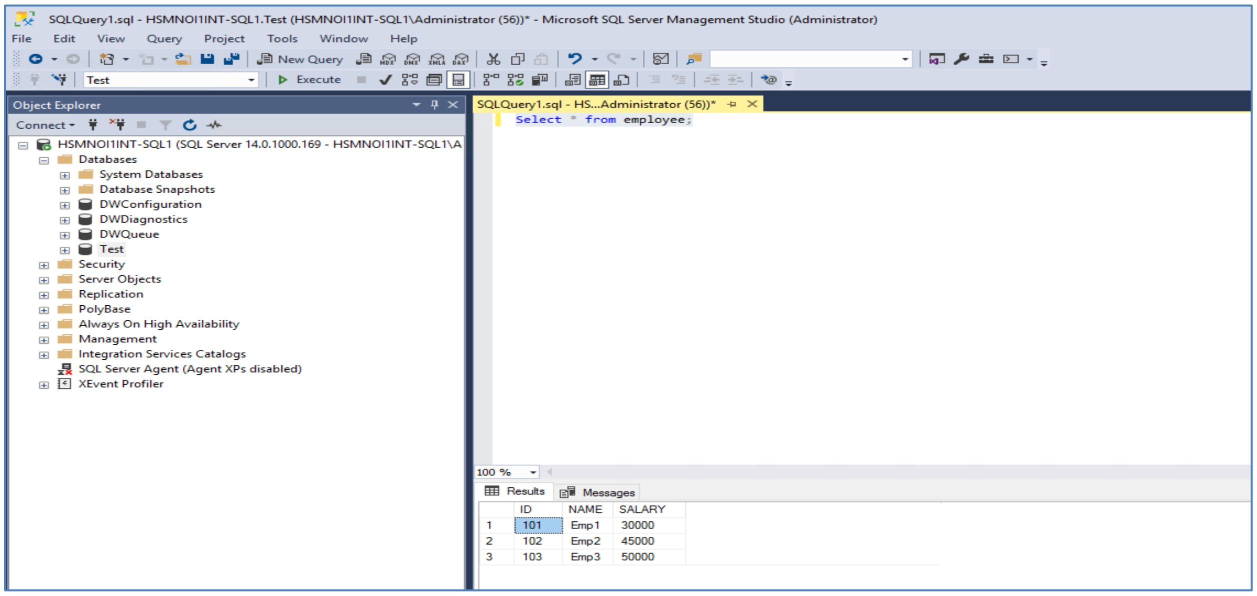
```
use Test;
Create Table Employee(
ID numeric(10),
NAME varchar (50),
SALARY varchar (max),);
```

2. Insert some values into the table.

```
INSERT INTO dbo.Employee
values( 101,'Emp1','30000'),(102,'Emp2','45000'),(103,'Emp3','50000');
```

3. View the table contents in plain text.

```
Select * from dbo.Employee;
```


SQLQuery1.sql - HSMNOI1INT-SQL1.Test (HSMNOI1INT-SQL1\Administrator (56))* - Microsoft SQL Server Management Studio (Administrator)
File Edit View Query Project Tools Window Help
New Query
Test
Execute
Object Explorer
Connect
HSMNOI1INT-SQL1 (SQL Server 14.0.1000.169 - HSMNOI1INT-SQL1\A
Databases
System Databases
Database Snapshots
DWConfiguration
DWDiagnostics
DWQueue
Test
Security
Server Objects
Replication
PolyBase
Always On High Availability
Management
Integration Services Catalogs
SQL Server Agent (Agent XPs disabled)
XEvent Profiler
SQLQuery1.sql - HS...Administrator (56))*
100 %
Results
Messages

```
Select * from employee;
```

	ID	NAME	SALARY
1	101	Emp1	30000
2	102	Emp2	45000
3	103	Emp3	50000

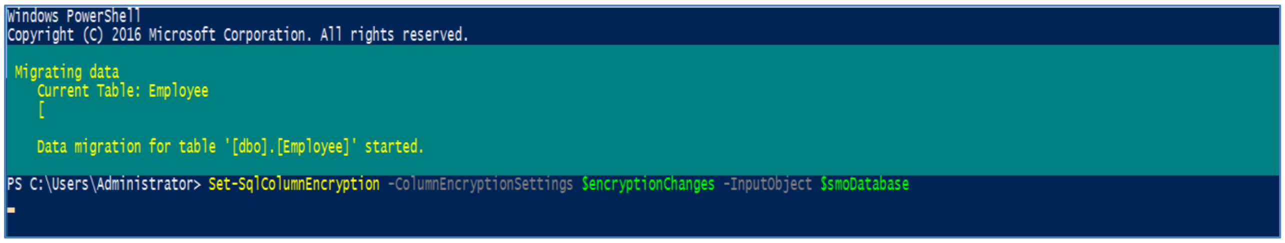
4. Open a PowerShell session with elevated permissions (right click and select “Run as Administrator”) and run the following script to encrypt a given column in the specified database. Adjust the values highlighted in bold to your database name and desired data columns.

   EncryptionType values are one of the following:

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

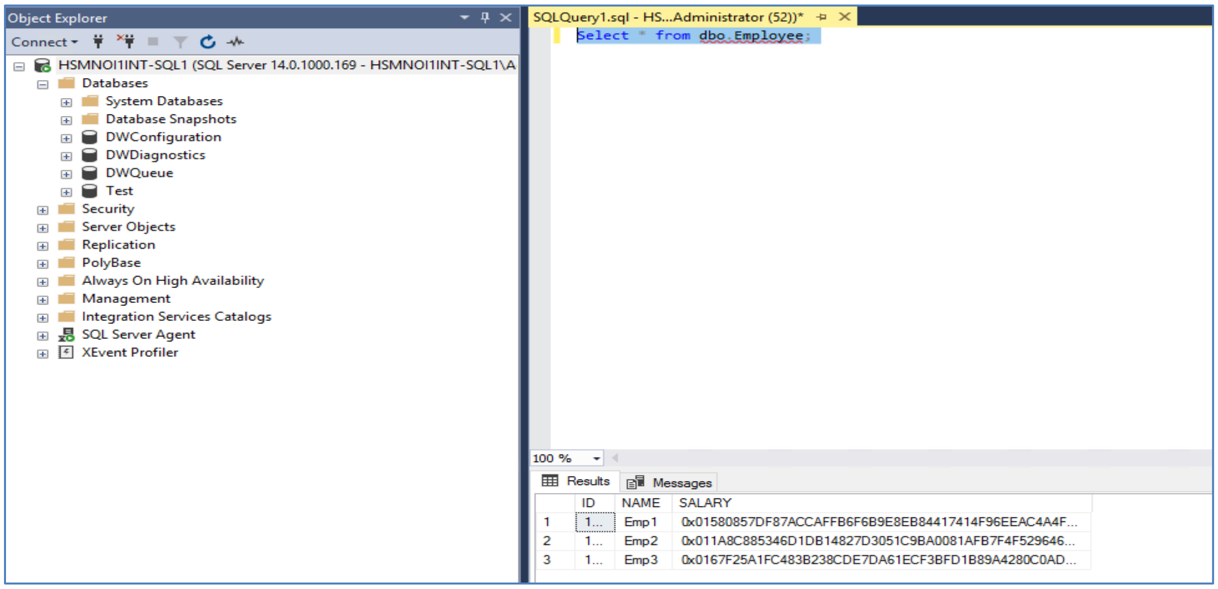
```
-------------------------------------------------------------------------------

#Import Module SqlServer

Import-Module SqlServer

#Set up connection and database SMO objects

$sqlConnectionString = "Data Source=server_name;Initial
Catalog=database_name;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].[Employee]

$encryptionChanges += New-SqlColumnEncryptionSettings -ColumnName
dbo.Employee.SALARY -EncryptionType Randomized -EncryptionKey "AECEK_1"
```

```
Set-SqlColumnEncryption -ColumnEncryptionSettings $encryptionChanges -
InputObject $smoDatabase
```



```
Windows PowerShell
Copyright (C) 2016 Microsoft Corporation. All rights reserved.

 Migrating data
    Current Table: Employee
    [

    Data migration for table '[dbo].[Employee]' started.

PS C:\Users\Administrator> Set-SqlColumnEncryption -ColumnEncryptionSettings $encryptionChanges -InputObject $smoDatabase
```

5. Verify that the column has been encrypted.



```
Select * from dbo.Employee;
```

	ID	NAME	SALARY
1	1...	Emp1	0x01580857DF87ACCAFFB6F6B9E8EB84417414F96EEAC4A4F...
2	1...	Emp2	0x011A8C885346D1DB14827D3051C9BA0081AFB7F4F529646...
3	1...	Emp3	0x0167F25A1FC483B238CDE7DA61ECF3BFD1B89A4280C0AD...

To view the encrypted columns in plain text refer to the "Viewing Always Encrypted Data" section in this guide.

Implement Always Encrypted using PowerShell: Role Separation

All commands must be executed using Administrator in the PowerShell. Open the PowerShell by right click and select **Run as Administrator**.

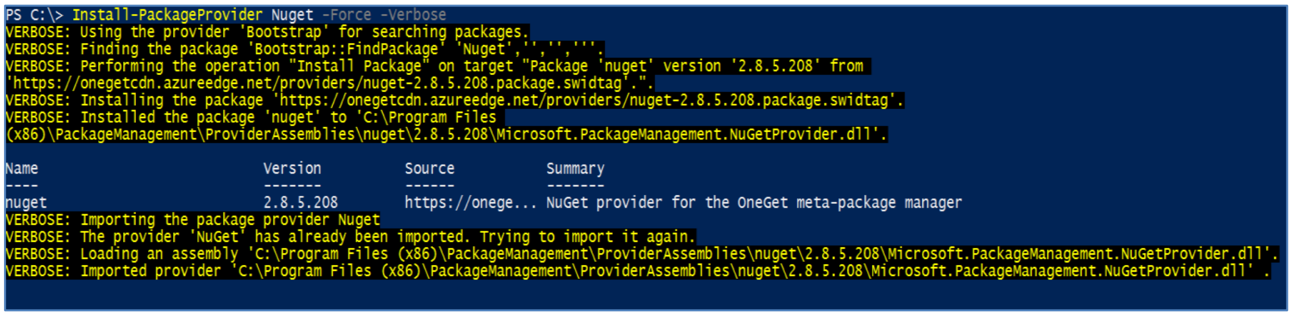
> **NOTE:** Ensure that all values are adjusted according to your environment. The values in this integration guide are example values only.

Install and configure SQLServer PowerShell module

Before proceeding, ensure that you have the required PowerShell modules installed.

1. Install **Nuget** provider using the command below:

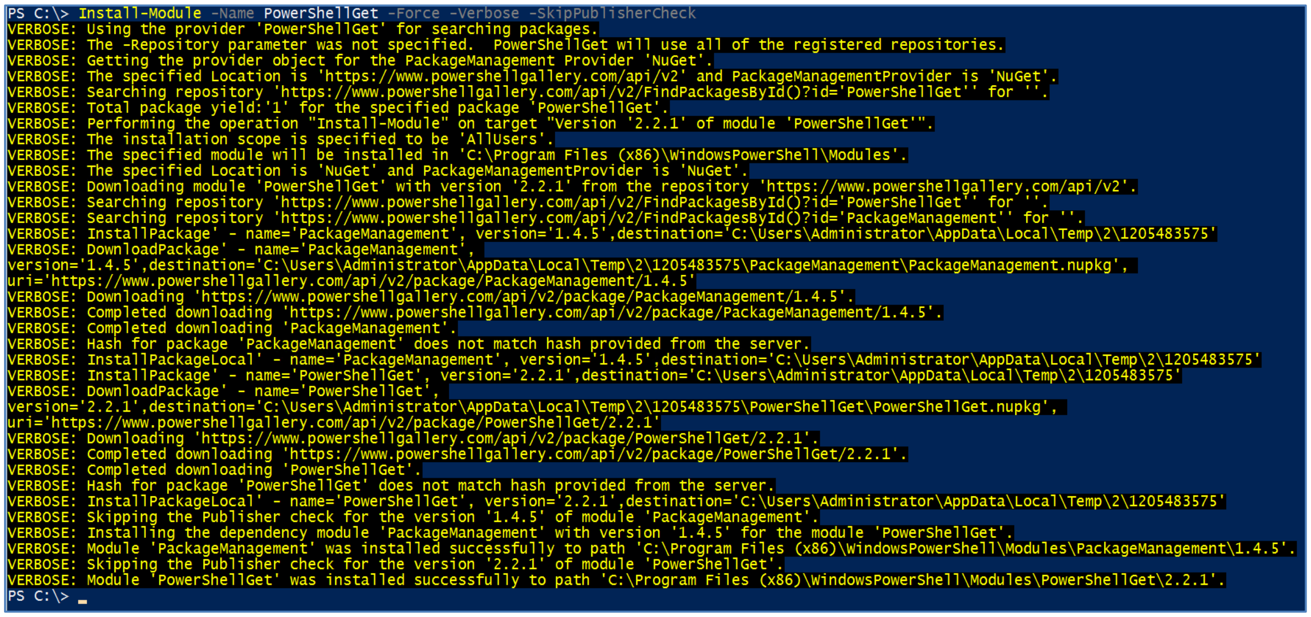
   ```
   Install-PackageProvider Nuget –Force –Verbose
   ```



```
PS C:\> Install-PackageProvider Nuget -Force -Verbose
VERBOSE: Using the provider 'Bootstrap' for searching packages.
VERBOSE: Finding the package 'Bootstrap::FindPackage' 'Nuget','','',''.
VERBOSE: Performing the operation "Install Package" on target "Package 'nuget' version '2.8.5.208' from
'https://onegetcdn.azureedge.net/providers/nuget-2.8.5.208.package.swidtag'.".
VERBOSE: Installing the package 'https://onegetcdn.azureedge.net/providers/nuget-2.8.5.208.package.swidtag'.
VERBOSE: Installed the package 'nuget' to 'C:\Program Files
(x86)\PackageManagement\ProviderAssemblies\nuget\2.8.5.208\Microsoft.PackageManagement.NuGetProvider.dll'.

Name                             Version          Source           Summary
----                             -------          ------           -------
nuget                            2.8.5.208        https://onege... NuGet provider for the OneGet meta-package manager
VERBOSE: Importing the package provider Nuget
VERBOSE: The provider 'NuGet' has already been imported. Trying to import it again.
VERBOSE: Loading an assembly 'C:\Program Files (x86)\PackageManagement\ProviderAssemblies\nuget\2.8.5.208\Microsoft.PackageManagement.NuGetProvider.dll'.
VERBOSE: Imported provider 'C:\Program Files (x86)\PackageManagement\ProviderAssemblies\nuget\2.8.5.208\Microsoft.PackageManagement.NuGetProvider.dll' .
```

2. Install **PowerShellGet** module using the following command:

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

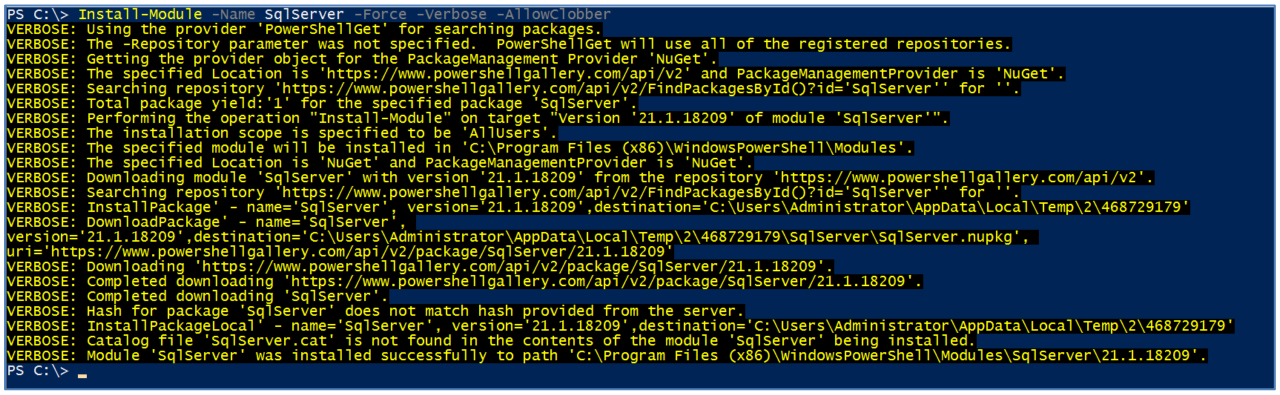


```
PS C:\> Install-Module -Name PowerShellGet -Force -Verbose -SkipPublisherCheck
VERBOSE: Using the provider 'PowerShellGet' for searching packages.
VERBOSE: The -Repository parameter was not specified.  PowerShellGet will use all of the registered repositories.
VERBOSE: Getting the provider object for the PackageManagement Provider 'NuGet'.
VERBOSE: The specified Location is 'https://www.powershellgallery.com/api/v2' and PackageManagementProvider is 'NuGet'.
VERBOSE: Searching repository 'https://www.powershellgallery.com/api/v2/FindPackagesById()?id='PowerShellGet'' for ''.
VERBOSE: Total package yield:'1' for the specified package 'PowerShellGet'.
VERBOSE: Performing the operation "Install-Module" on target "Version '2.2.1' of module 'PowerShellGet'".
VERBOSE: The installation scope is specified to be 'AllUsers'.
VERBOSE: The specified module will be installed in 'C:\Program Files (x86)\WindowsPowerShell\Modules'.
VERBOSE: The specified Location is 'NuGet' and PackageManagementProvider is 'NuGet'.
VERBOSE: Downloading module 'PowerShellGet' with version '2.2.1' from the repository 'https://www.powershellgallery.com/api/v2'.
VERBOSE: Searching repository 'https://www.powershellgallery.com/api/v2/FindPackagesById()?id='PowerShellGet'' for ''.
VERBOSE: Searching repository 'https://www.powershellgallery.com/api/v2/FindPackagesById()?id='PackageManagement'' for ''.
VERBOSE: InstallPackage' - name='PackageManagement', version='1.4.5',destination='C:\Users\Administrator\AppData\Local\Temp\2\1205483575'
VERBOSE: DownloadPackage' - name='PackageManagement',
version='1.4.5',destination='C:\Users\Administrator\AppData\Local\Temp\2\1205483575\PackageManagement\PackageManagement.nupkg',
uri='https://www.powershellgallery.com/api/v2/package/PackageManagement/1.4.5'
VERBOSE: Downloading 'https://www.powershellgallery.com/api/v2/package/PackageManagement/1.4.5'.
VERBOSE: Completed downloading 'https://www.powershellgallery.com/api/v2/package/PackageManagement/1.4.5'.
VERBOSE: Completed downloading 'PackageManagement'.
VERBOSE: Hash for package 'PackageManagement' does not match hash provided from the server.
VERBOSE: InstallPackageLocal' - name='PackageManagement', version='1.4.5',destination='C:\Users\Administrator\AppData\Local\Temp\2\1205483575'
VERBOSE: InstallPackage' - name='PowerShellGet', version='2.2.1',destination='C:\Users\Administrator\AppData\Local\Temp\2\1205483575'
VERBOSE: DownloadPackage' - name='PowerShellGet',
version='2.2.1',destination='C:\Users\Administrator\AppData\Local\Temp\2\1205483575\PowerShellGet\PowerShellGet.nupkg',
uri='https://www.powershellgallery.com/api/v2/package/PowerShellGet/2.2.1'
VERBOSE: Downloading 'https://www.powershellgallery.com/api/v2/package/PowerShellGet/2.2.1'.
VERBOSE: Completed downloading 'https://www.powershellgallery.com/api/v2/package/PowerShellGet/2.2.1'.
VERBOSE: Completed downloading 'PowerShellGet'.
VERBOSE: Hash for package 'PowerShellGet' does not match hash provided from the server.
VERBOSE: InstallPackageLocal' - name='PowerShellGet', version='2.2.1',destination='C:\Users\Administrator\AppData\Local\Temp\2\1205483575'
VERBOSE: Skipping the Publisher check for the version '1.4.5' of module 'PackageManagement'.
VERBOSE: Installing the dependency module 'PackageManagement' with version '1.4.5' for the module 'PowerShellGet'.
VERBOSE: Module 'PackageManagement' was installed successfully to path 'C:\Program Files (x86)\WindowsPowerShell\Modules\PackageManagement\1.4.5'.
VERBOSE: Skipping the Publisher check for the version '2.2.1' of module 'PowerShellGet'.
VERBOSE: Module 'PowerShellGet' was installed successfully to path 'C:\Program Files (x86)\WindowsPowerShell\Modules\PowerShellGet\2.2.1'.
PS C:\> _
```

NOTE: If any validation error occurred, use the –SkipPublisherCheck flag with the command.

3. Install **SqlServer** module using the following command:

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

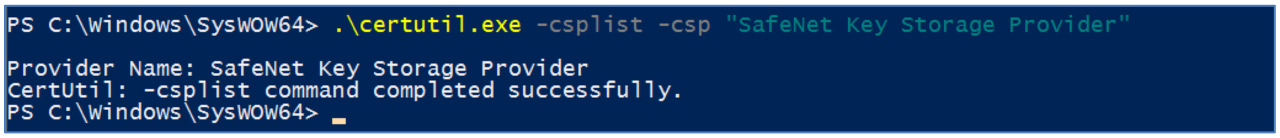


```
PS C:\> Install-Module -Name SqlServer -Force -Verbose -AllowClobber
VERBOSE: Using the provider 'PowerShellGet' for searching packages.
VERBOSE: The -Repository parameter was not specified.  PowerShellGet will use all of the registered repositories.
VERBOSE: Getting the provider object for the PackageManagement Provider 'NuGet'.
VERBOSE: The specified Location is 'https://www.powershellgallery.com/api/v2' and PackageManagementProvider is 'NuGet'.
VERBOSE: Searching repository 'https://www.powershellgallery.com/api/v2/FindPackagesById()?id='SqlServer'' for ''.
VERBOSE: Total package yield:'1' for the specified package 'SqlServer'.
VERBOSE: Performing the operation "Install-Module" on target "Version '21.1.18209' of module 'SqlServer'".
VERBOSE: The installation scope is specified to be 'AllUsers'.
VERBOSE: The specified module will be installed in 'C:\Program Files (x86)\WindowsPowerShell\Modules'.
VERBOSE: The specified Location is 'NuGet' and PackageManagementProvider is 'NuGet'.
VERBOSE: Downloading module 'SqlServer' with version '21.1.18209' from the repository 'https://www.powershellgallery.com/api/v2'.
VERBOSE: Searching repository 'https://www.powershellgallery.com/api/v2/FindPackagesById()?id='SqlServer'' for ''.
VERBOSE: InstallPackage' - name='SqlServer', version='21.1.18209',destination='C:\Users\Administrator\AppData\Local\Temp\2\468729179'
VERBOSE: DownloadPackage' - name='SqlServer',
version='21.1.18209',destination='C:\Users\Administrator\AppData\Local\Temp\2\468729179\SqlServer\SqlServer.nupkg',
uri='https://www.powershellgallery.com/api/v2/package/SqlServer/21.1.18209'
VERBOSE: Downloading 'https://www.powershellgallery.com/api/v2/package/SqlServer/21.1.18209'.
VERBOSE: Completed downloading 'https://www.powershellgallery.com/api/v2/package/SqlServer/21.1.18209'.
VERBOSE: Completed downloading 'SqlServer'.
VERBOSE: Hash for package 'SqlServer' does not match hash provided from the server.
VERBOSE: InstallPackageLocal' - name='SqlServer', version='21.1.18209',destination='C:\Users\Administrator\AppData\Local\Temp\2\468729179'
VERBOSE: Catalog file 'SqlServer.cat' is not found in the contents of the module 'SqlServer' being installed.
VERBOSE: Module 'SqlServer' was installed successfully to path 'C:\Program Files (x86)\WindowsPowerShell\Modules\SqlServer\21.1.18209'.
PS C:\> _
```

Configure SafeNet KSP

Refer to the Configuring the SafeNet KSP section of this guide for details on installing and registering the SafeNet KSP.

Verify that the SafeNet KSP is installed correctly.



```
PS C:\Windows\SysWOW64> .\certutil.exe -csplist -csp "SafeNet Key Storage Provider"

Provider Name: SafeNet Key Storage Provider
CertUtil: -csplist command completed successfully.
PS C:\Windows\SysWOW64> _
```

Create Always Encrypted Column Master Key using SafeNet KSP

For the purpose of this integration guide, when integrating with role separation, roles and processes are defined in table below. The table shows the separation and function of these roles with reference to Security Administrator and Database Administrator.

Process	Role
Generating the Column Master Key (CMK)	Security Administrator
Generating / encryption of Column Encryption Key (CEK)	Security Administrator
Defining the CMK and CEK in the database	Database Administrator
Encrypt database columns with CEK	Security Administrator

Once you have successfully installed the SafeNet Key Storage Provider and registered it for use with Luna HSM partition, you can begin to configure Always Encrypted. The Security Administrator 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.

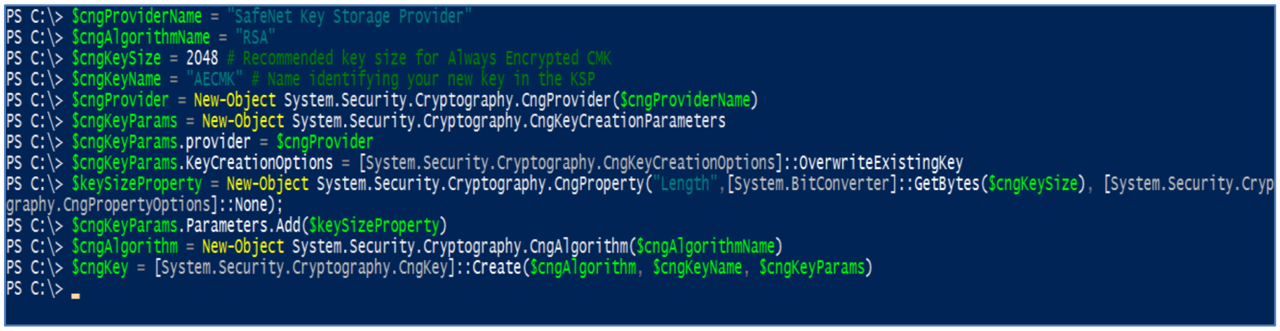
Generate CMK as Security Administrator

1. Log in as the Security Administrator.
2. Confirm that the SqlServer module is present by running the following command in the new PowerShell session.

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

3. Generate a RSA key pair to use as a Column Master Key using the script below:

   ```
   $cngProviderName = "SafeNet Key Storage Provider"

   $cngAlgorithmName = "RSA"

   $cngKeySize = 2048        # Recommended key size for Always Encrypted CMK

   $cngKeyName = "AECMK"     # Name identifying your new key in the KSP

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

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

   $cngKeyParams.provider = $cngProvider

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

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

   $cngKeyParams.Parameters.Add($keySizeProperty)

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

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



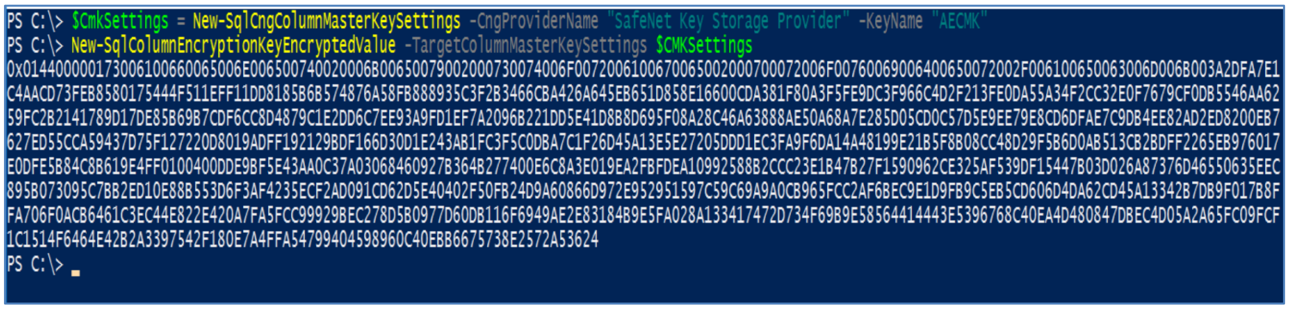
```
PS C:\> $cngProviderName = "SafeNet Key Storage Provider"
PS C:\> $cngAlgorithmName = "RSA"
PS C:\> $cngKeySize = 2048 # Recommended key size for Always Encrypted CMK
PS C:\> $cngKeyName = "AECMK" # Name identifying your new key in the KSP
PS C:\> $cngProvider = New-Object System.Security.Cryptography.CngProvider($cngProviderName)
PS C:\> $cngKeyParams = New-Object System.Security.Cryptography.CngKeyCreationParameters
PS C:\> $cngKeyParams.provider = $cngProvider
PS C:\> $cngKeyParams.KeyCreationOptions = [System.Security.Cryptography.CngKeyCreationOptions]::OverwriteExistingKey
PS C:\> $keySizeProperty = New-Object System.Security.Cryptography.CngProperty("Length",[System.BitConverter]::GetBytes($cngKeySize), [System.Security.Cryp
graphy.CngPropertyOptions]::None);
PS C:\> $cngKeyParams.Parameters.Add($keySizeProperty)
PS C:\> $cngAlgorithm = New-Object System.Security.Cryptography.CngAlgorithm($cngAlgorithmName)
PS C:\> $cngKey = [System.Security.Cryptography.CngKey]::Create($cngAlgorithm, $cngKeyName, $cngKeyParams)
PS C:\> _
```

The above script, when executed successfully, will generate a 2048-bit RSA key pair with Name **AECMK**.

4. Specify the Column Master Key settings for importing into the database using the command below:

   ```
   $CmkSettings = New-SqlCngColumnMasterKeySettings -CngProviderName "SafeNet Key
   Storage Provider" -KeyName "AECMK"
   ```

5. 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

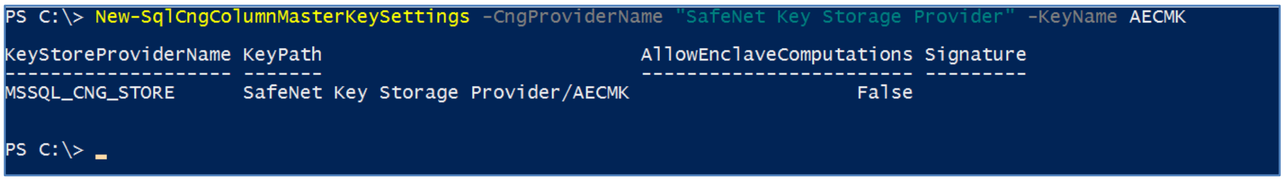


```
PS C:\> $CmkSettings = New-SqlCngColumnMasterKeySettings -CngProviderName "SafeNet Key Storage Provider" -KeyName "AECMK"
PS C:\> New-SqlColumnEncryptionKeyEncryptedValue -TargetColumnMasterKeySettings $CMKSettings
0x014400000173006100660065006E006500740020006B0065007900200073007400 6F007200610067006500200070007200 6F0076006900640065007200 2F006100650063006D006B003A2DFA7E1
C4AACD73FEB8580175444F511EFF11DD8185B6B574876A58FB888935C3F2B3466CBA426A645EB651D858E16600CDA381F80A3F5FE9DC3F966C4D2F213FE0DA55A34F2CC32E0F7679CF0DB5546AA62
59FC2B2141789D17DE85B69B7CDF6CC8D4879C1E2DD6C7EE93A9FD1EF7A2096B221DD5E41D8B8D695F08A28C46A63888AE50A68A7E285D05CD0C57D5E9EE79E8CD6DFAE7C9DB4EE82AD2ED8200EB7
627ED55CCA59437D75F127220D8019ADFF192129BDF166D30D1E243AB1FC3F5C0DBA7C1F26D45A13E5E27205DDD1EC3FA9F6DA14A48199E21B5F8B08CC48D29F5B6D0AB513CB2BDFF2265EB976017
E0DFE5B84C8B619E4FF0100400DDE9BF5E43AA0C37A03068460927B364B277400E6C8A3E019EA2FBFDEA10992588B2CCC23E1B47B27F1590962CE325AF539DF15447B03D026A87376D46550635EEC
895B073095C7BB2ED10E88B553D6F3AF4235ECF2AD091CD62D5E40402F50FB24D9A60866D972E952951597C59C69A9A0CB965FCC2AF6BEC9E1D9FB9C5EB5CD606D4DA62CD45A13342B7DB9F017B8F
FA706F0ACB6461C3EC44E822E420A7FA5FCC99929BEC278D5B0977D60DB116F6949AE2E83184B9E5FA028A133417472D734F69B9E58564414443E5396768C40EA4D480847DBEC4D05A2A65FC09FCF
1C1514F6464E42B2A3397542F180E7A4FFA54799404598960C40EBB6675738E2572A53624
PS C:\> _
```

Define objects in database and generate CEK as Database Administrator

1. Log in as the Database Administrator.
2. Open PowerShell as Administrator and run the following commands in order to create a **SqlcngcolumnMasterKeySettings** object that contains information about the location of your column master key.

```
New-SqlCngColumnMasterKeySettings -CngProviderName "SafeNet Key Storage
Provider" -KeyName AECMK
```

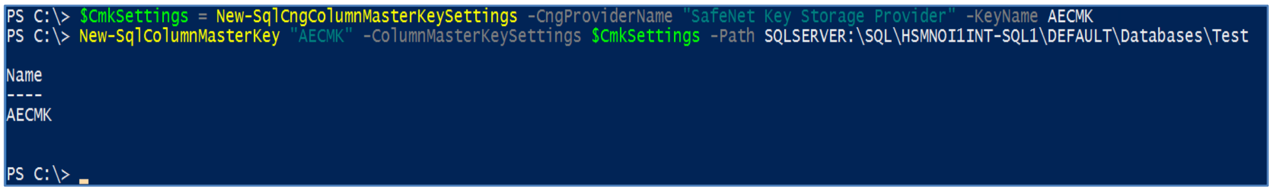


```
PS C:\> New-SqlCngColumnMasterKeySettings -CngProviderName "SafeNet Key Storage Provider" -KeyName AECMK

KeyStoreProviderName KeyPath                                AllowEnclaveComputations Signature
-------------------- -------                                ------------------------ ---------
MSSQL_CNG_STORE      SafeNet Key Storage Provider/AECMK                        False

PS C:\> _
```

```
$CmkSettings = New-SqlCngColumnMasterKeySettings -CngProviderName "SafeNet Key
Storage Provider" -KeyName AECMK
```

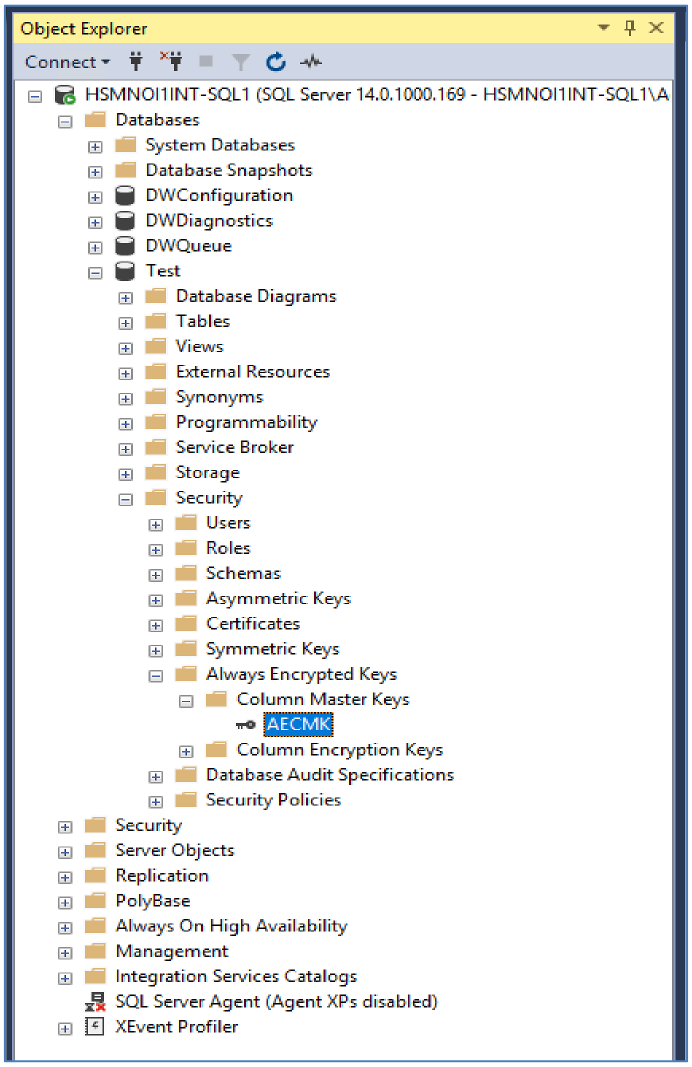
```
New-SqlColumnMasterKey "AECMK" -ColumnMasterKeySettings $CmkSettings -Path
SQLSERVER:\SQL\<your_server_name>\DEFAULT\Databases\<your_database_name>
```

> **NOTE:** Replace the <server_name> and <database_name> with actual values in your environment.



```
PS C:\> $CmkSettings = New-SqlCngColumnMasterKeySettings -CngProviderName "SafeNet Key Storage Provider" -KeyName AECMK
PS C:\> New-SqlColumnMasterKey "AECMK" -ColumnMasterKeySettings $CmkSettings -Path SQLSERVER:\SQL\HSMNOI1INT-SQL1\DEFAULT\Databases\Test

Name
----
AECMK


PS C:\> _
```

3. Confirm the presence of the newly imported CMK using Object Explorer in SSMS.



```
Object Explorer
Connect ▾
⊟ HSMNOI1INT-SQL1 (SQL Server 14.0.1000.169 - HSMNOI1INT-SQL1\A
   ⊟ Databases
      ⊞ System Databases
      ⊞ Database Snapshots
      ⊞ DWConfiguration
      ⊞ DWDiagnostics
      ⊞ DWQueue
      ⊟ Test
         ⊞ Database Diagrams
         ⊞ Tables
         ⊞ Views
         ⊞ External Resources
         ⊞ Synonyms
         ⊞ Programmability
         ⊞ Service Broker
         ⊞ Storage
         ⊟ Security
            ⊞ Users
            ⊞ Roles
            ⊞ Schemas
            ⊞ Asymmetric Keys
            ⊞ Certificates
            ⊞ Symmetric Keys
            ⊟ Always Encrypted Keys
               ⊟ Column Master Keys
                     AECMK
               ⊞ Column Encryption Keys
            ⊞ Database Audit Specifications
            ⊞ Security Policies
   ⊞ Security
   ⊞ Server Objects
   ⊞ Replication
   ⊞ PolyBase
   ⊞ Always On High Availability
   ⊞ Management
   ⊞ Integration Services Catalogs
     SQL Server Agent (Agent XPs disabled)
   ⊞ XEvent Profiler
```

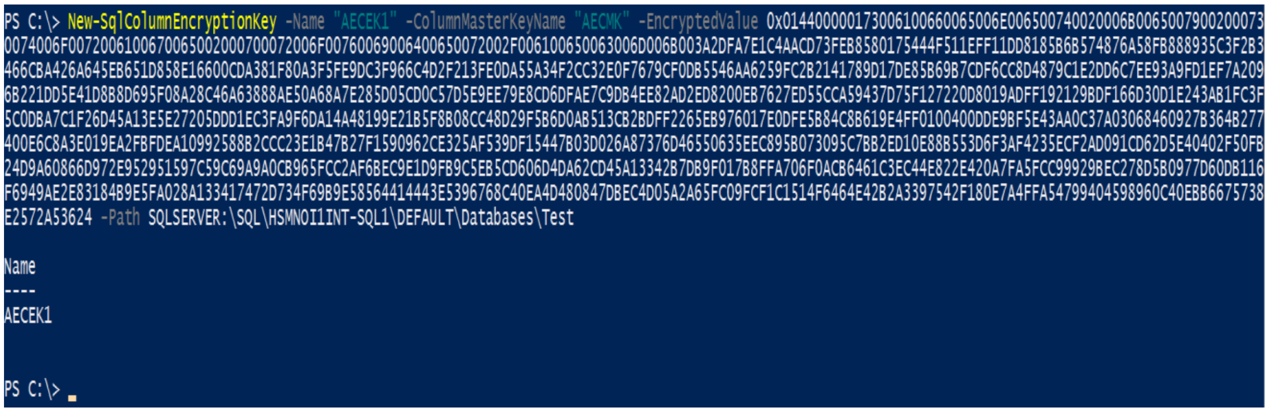
Now the **New-SqlColumnEncryptionKey** cmdlet creates a column encryption key object in the form of an "Encrypted Value" in the database. A Column Encryption Key (CEK) object is an encrypted value of the Column Encryption Key. This object can encrypt database columns using the Always Encrypted feature.

4. Run the following command to create the CEK, specifying a path to your database. 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.

```
New-SqlColumnEncryptionKey -Name "AECEK1" -ColumnMasterKeyName "AECMK" –
EncryptedValue
0x01440000017300610066006500 6E006500740020006B006500790020007300740 06F007200610
```

```
E5B84C8B619E4FF0100400DDE9BF5E43AA0C37A03068460927B364B277400E6C8A3E019EA2FBFDE
A10992588B2CCC23E1B47B27F1590962CE325AF539DF15447B03D026A87376D46550635EEC895B0
73095C7BB2ED10E88B553D6F3AF4235ECF2AD091CD62D5E40402F50FB24D9A60866D972E9529515
97C59C69A9A0CB965FCC2AF6BEC9E1D9FB9C5EB5CD606D4DA62CD45A13342B7DB9F017B8FFA706F
0ACB6461C3EC44E822E420A7FA5FCC99929BEC278D5B0977D60DB116F6949AE2E83184B9E5FA028
A133417472D734F69B9E58564414443E5396768C40EA4D480847DBEC4D05A2A65FC09FCF1C1514F
6464E42B2A3397542F180E7A4FFA54799404598960C40EBB6675738E2572A53624 -Path
SQLSERVER:\SQL\<your_server_name>\DEFAULT\Databases\<your_database_name>
```

> **NOTE:** It is advisable to use a word editor to align the encrypted value to a single line before copying to the PowerShell) and replace the <server_name> and <database_name> with actual values in your environment.



```
PS C:\> New-SqlColumnEncryptionKey -Name "AECEK1" -ColumnMasterKeyName "AECMK" -EncryptedValue 0x014400000173006100660065006E006500740020006B0065007900200073
0074006F0072006100670065002000700072006F00760069006400650072002F006100650063006D006B003A2DFA7E1C4AACD73FEB8580175444F511EFF11DD8185B6B574876A58FB888935C3F2B3
466CBA426A645EB651D858E16600CDA381F80A3F5FE9DC3F966C4D2F213FE0DA55A34F2CC32E0F7679CF0DB5546AA6259FC2B2141789D17DE85B69B7CDF6CC8D4879C1E2DD6C7EE93A9FD1EF7A209
6B221DD5E41D8B8D695F08A28C46A63888AE50A68A7E285D05CD0C57D5E9EE79E8CD6DFAE7C9DB4EE82AD2ED8200EB7627ED55CCA59437D75F127220D8019ADFF192129BDF166D30D1E243AB1FC3F
5C0DBA7C1F26D45A13E5E27205DDD1EC3FA9F6DA14A48199E21B5F8B08CC48D29F5B6D0AB513CB2BDFF2265EB976017E0DFE5B84C8B619E4FF0100400DDE9BF5E43AA0C37A03068460927B364B277
400E6C8A3E019EA2FBFDEA10992588B2CCC23E1B47B27F1590962CE325AF539DF15447B03D026A87376D46550635EEC895B073095C7BB2ED10E88B553D6F3AF4235ECF2AD091CD62D5E40402F50FB
24D9A60866D972E952951597C59C69A9A0CB965FCC2AF6BEC9E1D9FB9C5EB5CD606D4DA62CD45A13342B7DB9F017B8FFA706F0ACB6461C3EC44E822E420A7FA5FCC99929BEC278D5B0977D60DB116
F6949AE2E83184B9E5FA028A133417472D734F69B9E58564414443E5396768C40EA4D480847DBEC4D05A2A65FC09FCF1C1514F6464E42B2A3397542F180E7A4FFA54799404598960C40EBB6675738
E2572A53624 -Path SQLSERVER:\SQL\HSMNOI1INT-SQL1\DEFAULT\Databases\Test

Name
----
AECEK1


PS C:\> _
```

The screenshot above shows the output confirming the creation of a new CEK, called **AECEK1**. This CEK was generated and then encrypted using the encapsulated CMK metadata provided to the DBA by the Security Administrator.

5. Confirm the presence of the newly imported CEK using Object Explorer in SSMS.

This concludes role separation activities performed by the Database Administrator. The Security Administrator can now connect to the database and use the provisioned key metadata to encrypt the desired columns using Always Encrypted.

To encrypt a column using SSMS, refer to the “Implementing Always Encrypted using SSMS” section of this guide. To encrypt columns using PowerShell, refer to the Encrypt columns using PowerShell as Security Administrator section below.

Encrypt columns using PowerShell as Security Administrator

1. Connect to database through the SQL Server Management Studio or PowerShell.
2. If you have not created a table, create the table **Employee** with some fields in database **Test**.

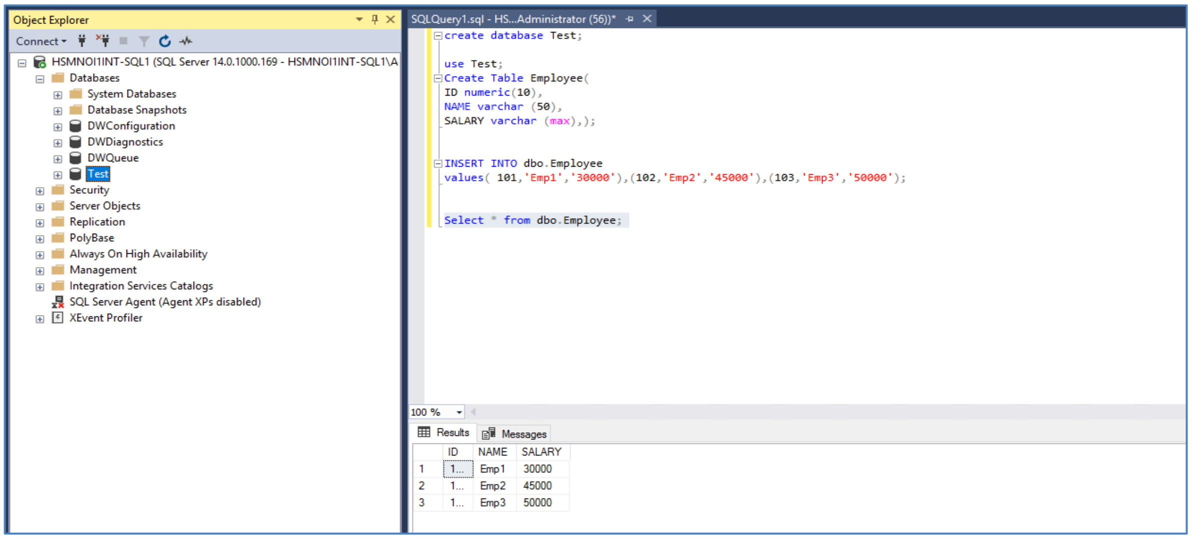
```
use Test;

Create Table Employee(

ID numeric(10),

NAME varchar (50),

SALARY varchar (max),);
```

3. Insert some values into the table.

   ```
   INSERT INTO dbo.Employee
   values( 101,'Emp1','30000'),(102,'Emp2','45000'),(103,'Emp3','50000');
   ```

4. View the table contents in plaintext.

   ```
   Select * from dbo.Employee;
   ```


Object Explorer
Connect ▾
HSMNOI1INT-SQL1 (SQL Server 14.0.1000.169 - HSMNOI1INT-SQL1\A
Databases
System Databases
Database Snapshots
DWConfiguration
DWDiagnostics
DWQueue
Test
Security
Server Objects
Replication
PolyBase
Always On High Availability
Management
Integration Services Catalogs
SQL Server Agent (Agent XPs disabled)
XEvent Profiler
SQLQuery1.sql - HS...Administrator (56))*
100 %
Results
Messages

```
create database Test;

use Test;
Create Table Employee(
ID numeric(10),
NAME varchar (50),
SALARY varchar (max),);


INSERT INTO dbo.Employee
values( 101,'Emp1','30000'),(102,'Emp2','45000'),(103,'Emp3','50000');


Select * from dbo.Employee;
```

	ID	NAME	SALARY
1	1...	Emp1	30000
2	1...	Emp2	45000
3	1...	Emp3	50000

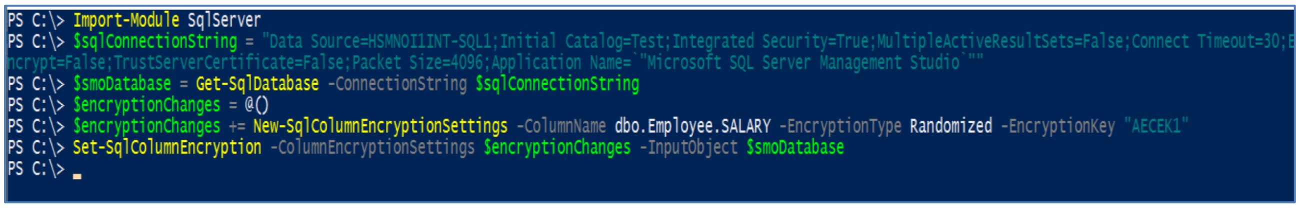
5. Open a PowerShell session with elevated permissions (right click and select “Run as Administrator”) and run the following script to encrypt a given column in the specified database. Adjust the values highlighted in bold to those suitable for your database name and data columns that you want to encrypt.

   EncryptionType values can be one of the following:

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

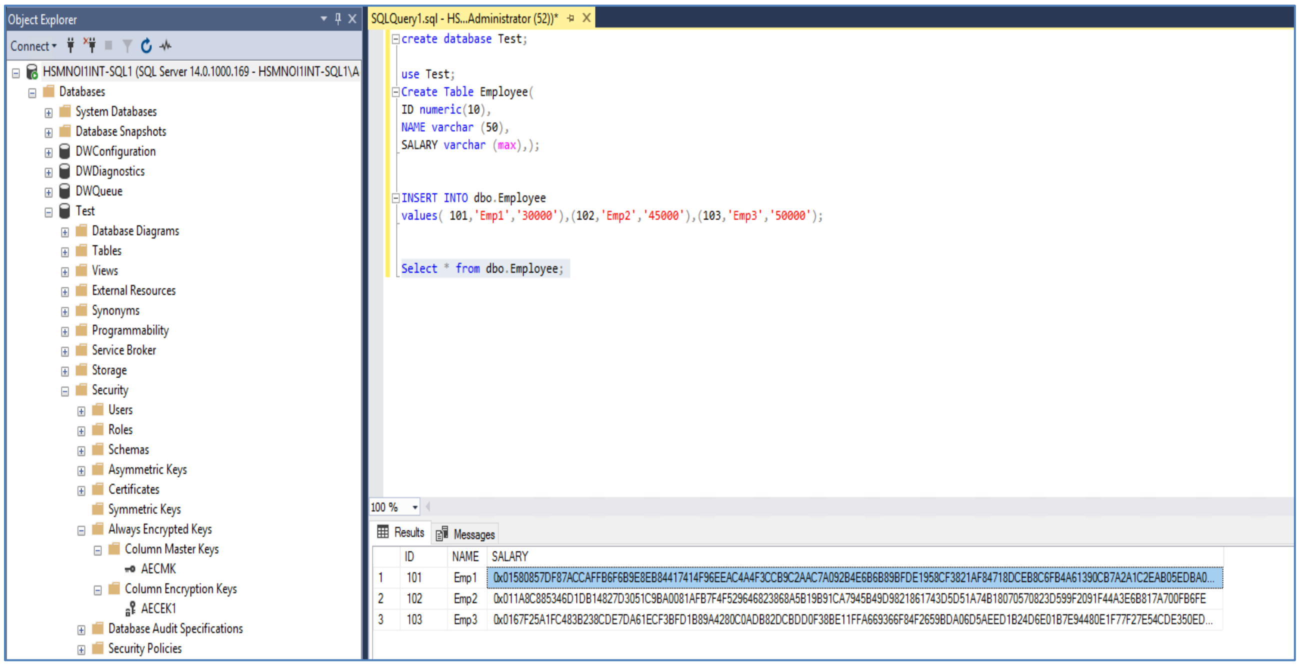
   ```
   --------------------------------------------------------------------------------

   #Import Module SqlServer

   Import-Module SqlServer

   #Set up connection and database SMO objects

   $sqlConnectionString = "Data Source=server_name;Initial
   Catalog=database_name;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].[Employee]

$encryptionChanges += New-SqlColumnEncryptionSettings -ColumnName
dbo.Employee.SALARY -EncryptionType Randomized -EncryptionKey "AECEK1"

Set-SqlColumnEncryption -ColumnEncryptionSettings $encryptionChanges -
InputObject $smoDatabase

-------------------------------------------------------------------------------
```



```
PS C:\> Import-Module SqlServer
PS C:\> $sqlConnectionString = "Data Source=HSMNOI1INT-SQL1;Initial Catalog=Test;Integrated Security=True;MultipleActiveResultSets=False;Connect Timeout=30;E
ncrypt=False;TrustServerCertificate=False;Packet Size=4096;Application Name=`"Microsoft SQL Server Management Studio`""
PS C:\> $smoDatabase = Get-SqlDatabase -ConnectionString $sqlConnectionString
PS C:\> $encryptionChanges = @()
PS C:\> $encryptionChanges += New-SqlColumnEncryptionSettings -ColumnName dbo.Employee.SALARY -EncryptionType Randomized -EncryptionKey "AECEK1"
PS C:\> Set-SqlColumnEncryption -ColumnEncryptionSettings $encryptionChanges -InputObject $smoDatabase
PS C:\> _
```

6. Verify that the column has been encrypted.



```
create database Test;

use Test;
Create Table Employee(
ID numeric(10),
NAME varchar (50),
SALARY varchar (max),);

INSERT INTO dbo.Employee
values( 101,'Emp1','30000'),(102,'Emp2','45000'),(103,'Emp3','50000');

Select * from dbo.Employee;
```

```
   ID    NAME   SALARY
1  101   Emp1   0x01580857DF87ACCAFFB6F6B9E8EB84417414F96EEAC4A4F3CCB9C2AAC7A092B4E6B6B89BFDE1958CF3821AF84718DCEB8C6FB4A61390CB7A2A1C2EAB05EDBA0...
2  102   Emp2   0x011A8C885346D1DB14827D3051C9BA0081AFB7F4F529646823868A5B19B91CA7945B49D9821861743D5D51A74B18070570823D599F2091F44A3E6B817A700FB6FE
3  103   Emp3   0x0167F25A1FC483B238CDE7DA61ECF3BFD1B89A4280C0ADB82DCBDD0F38BE11FFA669366F84F2659BDA06D5AEED1B24D6E01B7E94480E1F77F27E54CDE350ED...
```

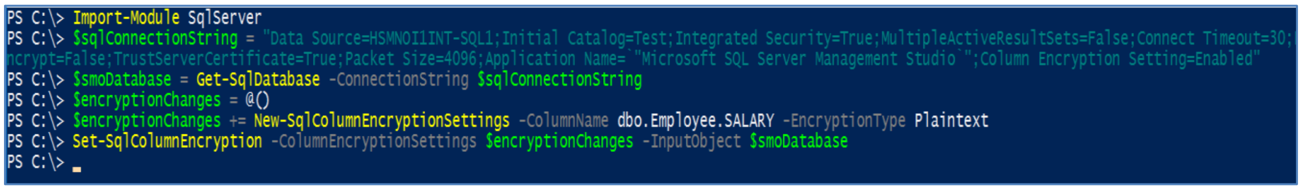
To view the encrypted columns in plain text, refer the "Viewing Always Encrypted Data" section of this guide.

Remove Always Encrypted Column Encryption

To remove column encryption from previously encrypted column data, replace the actual values of highlighted in bold and **-EncryptionType** value with the string **Plaintext**, and execute the following script.

```
-------------------------------------------------------------------------------

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=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.Employee.SALARY -EncryptionType Plaintext

Set-SqlColumnEncryption –ColumnEncryptionSettings $encryptionChanges –InputObject
$smoDatabase
---------------------------------------------------------------------------------
```



```
PS C:\> Import-Module SqlServer
PS C:\> $sqlConnectionString = "Data Source=HSMNOI1INT-SQL1;Initial Catalog=Test;Integrated Security=True;MultipleActiveResultSets=False;Connect Timeout=30;
ncrypt=False;TrustServerCertificate=True;Packet Size=4096;Application Name=`"Microsoft SQL Server Management Studio`";Column Encryption Setting=Enabled"
PS C:\> $smoDatabase = Get-SqlDatabase -ConnectionString $sqlConnectionString
PS C:\> $encryptionChanges = @()
PS C:\> $encryptionChanges += New-SqlColumnEncryptionSettings -ColumnName dbo.Employee.SALARY -EncryptionType Plaintext
PS C:\> Set-SqlColumnEncryption -ColumnEncryptionSettings $encryptionChanges -InputObject $smoDatabase
PS C:\> _
```

The Always 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, 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 before you can drop the Column Master Key.

This completes the configuration of SQL Server Always Encrypted. The column master key is secured in a Luna HSM or Cloud HSM service and the column encryption key is encrypted using the securely stored master key.

Troubleshooting Tips

Problem – 1

Failed to verify Authenticode signature on DLL `"C:\Program Files\LunaPCI\EKM\LunaEKM.dll".`

Solution

This error could appear in SQL logs if the certificate in the signature of dll cannot be verified because there are no corresponding certificates for this issuer and therefore it is not trusted.

Go to http://www.verisign.com/support/roots.html and download the all root certificates. Install the certificate and install/import it to Trusted Root Certification Authorities store.

Problem – 2

`CREATE CRYPTOGRAPHIC PROVIDER EKMProvider FROM FILE = <Path to EKM DLL>'` fails with below error on Windows 2012:

```
Error:
Msg 33029, Level 16, State 1, Line 3
Cannot initialize cryptographic provider.  Provider error code: 1. (Failure -
Consult EKM Provider for details)
```

Solution

Reboot the OS server and try to create cryptographic provider.

Problem – 3

Unable to open Symmetric key which is encrypted by Asymmetric Key in Microsoft SQL Server 2017.

```
Error:
Msg 15466, Level 16, State 28, Line 1
An error occurred during decryption.
```

Solution

Download the cumulative update package and apply for SQL Server provided by Microsoft to resolve the issue:

https://support.microsoft.com/en-us/help/4342123/cumulative-update-10-for-sql-server-2017

Contacting Customer Support

If you encounter a problem during this integration, contact your supplier or Thales Customer Support. Thales Customer Support operates 24 hours a day, 7 days a week. Your level of access to this service is governed by the support plan arrangements made between Thales and your organization. Please consult this support plan for further information about your entitlements, including the hours when telephone support is available to you.

Customer Support Portal

The Customer Support Portal, at https://supportportal.thalesgroup.com, is a database where you can find solutions for most common problems. The Customer Support Portal is a comprehensive, fully searchable repository of support resources, including software and firmware downloads, release notes listing known problems and workarounds, a knowledge base, FAQs, product documentation, technical notes, and more. You can also use the portal to create and manage support cases.

NOTE: You require an account to access the Customer Support Portal. To create a new account, go to the portal and click on the **REGISTER** link.

Telephone Support

If you have an urgent problem, or cannot access the Customer Support Portal, you can contact Thales Customer Support by telephone at +1 410-931-7520. Additional local telephone support numbers are listed on the support portal.

Email Support

You can also contact technical support by email at technical.support.DIS@thalesgroup.com.