Microsoft

SQL Server EKM Provider

v1.5

Integration Guide

u.trust GP HSM Se-Series

6.0.0 and 6.1.1

utimaco®

Imprint

Copyright 2025	Utimaco IS GmbH
	Germanusstr. 4
	D-52080 Aachen
	Germany
Phone	AMERICAS +1-844-UTIMACO (+1 844-884-6226)
	EMEA +49 800-627-3081
	APAC +81 800-919-1301
Internet	https://support.hsm.utimaco.com/
e-mail	support@utimaco.com
Document Version	0.1.0
Date	2025-10-07
Status	PUBLISHED
Document No.	IG-2025-0030

Table of Contents

1 Introduction

This guide is part of the information and support provided by Utimaco. Additional documentation produced to support your Utimaco u.trust product can be found in the document directory of the Utimaco u.trust product bundle. All Utimaco u.trust product documentation is available from Utimaco's web site at https://utimaco.com/.

1.1 About This Guide

This guide describes how to set up and use the Utimaco SecurityServer EKM provider. The Microsoft SQL Server provides data encryption, decryption, and key management capabilities. Together with the Extensible Key Management (EKM), the management of encryption keys for data and key encryption is very easy. This enables Microsoft SQL Server to access the advanced encryption and protection features of the Utimaco u.trust GP HSM device.

1.2 Target Audience for This Guide

This guide is intended for administrators of Microsoft SQL Server and Utimaco HSMs.

1.3 Document Conventions

The following conventions are used in this guide:

Convention	***Use***	***Example***
Bold	Items of the Graphical User Interface (GUI), e.g., menu options	Press the **OK** button.
`Monospaced`	File names, folder and directory names, commands, file outputs, programming code samples	You will find the file `example.conf` in the `/exmp/demo/` directory.

Convention	*Use*	*Example*
Italic	References and important terms	See Chapter 3, "Sample Chapter", in the *u.trust Anchor - csadm Manual* or [CSADM].

Table 1: Document conventions

Special icons are used to highlight the most important notes and information.

This message marks the result expected after the successful execution of an instruction

Here you find additional notes or supplementary information.

Here you find important safety information that should be followed.

1.4 Abbreviations

Abbreviation	*Meaning*
AES	Advanced Encryption Standard
CA	Certificate Authority
CAT	CryptoServer Administration Tool
CNG	Cryptography API Next Generation

Abbreviation	*Meaning*
CXI	Cryptographic eXtended Interface
DES	Data Encryption Standard
DHCP	Dynamic Host Configuration Protocol
EKM	Extensible Key Management
FIPS	Federal Information Processing Standards
GUI	Graphical User Interface
HSM	Hardware Security Module
KEK	Key Encryption Key
MBK	Master Backup Key
RSA	Rivest-Shamir-Adleman
SSMS	SQL Server Management Studio
SQL	Structured Query Language
TDE	Transparent Data Encryption

Table 2: List of Abbreviations

2 Overview

2.1 Microsoft SQL Server EKM Provider

Microsoft SQL Server provides different types of encryption to help protect data. The data encrypted in the traditional key hierarchy is done using a symmetric data encryption key (DEK). In this model, the symmetric data encryption key is additionally protected by encrypting it with a hierarchy of keys stored in the SQL Server.

An alternative to this model is to use the Extensible Key Management (EKM) provider. The Microsoft SQL Server EKM Provider enables external (third-party) EKM/HSM vendors to integrate their modules into the Microsoft SQL Server. After integration, SQL Server users can use the encryption keys stored on EKM modules. This model adds an additional layer of security and separates the management of keys and data.

2.2 Utimaco u.trust GP HSM

CryptoServer is a hardware security module developed by Utimaco IS GmbH. CryptoServer is a physically protected specialized computer unit designed to perform sensitive cryptographic tasks and to securely manage as well as store cryptographic keys and data. It can be used as a universal, independent security component for heterogeneous computer systems.

CryptoServer is the hardware. SecurityServer is the firmware package and is also the host side software package that contains the EKM provider.

3 Integration Requirements and Prerequisites

Ensure that the system environment you will be using meets the following hardware and software requirements.

3.1 Tested Versions

The integrations that have been successfully tested with different versions of Microsoft Windows Server, Microsoft SQL Server, Utimaco CryptoServer product and Utimaco HSMs are shown in the following configurations below:

Microsoft Windows Server	*Microsoft SQL Server*	*Utimaco Security Server Version*	*Utimaco HSM*
Windows Server 2019 Windows Server 2016	SQL Server 2019 SQL Server 2017 SQL Server 2016	SecurityServer 4.45.1 or higher	CryptoServer CSe-Series/Se-Series

Table 3: List of Tested Versions

3.2 Hardware and Software Requirements

Software	*Software Requirements*
Java	Version 8, Update 271 or higher
HSM Interfaces	SecurityServer EKM provider

Table 4: List of Software Requirements

Hardware	*Hardware Requirements*

Utimaco LAN HSM	CryptoServer CSe-Series/Se-Series LAN with firmware SecurityServer 4.45.1 or higher
Utimaco PCI-e HSM	CryptoServer CSe-Series/Se-Series PCI-e with firmware SecurityServer 4.45.1 or higher

Table 5: List of Hardware Requirements

Setup an account on the Utimaco support portal and request download access at the following URL.

https://support.hsm.utimaco.com/

3.3 Prerequisites

Before you begin, please ensure that you have installed/setup:

- CryptoServer is setup and configured. Refer the CryptoServer documentations to setup the HSM
- MBK must be created and stored onto each HSM. Refer the CryptoServer documentations to setup the MBK
- CryptoServer Default Admin should be replaced with a new admin user
- Operating system listed in Tested Versions
- SQL Server listed in Tested Versions
- SQL Server Management Studio
- SecurityServer listed in Tested Versions with SecurityServer EKM provider
- A cryptographic user on that SecurityServer

You should also be familiar with SQL statements, as this guide makes intensive use of them.

After the successful SecurityServer setup, you should find these files on your system for use with the SecurityServer EKM provider:

`cssqlekm.dll` and `cssqlekmlib.dll`

The former is the provider library that will be loaded into SQL Server, and the latter is required by it. These files are located in C:\Program Files\Utimaco\SecurityServer\Lib\ which must be in system PATH.

`cssqlekm.cfg`

This file contains the parameters that the SecurityServer EKM provider will use when communicating with the HSM. Please see the next sections for details.

4 Software Download and Installation

This section describes the process of installing Utimaco HSM software with the EKM provider for Microsoft SQL Server.

4.1 Download Utimaco Software

If you have not already done so, please create and request an Utimaco Support Portal Account. This will allow you to download the software components needed for this installation.

If you have purchased an HSM from Utimaco, locate the included product bundle, which contains the Windows software packages.

Install the latest version of the SecurityServer software as described in the SecurityServer Manual for the HSM. Restart the Microsoft SQL service.

4.2 Configuration File

This section describes the setup steps for Microsoft SQL Server 2022, which is required for integration with the Utimaco SecurityServer EKM provider. Please ensure that all hardware and software prerequisites mentioned in the previous sections are fulfilled before proceeding.

Within the u.trust Anchor product bundle, you should find these files for use with the SecurityServer EKM provider under the path `.\u.trust_anchor_product_bundle-x.x\Software\Windows\Crypto_APIs\EKM`:

- `\lib\cssqlekm.dll`
- `\lib\cssqlekmlib.dll`

The former is the provider library that will be loaded into SQL Server, and the latter is required by it. These files are located in `C:\Program Files\Utimaco\SecurityServer\Lib\`, which must be in the system PATH.

The `C:\Program Files\Utimaco\SecurityServer\Lib\` is not present in the system PATH by default; it must be included manually.

- `cssqlekm.cfg`

This file contains the parameters that the SecurityServer EKM provider will use when communicating with the HSM. Please see the next sections for details.

4.2.1 Location of the Configuration File

The installation wizard copies a sample configuration file to `C:\ProgramData\Utimaco\EKM\cssqlekm.cfg`. Please see the next section on how to customize the file.

The installation wizard also adds the ConfigPath value to the Windows registry key `HKEY_LOCAL_MACHINE\SOFTWARE\Utimaco\EKM` containing the location of the configuration file.

Alternatively, it is possible to register the location of the configuration file using an environment variable, which takes precedence over the registry setting: Open the Control Panel, System, click on Advanced system settings and click on the Environment Variables. Create a new variable EKMCONFIGPATH in System variables. Add the path of `cssqlekm.cfg`.

4.2.2 Customization of the Configuration File

The SecurityServer EKM provider can be customized using the configuration file. Edit the configuration file to your needs. At least, change the Device setting. Make sure the Microsoft SQL Server service account has write access to the folder containing the external keystore and the log file.

Parameter	*Description*
Connection Timeout	Specifies the maximum time in milliseconds to wait before the connection establishment is aborted if the device is not responding.
Device	Specifies the device address of the SecurityServer device. This can be a local PCI-e card (PCI:0) or a network address ([port@]IP).
KeysExternal	To enable external keystore

KeyStorage Type	Database type for keystorage
KeyStorage Config	Configuration of the KeyStorage
LogFile	Specifies the path and the name of the log file.
LogLevel	Specifies the log level. Higher levels include the information of the lower levels. 0=no log, 1=errors, 2=warnings, 3=info, 4=trace, 5=debug.
LogSize	Defines the maximum size of the log file. If the maximum is reached, the old log file will be renamed to .bak and a new log file with the name defined by LogFile is created.
Timeout	Specifies the maximum time in milliseconds to wait for the answer from SecurityServer after sending a command.

Table 6: List of Configuration Parameter

4.3 Enable Extensible Key Management

The EKM provider enabled option controls Extensible Key Management device support in Microsoft SQL Server. This option is disabled by default and needs to be enabled to use any EKM provider. Connect to your SQL Server instance and login with administrative privileges. Open a query window for further execution of SQL statements. To enable Extensible Key Management, please run the following SQL.

SQL Statement

```
sp_configure 'show advanced', 1 GO

RECONFIGURE GO

sp_configure 'EKM provider enabled', 1 GO

RECONFIGURE
```

4.4 Register Provider

Run the following SQL to register the provider under the name utimaco.

? Unknown Attachment SQL Statement

```
CREATE CRYPTOGRAPHIC PROVIDER utimaco

FROM FILE = 'C:\Program Files\Utimaco\SecurityServer\Lib\cssqlekm.dll'
```

To verify the successful installation, run:

? Unknown Attachment SQL Statement

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

4.4.1 Alter Provider Location

To change the location of the provider, run the following at any time.

SQL Statement

```
ALTER CRYPTOGRAPHIC PROVIDER utimaco

FROM FILE = '<path-to-new-provider-dll>'
```

Since SQL Server stores the version of an EKM provider with the registration, it is also necessary to run this command after updating the provider.

4.4.2 Remove Provider

To remove the provider entirely:

1. Close all opened sessions that use the provider
2. Remove all credentials regarding the provider
3. Run the following SQL statement

SQL Statement

```
DROP CRYPTOGRAPHIC PROVIDER utimaco
```

4.5 Setting Up Credentials

The SecurityServer EKM provider exposes basic authentication to the SQL Server using username/-password pairs. These pairs are stored in so-called credentials that need to be created per EKM provider. Finally, a credential is mapped to an SQL Server login.

If a logged in user wants to access a certain EKM provider, the credential mapped to both the login and the EKM provider is looked up and the username/password is passed to the EKM provider. The SecurityServer EKM provider uses this information to perform login on the SecurityServer.

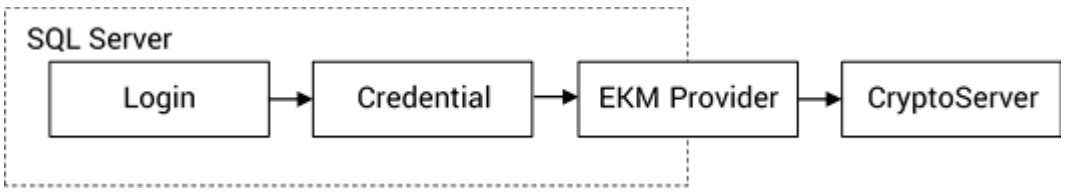

Figure 1: Credential Mapping

The same credential can be used for multiple SQL Server logins. Also, a login can be used with multiple credentials if the EKM providers are different. Otherwise, the lookup shown before will fail.

The following SQL will create a credential csekm for the SecurityServer user sqlekm with the password utimaco.

? Unknown Attachment SQL Statement

```
CREATE CREDENTIAL csekm WITH IDENTITY = 'sqlekm', SECRET = 'utimaco' FOR
 CRYPTOGRAPHIC PROVIDER utimaco
```

Creating a CryptoServer user need to be done either via csadm or CAT.

For detailed information refer to chapter 4.45.1 of the CryptoServer Manual Systemadministrator.

Use the following SQL statement to map the credential to any SQL Server account. You can for example substitute <user> with an integrated account like sa or a Windows account like [DB1\Administrator].

? Unknown Attachment SQL Statement

```
ALTER LOGIN <user> ADD CREDENTIAL csekm
```

4.5.1 Improving Security via CXI Group

By default, new EKM keys are generated without CXI group. The SecurityServer user does not need to have a CXI_GROUP attribute, but every cryptographic user on the SecurityServer can access the keys in the EKM keystore file. To provide better protection, a CXI group should be defined in the SQL Server credential's identity:

? Unknown Attachment SQL Statement

```
CREATE CREDENTIAL csekm WITH IDENTITY = 'sqlekm@ekmgroup', SECRET = 'utimaco'
FOR CRYPTOGRAPHIC PROVIDER utimaco
```

Now, new EKM keys are created in the CXI group ekmgroup and only SecurityServer users belonging to this group can access these keys. Therefore, the SecurityServer user sqlekm needs to be member of the CXI group ekmgroup by setting its CXI_GROUP attribute to ekmgroup on user creation.

Since key names (more specifically the PROVIDER_KEY_NAME) have to be unique per CXI group only, the use of different CXI groups for different credentials also prevents name collisions when SQLEKM is used with different databases from the same SQL Server.

4.5.2 Cryptographic User Hierarchy

The SecurityServer EKM provider also supports hierarchical users via wildcards in the CXI_GROUP user attribute. Imagine the following SQL Server credentials with their identities and the CXI_GROUP attribute of the matching SecurityServer users:

Credential	*Identity*	*CXI_GROUP User Attribute*
EKMuser1	EKMuser1@ekmgrp1	ekmgrp1
EKMuser2	EKMuser2@ekmgrp2	ekmgrp2
EKMadmin	EKMadmin@ekmgrp1	ekmgrp*

Table 7: List of Credential & Identity

An SQL Server account bound to credential EKMuser1 is logged into SecurityServer as EKMuser1. New keys are generated in CXI group ekmgrp1 and only keys in that group can be accessed. Similarly, an SQL Server account bound to credential EKMuser2 is logged in as user EKMuser2, and new keys are generated in CXI group ekmgrp2. Unsurprisingly, an account bound

to credential EKMadmin is logged into SecurityServer as EKMadmin. New keys are now generated in CXI group ekmgrp1 since this value is taken from the credential's identity.

However, this user can also use keys generated by user EKMuser2 in group ekmgrp2 since the CXI_GROUP attribute grants access to these keys as well. This works since SQL Server commands refer to a key by an SQL Server name, which is bound internally to an EKM provider key name defined in the CREATE … KEY statement, and the actual access is done using an identifier stored together with the SQL Server name. This identifier is also used when the key is deleted inside the provider.

Note that the CXI group from the credential's identity is also used when a key inside the SecurityServer is searched by name. This happens when a new SQL Server key is created from an existing SecurityServer key. Currently, supplying a different CXI group in the `CREATE… KEY` statement than the one given in the credential is not supported. Moreover, there would be no way to explicitly specify the CXI group when viewing all the keys from the EKM provider.

5 Using the Provider

SQL Server can create and store keys internally, protected by software only. With Extensible Key Management (EKM), SQL Server can use keys protected by an HSM for data and key encryption/ decryption.

The SecurityServer EKM provider offers EKM functionality for Utimaco SecurityServer HSMs, supporting different symmetric and asymmetric algorithms. The location of the database is defined in the configuration file.

5.1 Creating Keys

To create a new symmetric AES 256 key EKM_AES_256 and store it in the SecurityServer EKM-provider, use the following statement:

? Unknown Attachment SQL Statement

```
CREATE SYMMETRIC KEY EKM_AES_256

FROM PROVIDER utimaco WITH ALGORITHM = AES_256,

PROVIDER_KEY_NAME = 'EKM_AES_256', CREATION_DISPOSITION=CREATE_NEW
```

Note that the key name EKM_AES_256 appears twice here: first as key name for the SQL Server and second as the SecurityServer key name. However, it is not necessary that both names are identical. In fact, in SQL Server commands a key is referred to by its SQL Server

name. The CREATE … KEY statement creates a binding to the SecurityServer key, which can

be different, using a common identifier.

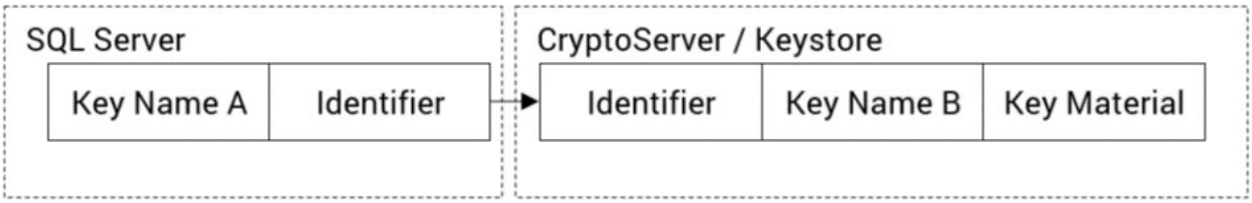

Figure 2: Key Mapping

The SQL Server key can also be created from an existing SecurityServer EKM provider key:

? Unknown Attachment SQL Statement

```
cxitool Dev=<port@IP> LogonPass=<user>,<password> Group="" Name=<key_name>
Spec=0 Usage=… GenerateKey=<key_type>,<key_size>
```

For AES keys use `key_type AES` *and* `key_size 256`.

For RSA keys use `key_type RSA` *and* `key_size 2048/3072/4096`.

To create key in the external keystore add the keystoretype, keystoreparam parameters

before GenerateKey: …

For Example, create an AES 256 key with the name `AEKM_AES_256` in external keystore.

? Unknown Attachment SQL Statement

```
cxitool Dev=3001@127.0.0.1 LogonPass=ekmuser1,123456 keystoretype=SDB
keystoreparam="C:\ProgramData\Utimaco\EKM\cssqlekm.sdb" group="" Name=
EKM_AES_256 Usage=ENCRYPT,DECRYPT,SIGN,VERIFY spec=0 generatekey=AES,256
```

? Unknown Attachment SQL Statement

```
CREATE SYMMETRIC KEY AEKM_AES_256

FROM PROVIDER utimaco

WITH PROVIDER_KEY_NAME = 'OtherAesKey', CREATION_DISPOSITION=OPEN_EXISTING
```

Here, a lookup for the given provider's key name is performed. For the SecurityServer EKM provider, the "CXI_GROUP" is also considered if one is specified in the credential's identity. This statement creates the aforementioned binding.

To create asymmetric keys, proceed in the same manner. Here is the statement to create an asymmetric RSA 2048 key:

? Unknown Attachment SQL Statement

```
CREATE ASYMMETRIC KEY EKM_RSA_2048

FROM PROVIDER utimaco

WITH ALGORITHM = RSA_2048, PROVIDER_KEY_NAME = 'EKM_RSA_2048',
CREATION_DISPOSITION=CREATE_NEW
```

5.2 Viewing Keys

The SQL Server provides several SQL statements to list existing cryptographic keys. The first set of statements is used to show the SQL Server keys, both keys stored internally and keys with references to EKM providers. This key listing is separated into symmetric and asymmetric listings.

Use the next SQL statement to show current symmetric keys:

SQL Statement

```
SELECT * from master.sys.symmetric_keys
```

This will show a listing of current asymmetric keys:

SQL Statement

```
SELECT * from master.sys.asymmetric_keys
```

Remember that only these keys can be used in SQL statements.

Additionally, all keys stored in the SecurityServer EKM provider can be listed with an extra SQL statement. These keys can be used in a CREATE … KEY statement as PROVIDER_KEY_NAME to create an SQL Server key binding.

SQL Statement

```
SELECT * FROM sys.dm_cryptographic_provider_keys(65536)
```

Note that the number of shown keys can differ between previous mentioned SQL statements since not all keys stored in SecurityServer EKM provider necessarily have an equivalent key in the SQL Server space and vice versa.

5.3 Deleting Keys

To delete a symmetric key in SQL Server, use this SQL statement:

? Unknown Attachment SQL Statement

```
DROP SYMMETRIC KEY <key name>
```

For asymmetric keys use ASYMMETRIC instead of SYMMETRIC in the SQL statement:

? Unknown Attachment SQL Statement

```
DROP ASYMMETRIC KEY <key name>
```

With the previous statements an internal SQL Server key or a binding to a key in an EKM provider is deleted. In the latter case, the key itself is still existing in the EKM provider. To delete both the binding and the EKM provider key use the following statement for a symmetric key:

? Unknown Attachment SQL Statement

```
DROP SYMMETRIC <key name> REMOVE PROVIDER KEY
```

For Example

? Unknown Attachment SQL Statement

```
DROP SYMMETRIC KEY EKM_AES_256 REMOVE PROVIDER KEY
```

Use this SQL statement to delete an asymmetric key.

? Unknown Attachment SQL Statement

```
DROP ASYMMETRIC KEY <key name> REMOVE PROVIDER KEY
```

For Example

? Unknown Attachment SQL Statement

```
DROP ASYMMETRIC KEY EKM_RSA_2048 REMOVE PROVIDER KEY
```

6 Column Level Encryption

An EKM provider can be used for column-level encryption and decryption. This chapter shows how to use the SecurityServer EKM provider as a column-level encryption and decryption engine.

1. To demonstrate encryption and decryption a table demo will be created first. Consider here that for cryptographic keys to be successfully used, they have to be generated within the same database as the table entries which you wish to encrypt.

SQL Statement

```
CREATE DATABASE utimaco GO

USE utimaco

CREATE TABLE [dbo].[demo] (

firstname varchar (255) NOT NULL, name varchar (255) NOT NULL, secret varbinary
(8000) NOT NULL

) GO

CREATE SYMMETRIC KEY CLE_AES_256

FROM PROVIDER utimaco WITH ALGORITHM = AES_256,

PROVIDER_KEY_NAME = 'CLE_AES_256', CREATION_DISPOSITION=CREATE_NEW

GO
```

2. New rows can be inserted like in the next SQL statement. This statement uses a symmetric column encryption for the column secret.

SQL Statement

```
INSERT INTO demo

VALUES ('John', 'Doe', ENCRYPTBYKEY(KEY_GUID('CLE_AES_256'), 'utimaco'))
```

3. In the same way an asymmetric encryption could be used.

SQL Statement

```
INSERT INTO demo
VALUES ('John', 'Doe', ENCRYPTBYASYMKEY(ASYMKEY_ID('CLE_RSA_2048'),
'utimaco'))
```

EncryptByAsymKey returns NULL if the input exceeds a certain number of bytes, depending on the algorithm.

The limits are:

- *a 2048 bit key can encrypt up to 245 bytes*

Encryption and decryption with an asymmetric key are very costly compared with encryption and decryption with a symmetric key.

4. To show the decrypted value of an encrypted column the next statements can be used. This decrypts the symmetric encrypted column address and shows all stored rows of this table:

SQL Statement

```
SELECT CONVERT(varchar, DECRYPTBYKEY(secret)) secret FROM demo
```

5. A decryption of asymmetric column can be achieved similar to the decryption of symmetric encrypted column:

SQL Statement

```
SELECT
CONVERT(varchar, DECRYPTBYASYMKEY(ASYMKEY_ID('CLE_RSA_2048'), secret))
secret FROM demo
```

7 Transparent Data Encryption

With the introduction of transparent data encryption (TDE) in SQL Server 2008, users now have the opportunity of full database-level encryption by using TDE. TDE is the optimal choice for bulk encryption to meet regulatory compliance or corporate data security standards. TDE works at the file level which encrypts data directly on the hard drive. TDE does not replace the column-level encryption. It is just another way of encrypting data of your database transparently. The next steps will guide you on how to enable TDE with the SecurityServer EKM provider.

1. First of all, a credential for TDE has to be created.

SQL Statement

```
CREATE CREDENTIAL tde WITH IDENTITY = 'tde', SECRET = 'utimaco' FOR
 CRYPTOGRAPHIC PROVIDER utimaco
```

2. Create an asymmetric key used as TDE KEK (Key Encryption Key) in the master database.

SQL Statement

```
USE master;

CREATE ASYMMETRIC KEY tdekey

FROM PROVIDER utimaco

WITH ALGORITHM = RSA_2048, PROVIDER_KEY_NAME = 'tdekey',
CREATION_DISPOSITION=CREATE_NEW;
```

3. Create a SQL Server login account from this asymmetric key:

SQL Statement

```
CREATE LOGIN tdelogin FROM ASYMMETRIC KEY tdekey
```

4. Link your SQL Server credential to your just created user account with the next statement:

SQL Statement

```
ALTER LOGIN tdelogin ADD CREDENTIAL tde
```

5. Switch to your database to be encrypted with TDE. In our example we create a database named demo first:

SQL Statement

```
CREATE DATABASE demo GO

USE demo
```

6. Create a database encryption key, in this example based on an AES algorithm.

SQL Statement

```
CREATE DATABASE ENCRYPTION KEY WITH ALGORITHM = AES_256

ENCRYPTION BY SERVER ASYMMETRIC KEY tdekey
```

7. Enable the transparent data encryption and start encryption of the database as a background thread. Depending on the size of the database it can take a while until the encryption has been completed.

SQL Statement

```
ALTER DATABASE demo SET ENCRYPTION ON;
```

8. To see the current state of the encryption, use the next SQL statement.

SQL Statement

```
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;
```

8 Microsoft SQL Server Clustering

Microsoft SQL Server clustering is nothing but a collection of two or more physical servers (cluster nodes), connected through the LAN, out of which each of the host in a SQL server instance and have the same access to shared storage. High availability and protection from disasters is achieved by clustering SQL servers, whenever a server hosting the SQL Server instance fails.

8.1 Installing Failover Clustering feature on the Cluster Nodes

Please execute the following steps on the cluster nodes

1. Log in to cluster node as a user with Administrative privileges
2. Select Start and then Server Manager to open Server Manager
3. Under Configure this Local Sever and click Add Roles and Features. The Add Roles and Features Wizard displays. Click Next
4. Select the radio button for Role-based or feature-based installation and click Next
5. Select the radio button for Select a server from the server pool option and from Server Pool. Select cluster node and click Next
6. Click Next
7. From the list of available features, select the Failover Clustering check box and click Next

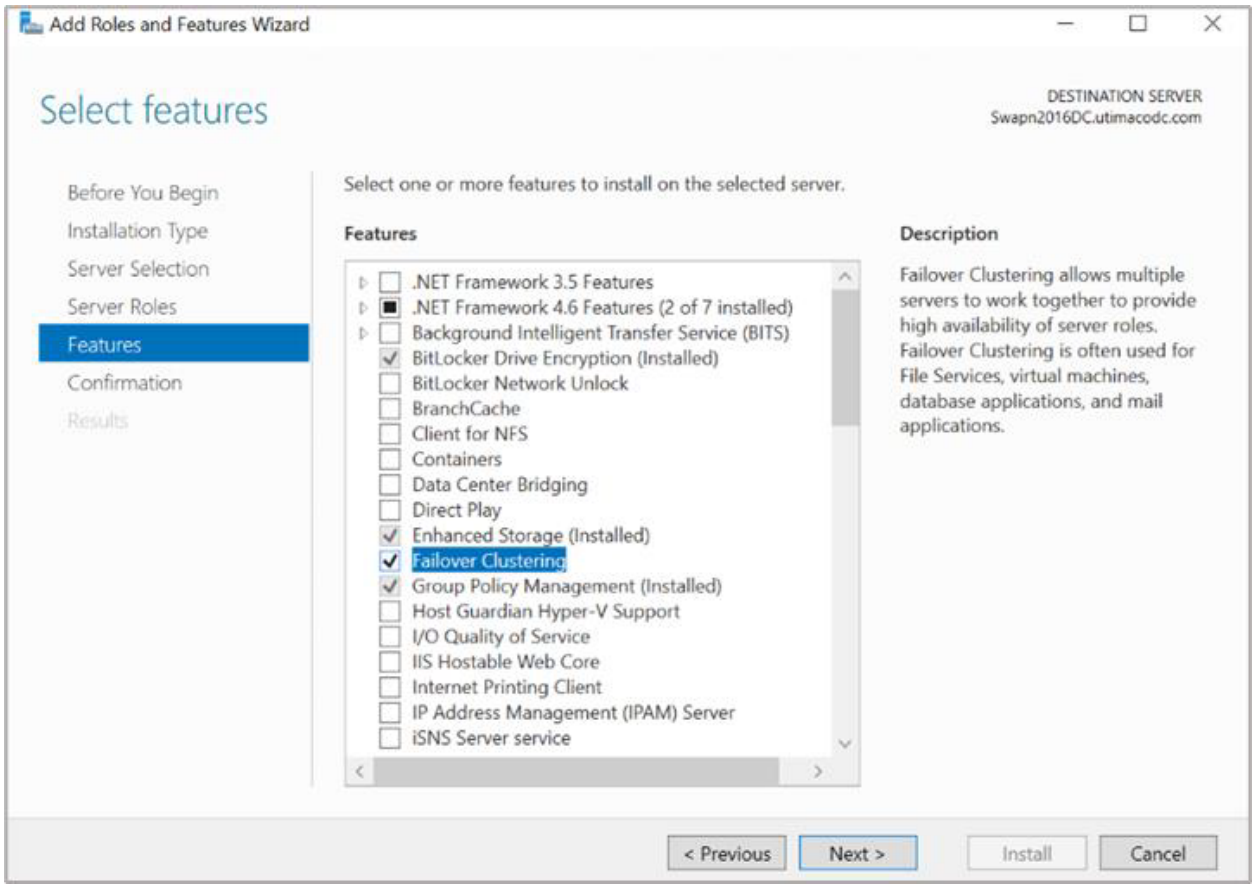

Figure 3: Select installation type window

8. A pop-up display stating Add features that are required for Failover Clustering? To add a feature, click the Add Features button
9. Click Next
10. Click Install
11. Once Feature installation is complete, click Close Similarly configure it with the available nodes in the cluster

8.2 Install and Configure MS SQL on a Failover Cluster

1. Configure a share drive (example - E: Drive) and install it on the cluster nodes
2. To configure Failover Cluster, open Failover Cluster Manager. Click on Create a Cluster from the Actions menu tab.

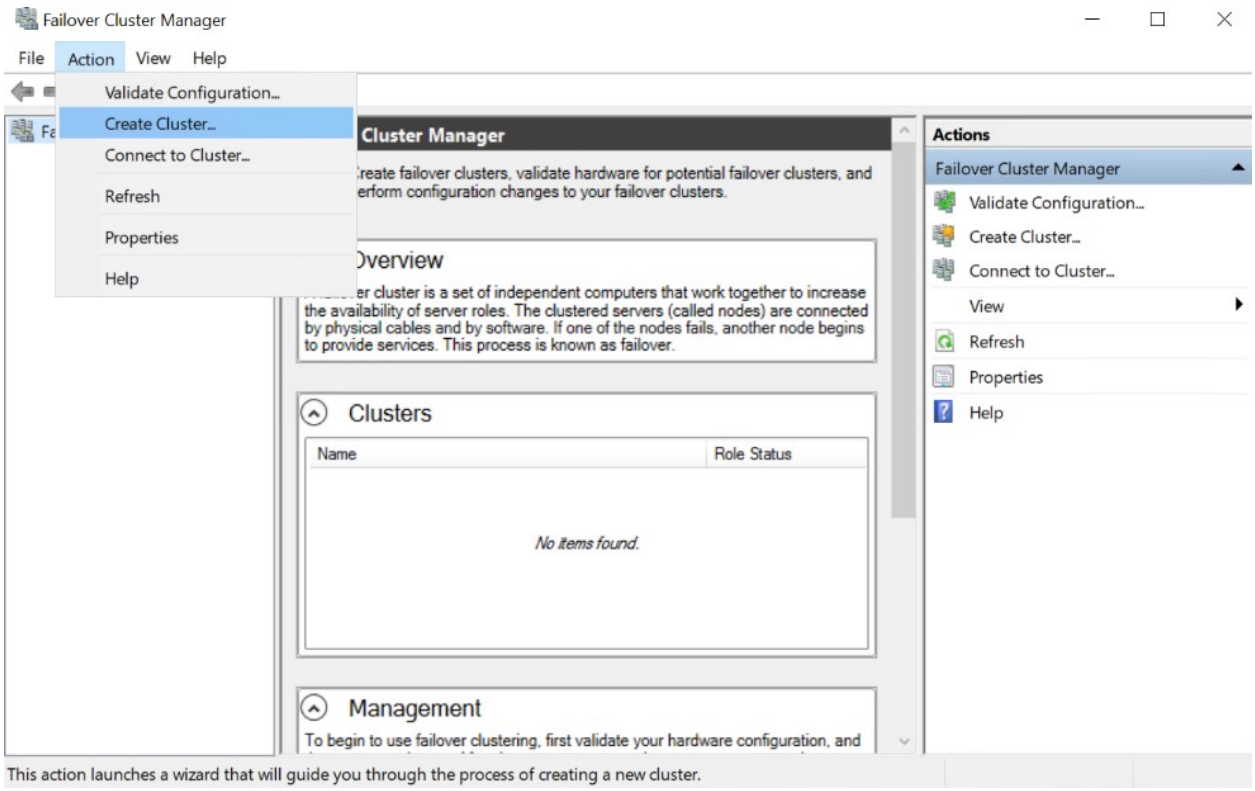

Figure 4: Failover Cluster Manager window

3. On the Before You Begin page, click Next
4. Enter the first cluster node name in the Enter Server Name field and click Add
5. Enter the second cluster node name in the Enter Server Name field and click Add
6. Click Next
7. Enter the Cluster Name and click Next until you reach the Summary page

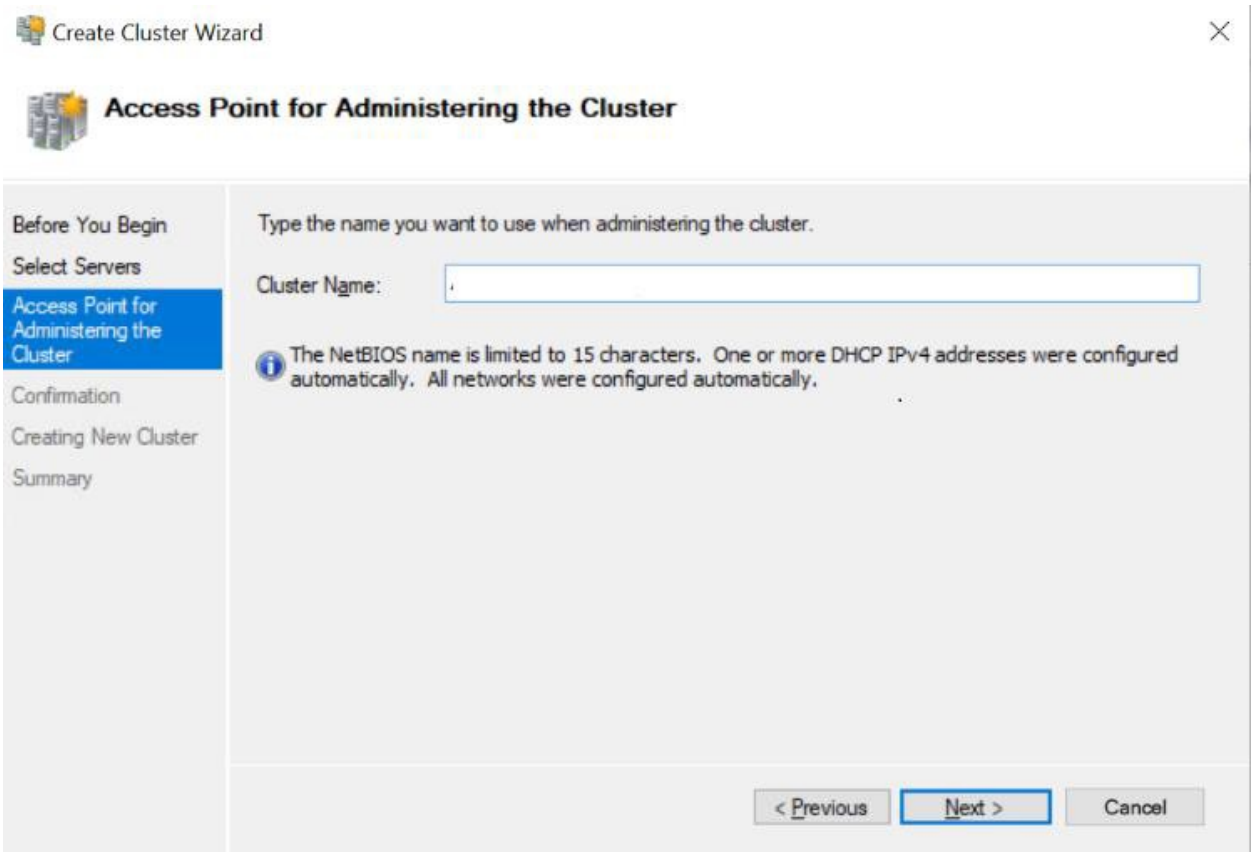

Figure 5: Create Cluster Wizard window

8. To perform the validation tests, chose Yes and click Next two times
9. Keep the default option to Run all tests and click Next two times
10. Verify the test report and click Finish
11. Provide the cluster name and click Next until you reach the Summary page
12. Verify the cluster got configured successfully. Check the status of the nodes, disk, and network. It should be in green

8.2.1 Installation of SQL Server on Cluster Nodes

1. Go to the Cluster Node, where Shared drive is mounted automatically by Failover Cluster Manager and install the SQL Server Setup
2. Double- Click on the SQL Server Setup then go to the Installation Tab and select New SQL Server Failover Cluster Installation
3. Enter the Product key, click Next

4. Select the check box for accepting the License Terms, click Next
5. Click Next till the user gets Install Failover Cluster Rules wizard
6. Verify the status for the Tests, Click Next
7. Select the appropriate checkbox required, Click Next
8. Select the appropriate Instance radio button and provide the instance name, click Next
9. Select the SQL Cluster Name from the drop-down menu, click Next
10. Select the appropriate Cluster Disk, click Next
11. Enter the IP address (DHCP/Static) for cluster network
12. Select the Admin Username enter the Password and enable the checkbox for Grant Perform Volume Maintenance Task privilege to SQL Server Database Engine Service, click Next

SQL Statement

```
USE Testdb;

CREATE ASYMMETRIC KEY 'RSA2048Key1'

FROM PROVIDER utimaco

WITH ALGORITHM = RSA_2048,
```

13. Select the Mix Mode radio button, enter the password for SQL Admin account and select the username for SQL Admin account
14. Select the Data Directories Tab and verify appropriate shared drive directory paths
15. Click Next for next two wizards, the installation for SQL Status starts
16. After Installation is complete, Click Close
17. Now perform the similar steps on the other nodes, the only change is Select Add node to a SQL Server Failover cluster from the SQL Server Setup

8.3 Verify Failover Cluster Configuration

1. Open Failover Cluster Manager then select Roles
2. Verify the status of the cluster is up and running
3. To verify the Failover, right-click on the cluster instance, click Move and then select the preferred node from the available list and click OK. This will change the Owner node

This verifies the working of the Failover Cluster as the Owner node is changed from previous node to the preferred node.

8.4 SQL Server Clustering with Utimaco HSM

After the setup of Microsoft SQL Server cluster, one or more Utimaco HSMs can be used (along with internal/external keystore). For the illustration purpose one HSM is configured in this SQL Server Cluster setup.

To configure EKM Provider on the cluster nodes, refer section Enable Extensible Key Management

The Keys can be used from internal keystore or the external keystore, for creating keys refer section Creating Keys

RSA algorithm is not supported in FIPS mode.

1. On Primary Cluster Node use below query for creating keys using Utimaco HSM.

PROVIDER_KEY_NAME = 'RSA2048Key1',

CREATION_DISPOSITION=CREATE_NEW;

2. Insert and encrypt the data using Utimaco HSM keys.

SQL Statement

```
USE Testdb GO

CREATE TABLE Customers (FirstName varchar (MAX), SecondName varchar(MAX),
CardNumber varbinary(MAX));

GO

INSERT INTO Customers (FirstName, SecondName, CardNumber)

VALUES ('Kyle', 'Hood', ENCRYPTBYASYMKEY (ASYMKEY_ID('RSA2048Key1'),
'2048204820482048'));
```

3. Follow the steps from Verify Failover Cluster Configuration to perform the Failover Scenario.

4. Now as the primary Cluster Node 1 is down, SQL Failover configuration will make Cluster Node 2 as primary. On the new Primary SQL Server, run the below query to decrypt values.

SQL Statement

```
USE Testdb GO

SELECT FirstName, SecondName, CONVERT (varchar, DECRYPTBYASYMKEY
(ASYMKEY_ID('RSA2048Key1'), CardNumber))

AS 'CardNumber' FROM Customers; GO
```

9 Database Mirroring

9.1 SQL Server Setup

1. SQL Servers (Principal, Mirror & Witness) must be in a Domain
2. Configure SQL Server Service with the Domain Administrator Account
    a. Go to the SQL Server Configuration Manager. Select SQL Server Services
    b. In the right pane, right-click on SQL Server (MSSQLSERVER). Select, Properties
    c. Click on Log On tab, select This Account radio button and click on Browse to select the Domain Administrator Account
    d. Click, OK. Restart the service
3. As the SQL Server service is configured with Domain Administrator Account, the user needs to configure the Security Login with the same account
    a. Open SSMS in the Object Explorer, expand the Security tab.
    b. Right-click on Login then select New Login. Click on Search button to select the Domain Administrator Account

9.2 Configure Database Mirroring

1. On the Principal SQL Server, open SMSS, right-click the database from the Object Explorer, select Tasks then select Backup Database Window appears. Now select appropriate database from the Source drop-down menu
2. Select Full from the Backup Type drop-down menu
3. Select the Backup File Path. Then Save with <file-name>.bak extension, click OK

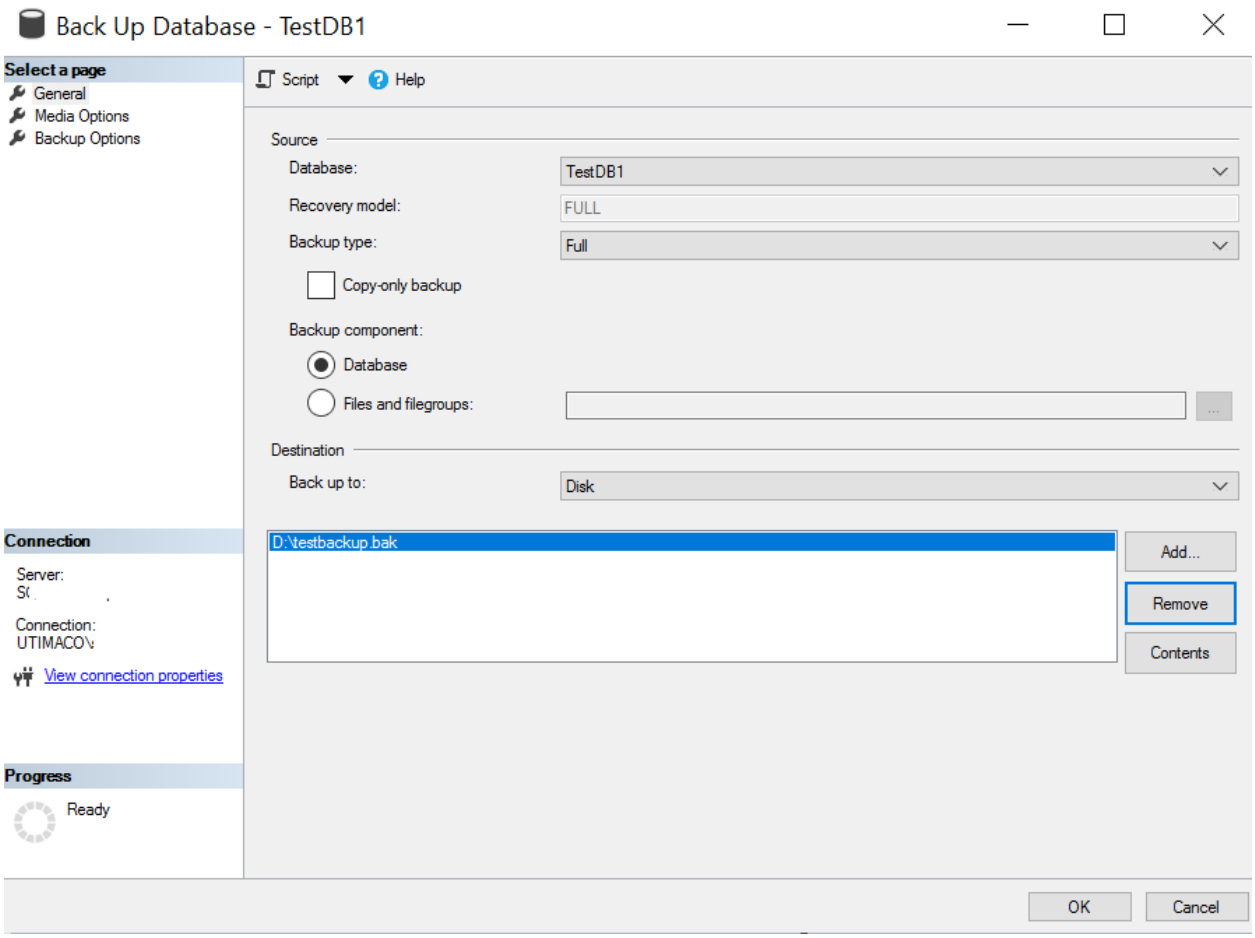

Figure 6: Back Up Database window

4. Similarly, select the same database. Right-Click and select Tasks and then select Restore Database
5. Select Transactional Log from the Backup Type drop-down menu
6. Select the Backup File Path then Save with <file-name>.trn extension , click OK
7. Now, copy the Backup files to the Mirror SQL Server. Restore the database on the Mirror SQL Server
8. On the Mirror SQL Server, open SMSS, Right-Click on the Database and select Restore Database
9. Select the Device radio button and select the new created Backup Files
10. Select the Options and select the RESTORE with NO RECOVERY from the Recovery State drop-down menu, click OK

This will restore the database from the Principal SQL Server to the Mirror SQL Server.

11. On the Principal SQL Server, select the same database. Right-Click and select Tasks and then select Mirror Database
12. Configure Database Mirroring Security wizard appears. Click Next for another two wizards
13. Select Witness Server checkbox
14. Now Click Next on the Principal SQL Server wizard

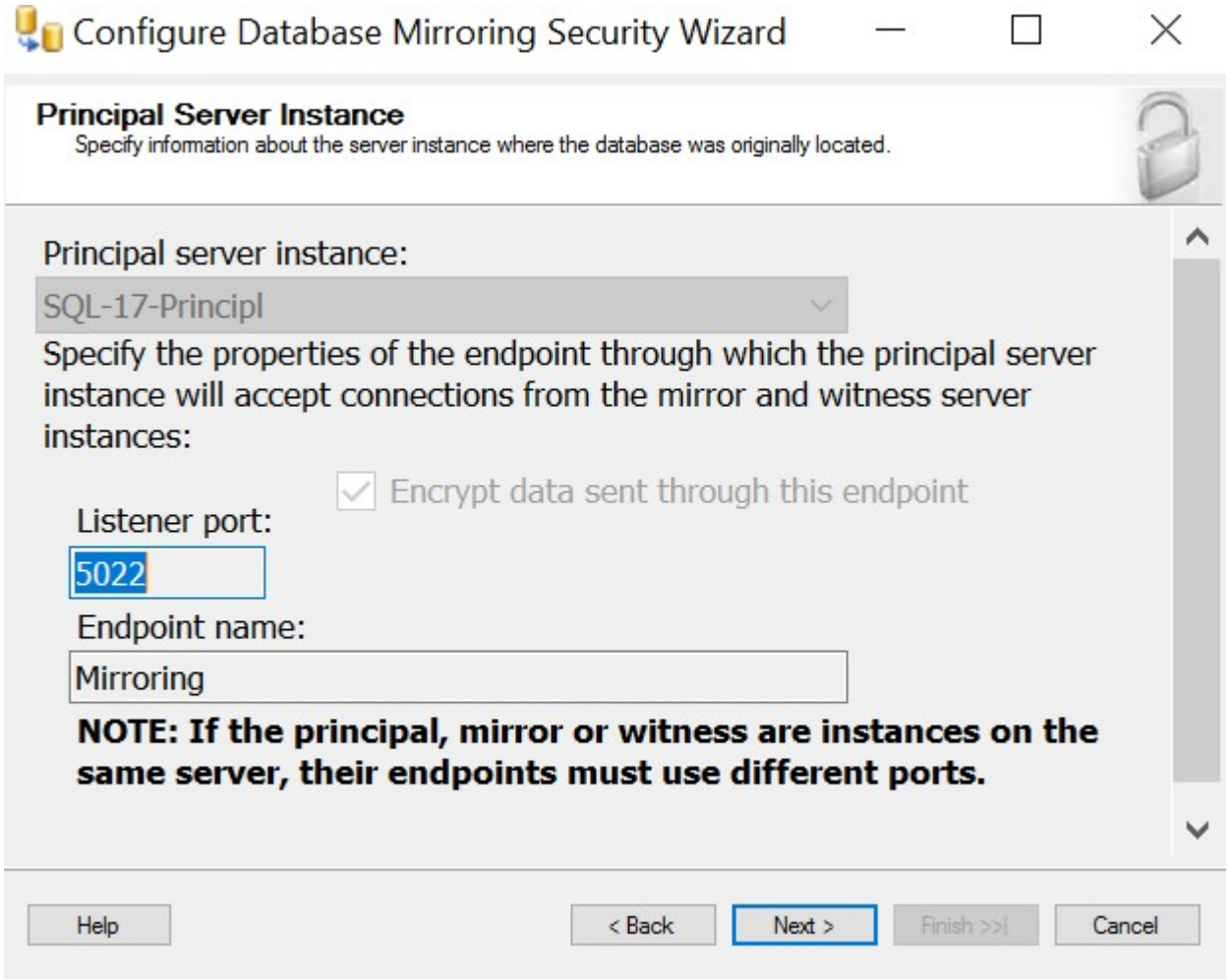

Figure 7: Configuring Database Mirroring Security Wizard

15. Select the Mirror server instance from the drop-down menu, click Connect and click Next

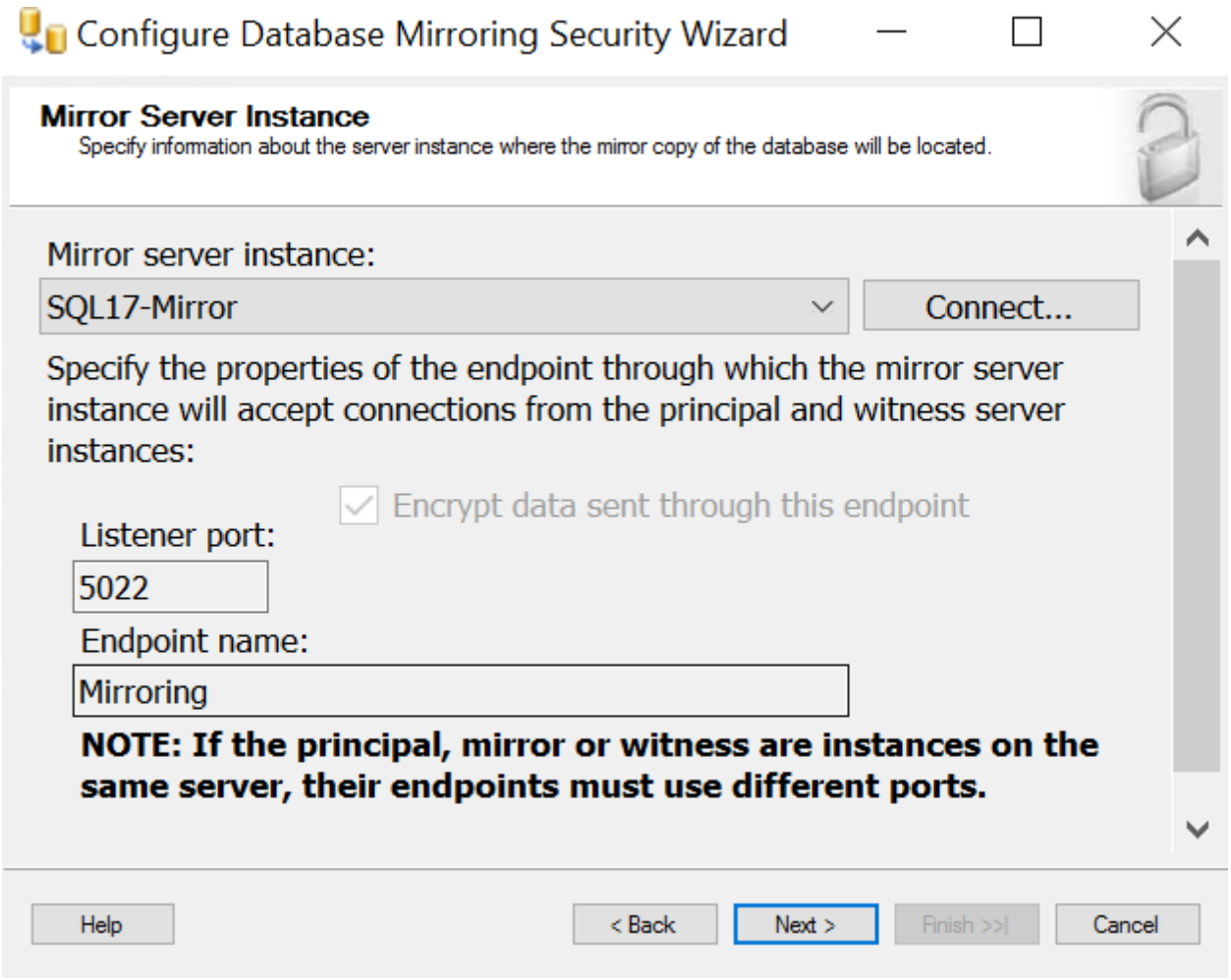

Figure 8: Configuring Database Mirroring Security Wizard

16. Select the Witness server instance from the drop-down menu, click Connect and click Next

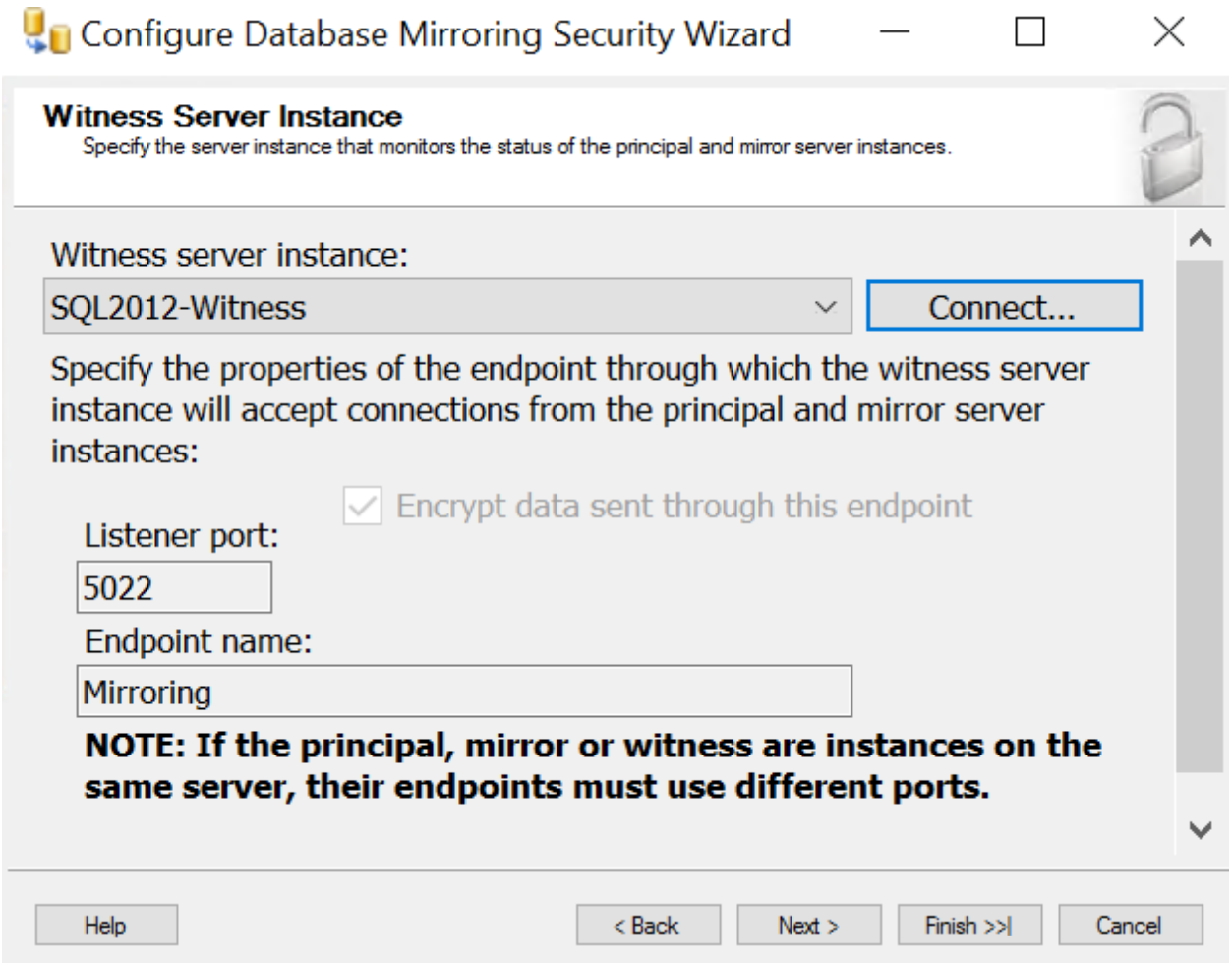

Figure 9: Configuring Database Mirroring Security Wizard

17. Enter the Domain Admin Account for Principal, Witness and Mirror Instance. Click Next

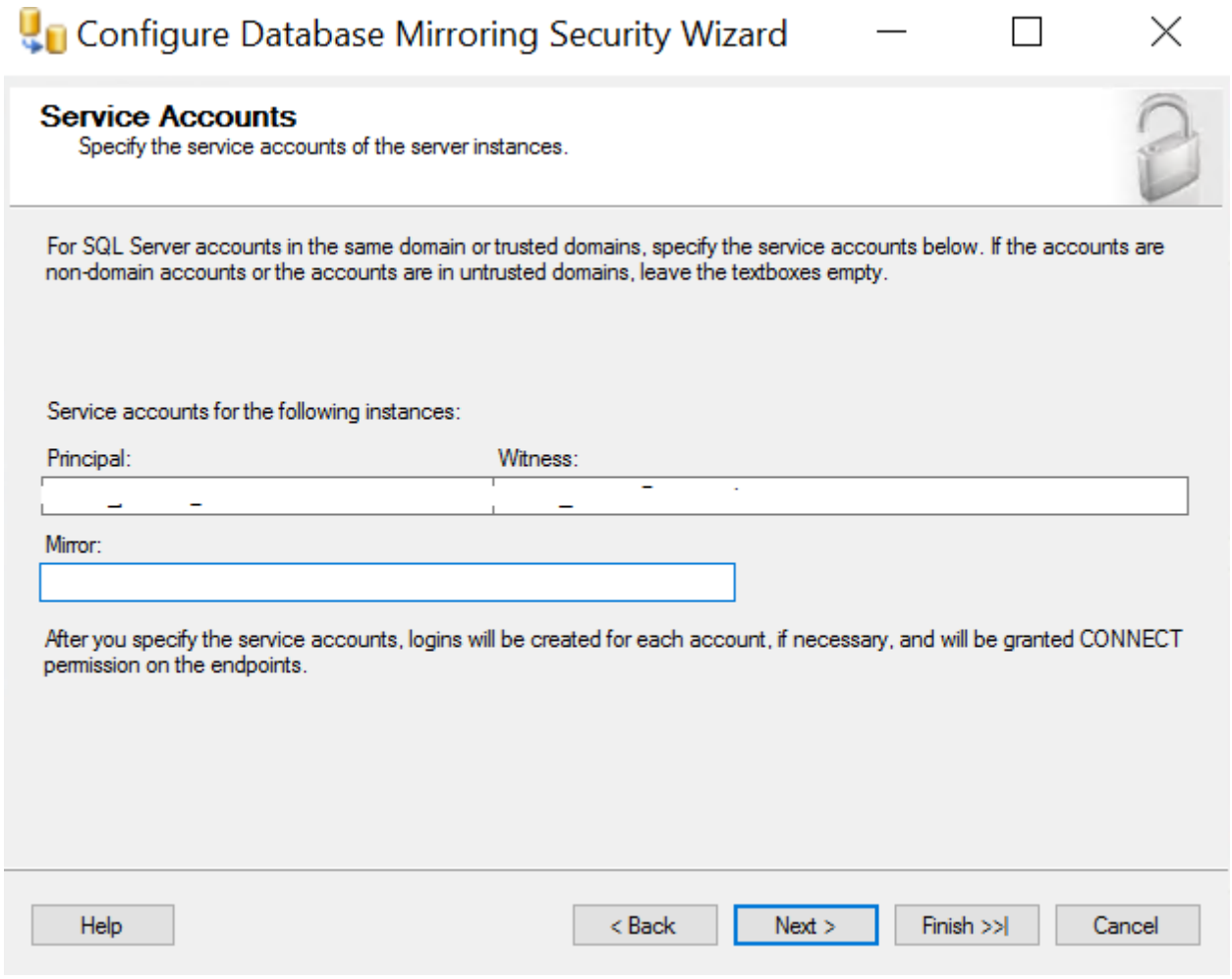

Figure 10: Configuring Database Mirroring Security Wizard

18. Verify the details, click Finish
19. Database Properties pop-up appears, select Start Mirroring

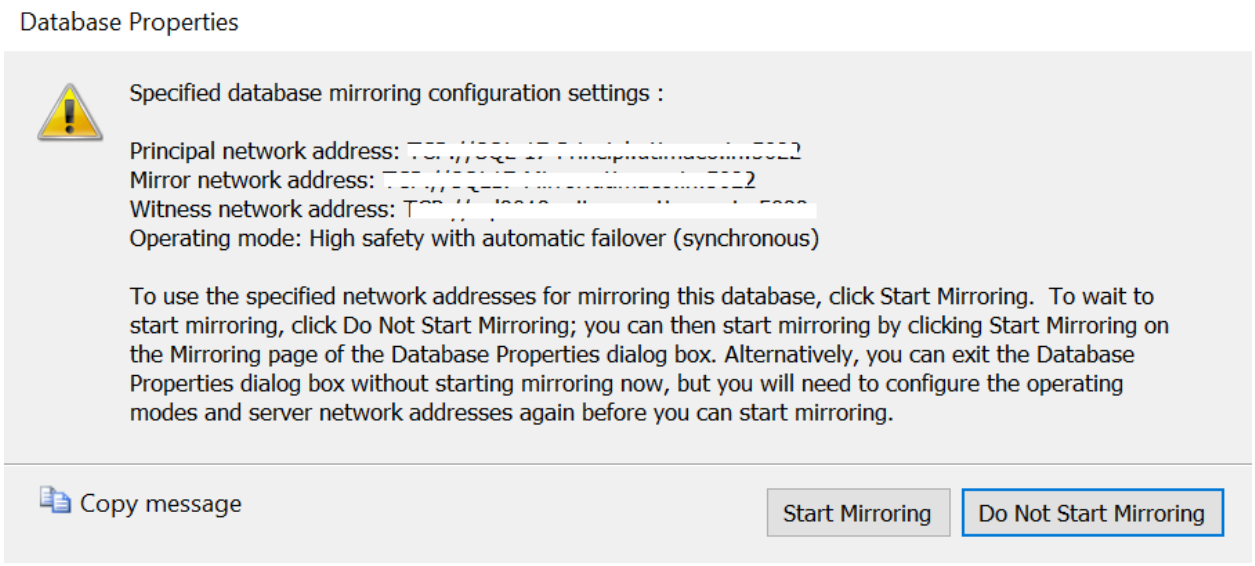

Figure 11: Database Properties Wizard

20. This might take few seconds, once done the below shown window is seen

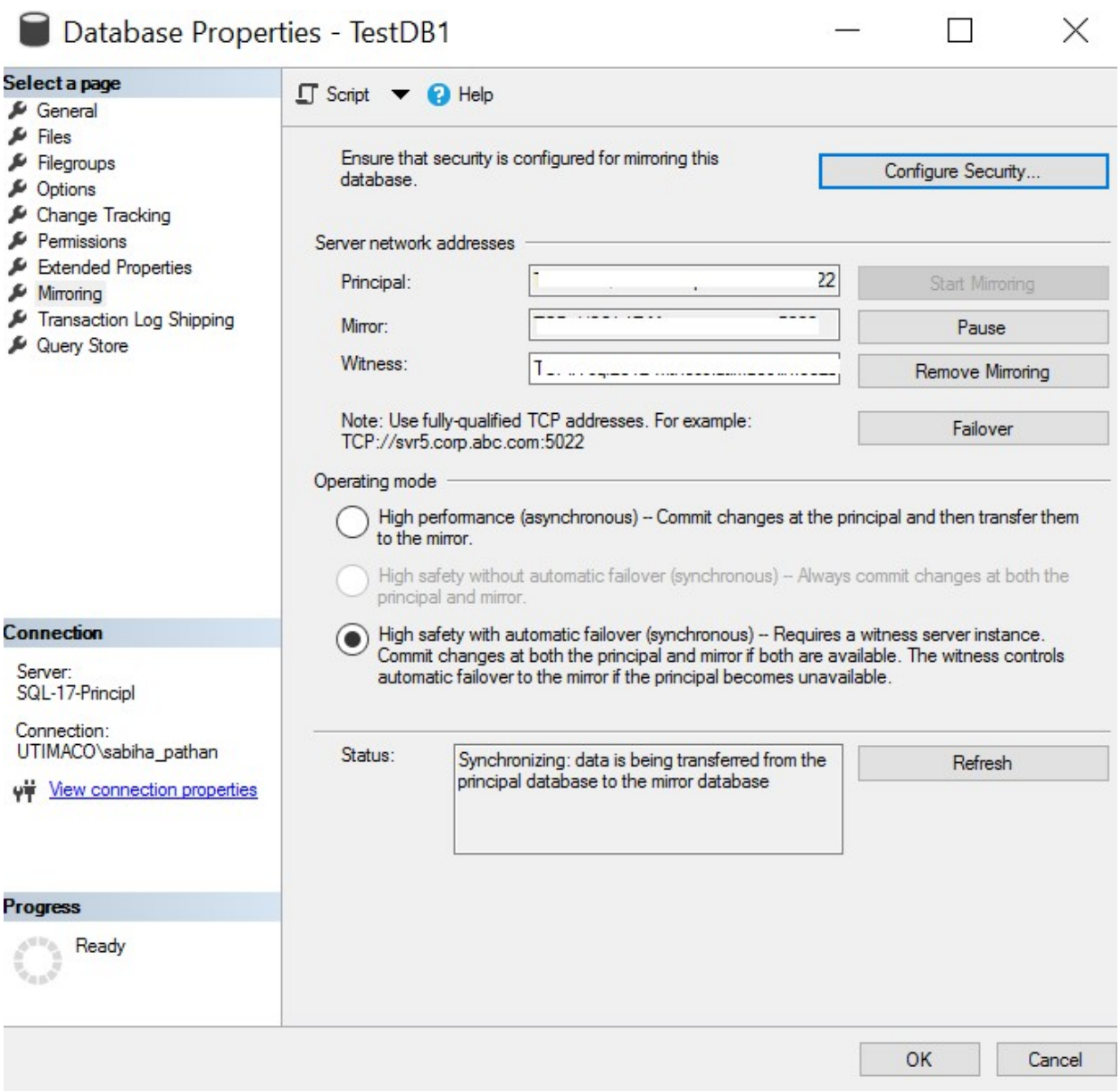

Figure 12: Database Properties Wizard

21. On the Principal SQL Server expand the Databases and Locate the appropriate database, which shows Principal, Synchronizing after its name

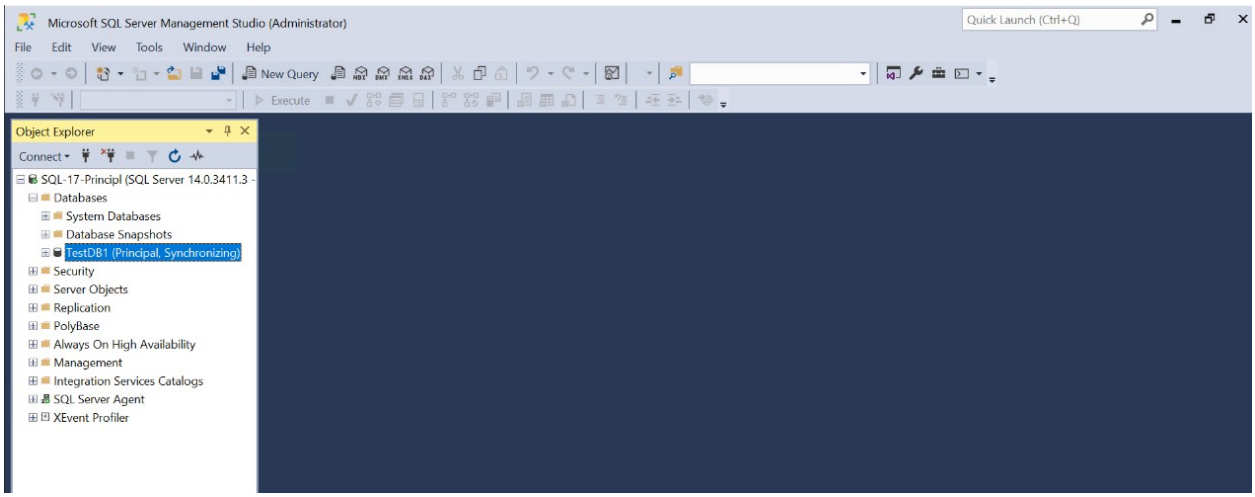

Figure 13: Database Properties Wizard

22. Similarly, on Mirror SQL Server expand the Databases and Locate the appropriate database, which shows Mirror, Synchronized/Restoring after its name

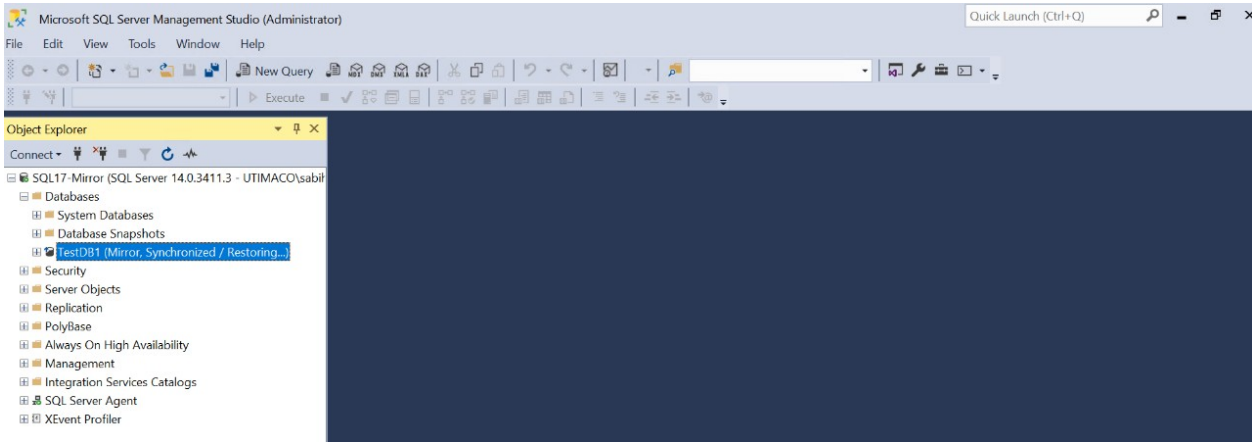

Figure 14: Database Properties Wizard

9.3 Database Mirroring with Utimaco HSM

Once the Database Mirroring is configured, one or more Utimaco HSMs can be used along with internal/external keystore. For the illustration purpose one HSM is configured in this SQL Server Database Mirroring configuration.

To configure EKM Provider on the cluster nodes, refer section Enable Extensible Key Management

The Keys can be used from internal keystore or the external keystore, for creating keys refer section Creating Keys

RSA algorithm is not supported in FIPS mode.

1. On Principal Server Instance use the Mirrored Database for creating keys using Utimaco HSMs.
2. Create an asymmetric key in the TestDB1 database.

SQL Statement

```
USE TestDB1;

CREATE ASYMMETRIC KEY tdekey

FROM PROVIDER utimaco

WITH ALGORITHM = RSA_2048, PROVIDER_KEY_NAME = 'tdekey',

CREATION_DISPOSITION=CREATE_NEW;
```

3. Insert the data into the table

SQL Statement

```
USE TestDB1 GO

CREATE TABLE Customers (FirstName varchar (MAX), SecondName varchar(MAX),
CardNumber varbinary(MAX));

GO

INSERT INTO Customers (FirstName, SecondName, CardNumber)

VALUES ('Iain', 'Hood', ENCRYPTBYASYMKEY (ASYMKEY_ID('RSA2048Key'),
'2048204820482048'));
```

4. The Key and the Database is created in the Principal Server using Utimaco HSM. This data gets synchronized automatically in Mirror Server.

9.3.1 To Test Failover Scenario

1. Select the database. Right-Click and select Tasks and then select Mirror

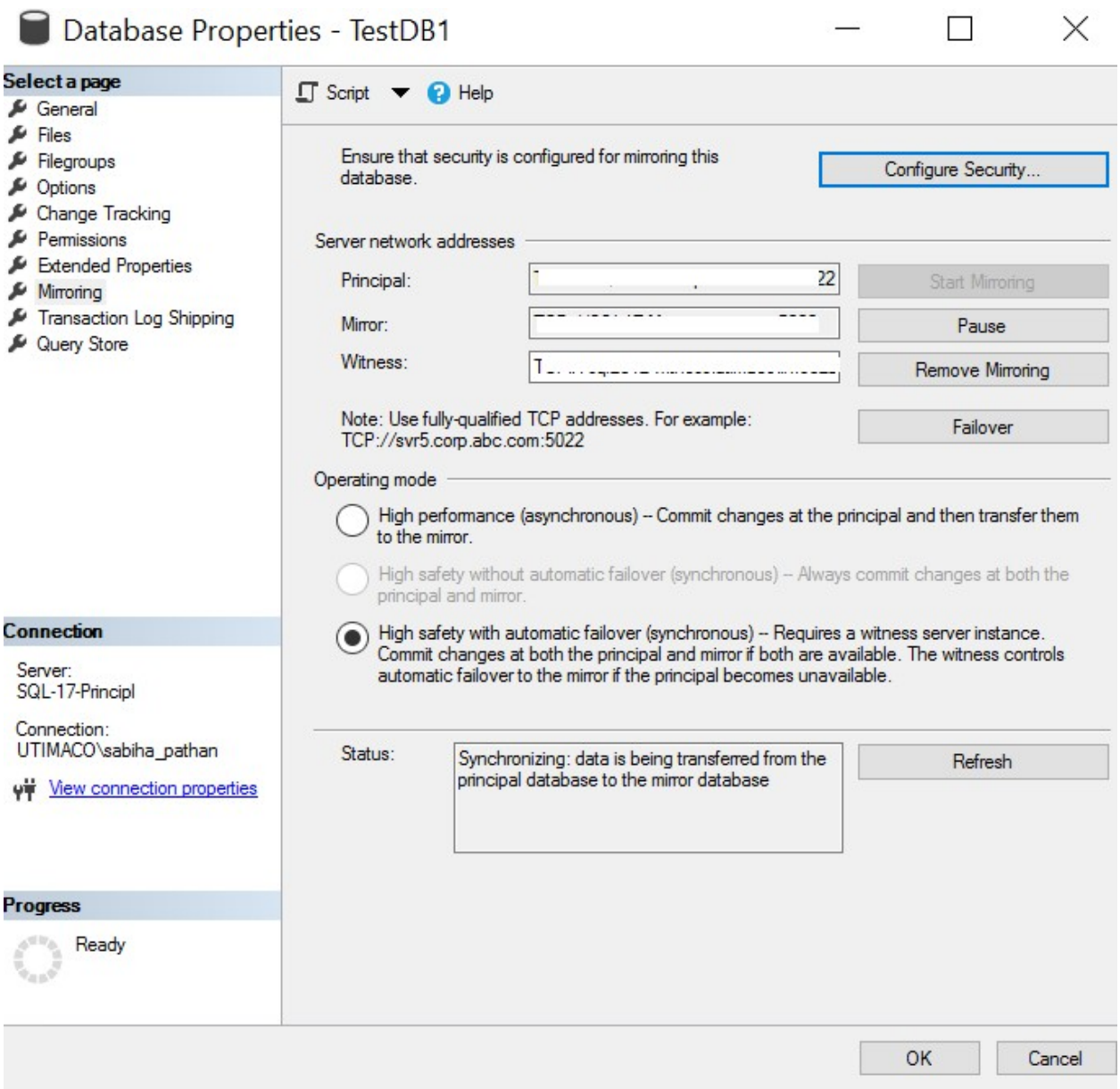

Figure 15: Database Properties Wizard

2. Select Failover and select Yes on the next Dialogue box that appears
3. This will make the current Principal Server as Mirror Server and vice-versa
4. Click Refresh, to reflect the changes

5. On the new Principal Server, run the below query to verify Database Mirroring with Utimaco HSM is working as expected

? Unknown Attachment SQL Statement

```
USE TestDB1 GO

SELECT FirstName, SecondName, CONVERT (varchar, DECRYPTBYASYMKEY
(ASYMKEY_ID('RSA2048Key'), CardNumber))

AS 'CardNumber' FROM Customers; GO
```

10 Troubleshooting

Error	*Diagnosis*
Cannot load library 'C:\Program Files\Utimaco\SeurityServer\Lib\cssqlekm.dll'. See error log for more information.	1. Create a new variable EKMCONFIGPATH in System variables. 2. Assign Path to Configuration File "C:\ProgramData\Utimaco\EKM\cssqlekm.cfg". 3. Restart is mandatory
Cannot continue the execution because the session is in the kill state. Msg 0, Level 20, State 0, Line 9 A severe error occurred on the current command. The results, if any, should be discarded.	1. Check if Security Server Application is Up Running and connected to HSM. 2. Check if the Configuration File pointing towards IP Address of HSM. 3. Restart the Microsoft SQL Server Service
Cannot open session for cryptographic provider 'utimaco'. Provider error code: 1. (Failure - Consult EKM Provider for details)	1. Check if the "Database via ODBC" section in Configuration File are Disabled. 2. Verify the correct Credentials mapped to Microsoft SQL Server login.
Server principal 'test/admin' has no credential associated with cryptographic provider 'utimaco'.	Map a Credential to Microsoft SQL Server login.

Key Type property of the key returned by EKM provider doesn't match the expected value	1. This error occurs when the user is trying to create Asymmetric key or Symmetric key with unsupported algorithm. 2. RSA Algorithm is not supported in FIPS Mode.
When viewing the data from the table, output displays the NULL value instead of encrypted/ decrypted data.	Check if you are using correct encrypt/ decrypt key or the user may not have permission to access the key.

Table 8: List of Error and its Diagnosis

11 Further Information

This document forms a part of the information and support which is provided by the Utimaco IS GmbH. Additional documentation can be found on the product CD in the Documentation directory.

All CryptoServer product documentation is also available at the Utimaco IS GmbH website:

https://utimaco.com/

For more information regarding Microsoft SQL Server documentation, please see the following link:

Microsoft SQL documentation - SQL Server | Microsoft Docs