Microsoft

SQL Server EKM Provider

v1.5

Integration Guide

u.trust GP HSM

6.0.0 and 6.1.1

utimaco®

Imprint

Copyright 2026	Utimaco IS GmbH Krefelder Straße 220 52070 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	1.0.0
Date	2026-08-07
Status	PUBLISHED
Document No.	IG-2026-0065

All rights reserved	No part of this documentation may be reproduced in any form (printing, photocopy or according to any other process) without the written approval of Utimaco IS GmbH or be processed, reproduced or distributed using electronic systems. Utimaco IS GmbH reserves the right to modify or amend the documentation at any time without prior notice. Utimaco IS GmbH assumes no liability for typographical errors and damages incurred due to them. All trademarks and registered trademarks are the property of their respective owners.

Table of Contents

1 Introduction 5
1.1 About This Guide 5
1.2 Target Audience for This Guide 5
1.3 Abbreviations 5
1.4 Document Conventions 6
2 Product Overview 8
2.1 Microsoft SQL Server EKM Provider 8
2.2 Utimaco u.trust GP HSM 8
3 Integration Requirements and Prerequisites 9
3.1 Tested Versions 9
3.2 Hardware and Software Requirements 9
3.3 Prerequisites 10
4 Installation and Configuration 11
4.1 Setting Up u.trust GP HSM 11
4.2 Setting Up Microsoft SQL Server 11
5 Integration Steps 13
5.1 Configuration on u.trust GP HSM 13
5.1.1 Location of the Configuration File 13
5.1.2 Customization of the Configuration File 13
5.1.3 Configuration of the External Keystore 14
5.2 Configuration on Microsoft SQL Server 16
5.2.1 Enable Extensible Key Management 16
5.2.2 Register Provider 16
5.2.2.1 Alter Provider Location 17
5.2.2.2 Remove Provider 17
5.3 Key/Token Management 17
5.3.1 Improving Security via CXI Group 19
5.3.2 Cryptographic User Hierarchy 19
6 Verification and Testing 21
6.1 Functional Testing 21
6.1.1 Key Management 21

6.1.1.1 Creating Keys 21
6.1.1.2 Viewing Keys 22
6.1.1.3 Deleting Keys 23
6.1.2 Column Level Encryption 24
6.1.3 Transparent Data Encryption 26
6.1.4 Microsoft SQL Server Clustering 28
6.1.4.1 Installing Failover Clustering Feature on the Cluster Nodes 28
6.1.4.2 Install and Configure MS SQL on a Failover Cluster 29
6.1.4.3 Verify Failover Cluster Configuration 32
6.1.4.4 SQL Server Clustering with Utimaco HSM 33
6.1.5 Database Mirroring 34
6.1.5.1 SQL Server Setup 34
6.1.5.2 Configure Database Mirroring 35
6.1.5.3 Database Mirroring with Utimaco HSM 43
7 Troubleshooting 47
7.1 Common Issues and How to Resolve Them 47
7.2 Log Locations and Interpretation 48
8 Appendices 50
8.1 References 50
8.2 Command Summary 50

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 Abbreviations

Abbreviation	***Meaning***
AES	Advanced Encryption Standard
CA	Certificate Authority
CAT	CryptoServer Administration Tool
CNG	Cryptography API Next Generation
CXI	Cryptographic eXtended Interface
DES	Data Encryption Standard

Abbreviation	*Meaning*
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 1: List of Abbreviations

1.4 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.
Italic	References and important terms	See Chapter 3, "Sample Chapter", in the *u.trust Anchor - csadm Manual* or [CSADM].

Table 2: 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.

2 Product 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

The u.trust GP HSM is a next-generation hardware security module developed by Utimaco IS GmbH. It is a multi-tenant, physically protected, and tamper-resistant cryptographic appliance designed to perform high-assurance cryptographic operations and manage cryptographic keys securely. The u.trust GP HSM is built on a modern, container-based design inspired by cloud technology. With support for up to 31 containers and multiple PKCS #11 partitions per cHSM, it ensures seamless application separation and key partitioning, making it an ideal choice for all types of cryptographic applications. It's also upgradeable for specific use cases like blockchain and 5G, and offers flexibility for custom solutions, including proprietary algorithms and customer key derivations via the Software Development Kit.

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, and Utimaco u.trust Anchor cHSM are shown in the following configurations below:

Microsoft Windows Server	Microsoft SQL Server	Utimaco u.trust Anchor cHSM Version	Utimaco HSM
Windows Server 2022	SQL Server 2022	u.trust Anchor cHSM 6.0.0 and 6.1.1	u.trust GP HSM

Table 3: List of Tested Versions

3.2 Hardware and Software Requirements

Hardware	Hardware Requirement
Utimaco u.trust GP HSM	u.trust GP HSM available with firmware SecurityServer 6.0.0 and 6.1.1

Table 4: List of Tested Versions

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

Table 5: List of Tested Versions

3.3 Prerequisites

Before you begin, please ensure that you have:

- A u.trust GP HSM set up and configured. Refer to the u.trust GP HSM documentation to set up the HSM.
- An MBK created and stored on each cHSM. Refer to the u.trust GP HSM documentation to set up the MBK.
- The u.trust GP HSM Default Admin replaced with a new admin user for production environments.
- An operating system listed in Tested Versions.
- An SQL Server listed in Tested Versions.
- An SQL Server Management Studio (optional).
- The cHSM SecurityServer version listed in Tested Versions with the 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.

4 Installation and Configuration

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

4.1 Setting Up u.trust GP HSM

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, you will need the associated product bundle containing the required Windows software packages. This bundle can be downloaded from the Utimaco Support Portal. Please ensure you request access to the product bundle beforehand through the support portal.

Install the latest version of the SecurityServer software following the instructions provided in the Section 4.1 of *u.trust Anchor Administration Manual* [UTAADMIN]. For integration with Microsoft SQL, only the EKM Application Interface is required. After completing the installation, restart the Microsoft SQL service.

4.2 Setting Up Microsoft SQL Server

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.

5 Integration Steps

5.1 Configuration on u.trust GP HSM

5.1.1 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`.

5.1.2 Customization of the Configuration File

The SecurityServer EKM provider can be customized using the configuration file. Edit the configuration file to your needs; at a minimum, 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***
ConnectionTimeout	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). By default a cHSM uses the port 4000 + cHSM slot (4001 for slot 1).
KeysExternal	To enable the external keystore.

KeyStorageType	The database type for keystorage. This can be `sdb` for the Legacy SDB file or `odbc` for an ODBC source.
KeyStorageConfig	Configuration of the KeyStorage. This can be a filepath for the Legacy SDB file or `"DSN=<ODBC DSN>"` for the ODBC source.
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	This variable 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 will be created.
Timeout	Specifies the maximum time in milliseconds to wait for the answer from SecurityServer after sending a command.

Table 6: List of Configuration Parameters

5.1.3 Configuration of the External Keystore

This section describes how to initialize and configure an external KeyStore using an SQL Server database. The SecurityServer EKM provider can use the external KeyStore to store and retrieve cryptographic keys via an ODBC connection.

Before proceeding, ensure the following:

- SQL Server is installed and accessible.
- The **ODBC Driver for SQL Server** is installed (latest version recommended).
- The `cxitool` utility is available as part of the SecurityServer software bundle.
- A SQL Server user with appropriate permissions exists, and the target database has been created or can be created.

ODBC Driver Download:
You can download the latest Microsoft ODBC Driver for SQL Server from:
https://learn.microsoft.com/en-us/sql/connect/odbc/download-odbc-driver-for-sql-server

To configure the external keystore, the following steps must be followed:

1. Configure the ODBC Data Source.
    a. Open the **ODBC Data Source Administrator** from the **Windows Start** menu.
    b. Select **Add**, then choose the **ODBC Driver for SQL Server** from the list.
    c. Configure the Data Source Name (DSN), SQL Server hostname/IP, authentication details, and database name.
    d. Test the connection to ensure successful communication with the SQL Server instance.

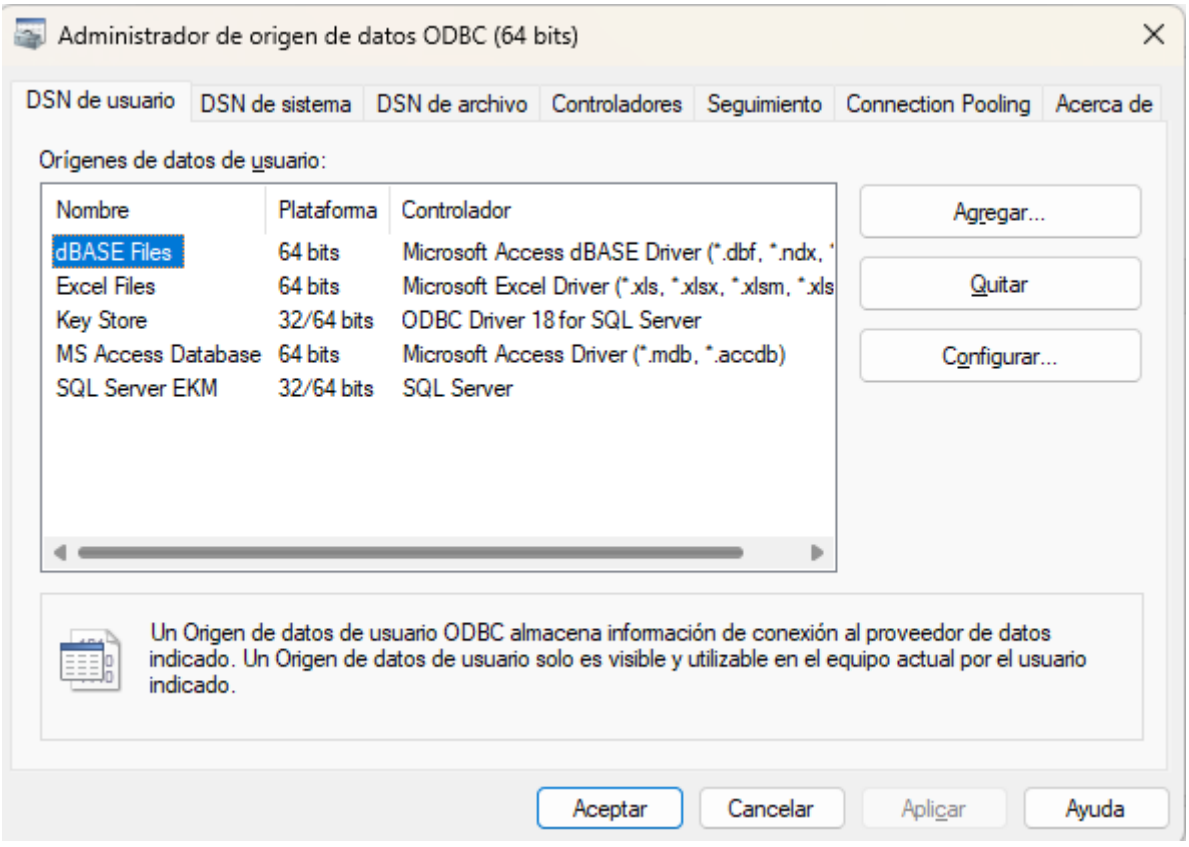


Figure 1 : ODBC Data Source Administrator

2. Initialize the KeyStore Database. Once the ODBC connection is configured, you can initialize the database schema for KeyStore storage using the following command.

```
cxitool dbConnString="DSN=<ODBC>;Uid=<Username>;Pwd=<Password>"
 CreateDBSchema=mssql
```

3. Update the EKM Provider Configuration. After initializing the KeyStore, you must update the EKM provider configuration file (`cssqlekm.cfg`) to use the external KeyStore.

Ensure that the SQL Server service account has read/write access to the KeyStore database and that the configuration file is correctly referenced in the system path or `EKMCONFIGPATH` environment variable.

5.2 Configuration on Microsoft SQL Server

5.2.1 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 log in with administrative privileges. Open a query window for further execution of SQL statements. To enable Extensible Key Management, please run the following SQL.

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

5.2.2 Register Provider

Run the following SQL to register the provider under the name "Utimaco."

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

```
GO
```

An error during this step could mean that the location of `cssqlekmlib.dll` is not included in the system PATH.

To verify the successful installation, run:

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

5.2.2.1 Alter Provider Location

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

```
ALTER CRYPTOGRAPHIC PROVIDER utimaco
FROM FILE = '<path-to-new-provider-dll>'
GO
```

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.

5.2.2.2 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.

```
DROP CRYPTOGRAPHIC PROVIDER utimaco
GO
```

5.3 Key/Token Management

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 a login on the SecurityServer.

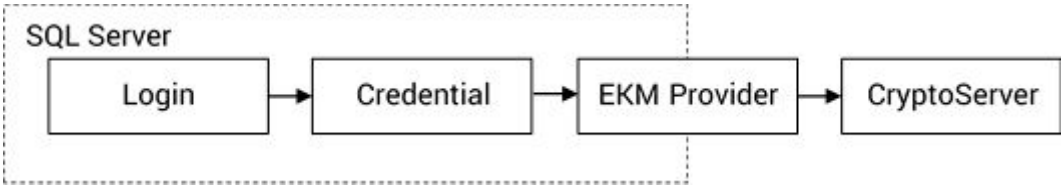


Figure 2 : Credential Mapping

Multiple SQL Server logins can use the same credentials. If the EKM providers are different, a login can be used with multiple credentials. Otherwise, the lookup shown before will fail.

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

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

Creating a cHSM user needs to be done either via csadm or CAT.

For detailed information, refer to chapter 3.3 of the u.trust Anchor cHSM Administration Manual.

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]`.

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

5.3.1 Improving Security via CXI Group

By default, new EKM keys are generated without a 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:

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

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 a 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, using different CXI groups for different credentials also prevents name collisions when SQLEKM is used with different databases from the same SQL Server.

5.3.2 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 Credentials and 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.

6 Verification and Testing

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.

6.1 Functional Testing

6.1.1 Key Management

6.1.1.1 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:

```
CREATE SYMMETRIC KEY EKM_AES_256
FROM PROVIDER utimaco WITH ALGORITHM = AES_256,
PROVIDER_KEY_NAME = 'EKM_AES_256', CREATION_DISPOSITION=CREATE_NEW
GO
```

Note that the key name `EKM_AES_256` appears twice here: first as a 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.

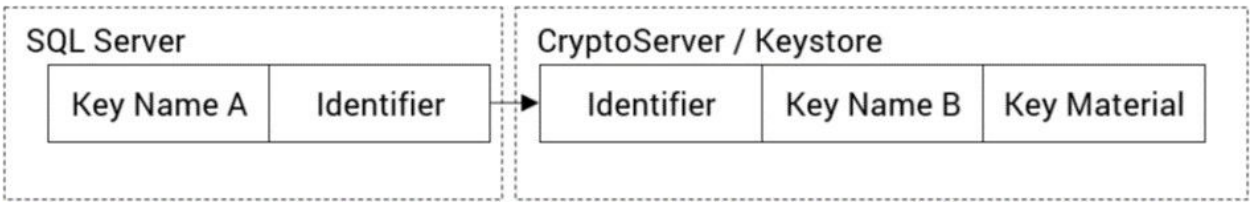


Figure 3 : Key Mapping

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

```
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 a 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 the external keystore.

```
cxitool Dev=<port@IP> LogonPass=<user>,<password> keystoretype=SDB |
keystoreparam="C:\ProgramData\Utimaco\EKM\cssqlekm.sdb" group="" |
Name= EKM_AES_256 Usage=ENCRYPT,DECRYPT,SIGN,VERIFY spec=0 generatekey=AES,256
```

```
CREATE SYMMETRIC KEY AEKM_AES_256
FROM PROVIDER utimaco
WITH PROVIDER_KEY_NAME = 'OtherAesKey', CREATION_DISPOSITION=OPEN_EXISTING
GO
```

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:

```
CREATE ASYMMETRIC KEY EKM_RSA_2048
FROM PROVIDER utimaco
WITH ALGORITHM = RSA_2048, PROVIDER_KEY_NAME = 'EKM_RSA_2048',
CREATION_DISPOSITION=CREATE_NEW
GO
```

6.1.1.2 Viewing Keys

The SQL Server provides several SQL statements to list existing cryptographic keys. The first set of statements shows 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 the current symmetric keys:

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

This will show a listing of current asymmetric keys:

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

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.

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

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

6.1.1.3 Deleting Keys

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

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

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

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

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 still exists in the EKM provider. To delete both the binding and the EKM provider key, use the following statement for a symmetric key:

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

For Example:

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

Use this SQL statement to delete an asymmetric key.

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

For Example:

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

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

```
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, such as in the next SQL statement. This statement uses a symmetric column encryption for the column `secret`.

```
INSERT INTO demo
VALUES ('John', 'Doe', ENCRYPTBYKEY(KEY_GUID('CLE_AES_256'), 'utimaco'))
GO
```

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

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



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

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

5. A decryption of an asymmetric column can be achieved similarly to the decryption of a symmetric encrypted column:

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

6.1.3 Transparent Data Encryption

With the introduction of transparent data encryption (TDE) in SQL Server 2008, users now have the opportunity for full database-level encryption. TDE is the optimal choice for bulk encryption to meet regulatory compliance or corporate data security standards. TDE works at the file level, encrypting data directly on the hard drive. TDE does not replace column-level encryption. It is just another way of transparently encrypting your database data. 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.

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

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

```
USE master;
GO

CREATE ASYMMETRIC KEY tdekey
FROM PROVIDER utimaco
WITH ALGORITHM = RSA_2048, PROVIDER_KEY_NAME = 'tdekey',
CREATION_DISPOSITION=CREATE_NEW;
GO
```

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

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

4. Link your SQL Server credentials to your new user account with the next 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:

```
CREATE DATABASE demo GO
USE demo
```

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

```
CREATE DATABASE ENCRYPTION KEY WITH ALGORITHM = AES_256

ENCRYPTION BY SERVER ASYMMETRIC KEY tdekey
```

7. Enable the transparent data encryption and start encrypting the database as a background thread. Depending on the size of the database, it can take a while for the encryption to be completed.

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

8. To see the current state of the encryption, use the following 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;
```

6.1.4 Microsoft SQL Server Clustering

Microsoft SQL Server clustering is a collection of two or more physical servers (cluster nodes) connected through the LAN. Each host in an SQL server instance has the same access to shared storage. Clustering SQL servers achieves high availability and protection from disasters whenever a server hosting the SQL Server instance fails.

6.1.4.1 Installing Failover Clustering Feature on the Cluster Nodes

Please execute the following steps on the cluster nodes:

1. Log in to the cluster node as a user with Administrative privileges.
2. Select **Start** and then **Server Manager** to open Server Manager.
3. Under **Configure this Local Server**, 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, select the cluster node from the **Server Pool,** and click **Next.**
6. In the section **Server Roles,** click **Next.**
7. From the list of available features, select the **Failover Clustering** check box and click **Next.**

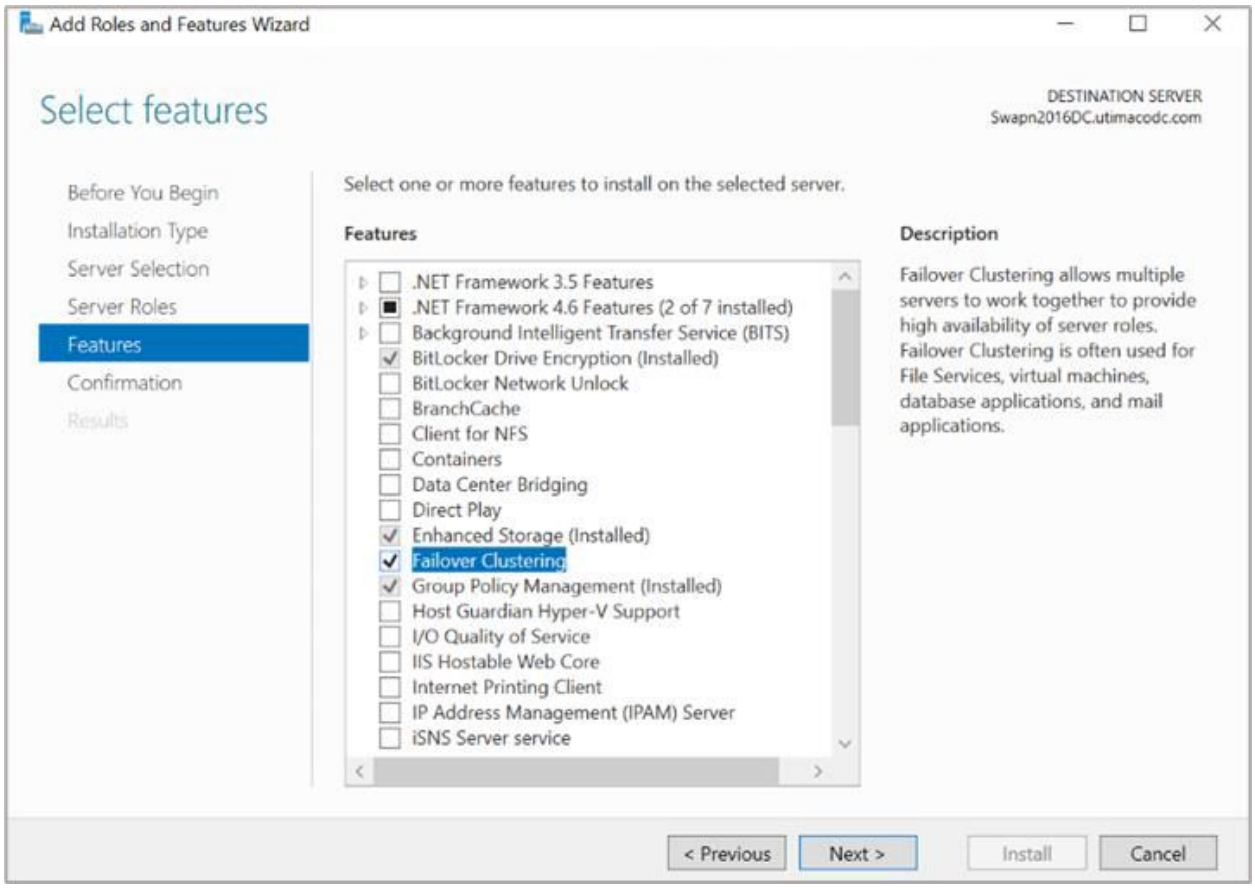


Figure 4 : "Select Installation Type" Window

8. A display pops up, 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 the Feature installation is complete, click **Close.** Similarly, configure it with the available nodes in the cluster.

6.1.4.2 Install and Configure MS SQL on a Failover Cluster

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

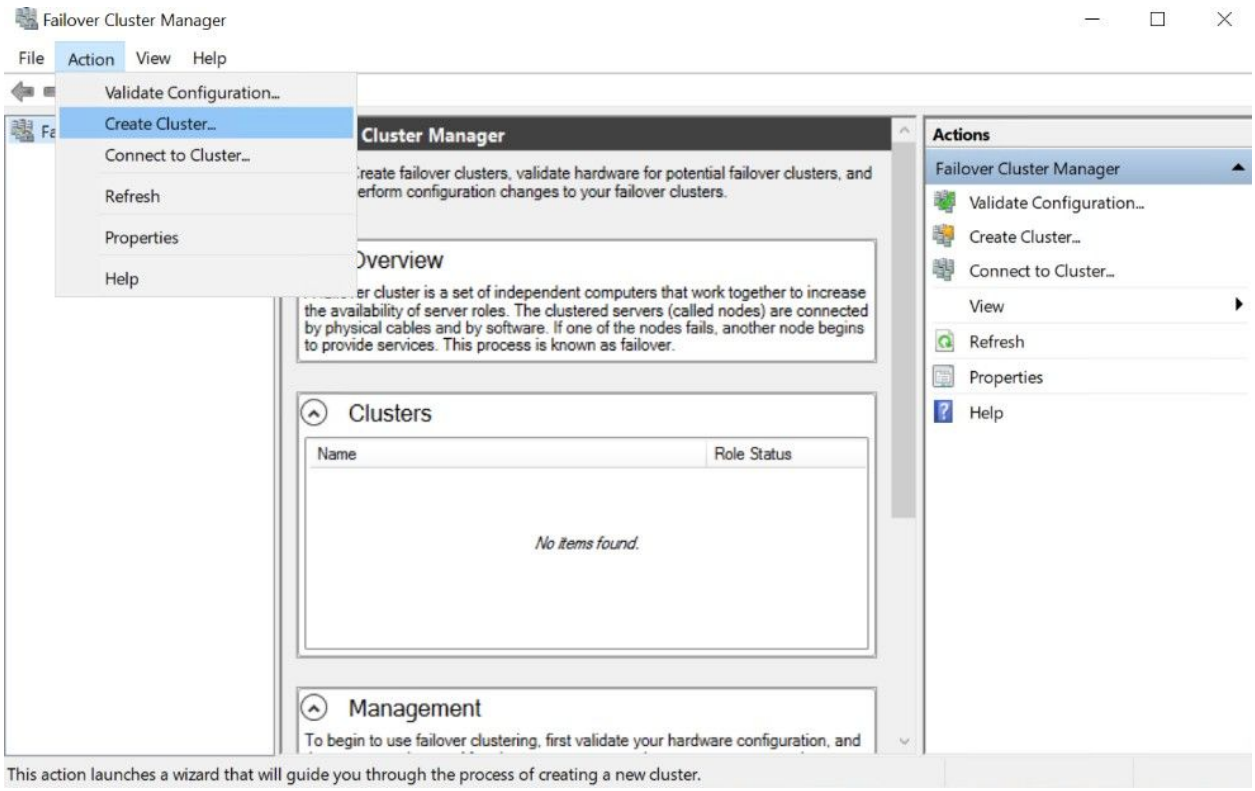


Figure 5 : "Failover Cluster Manager" Window

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.

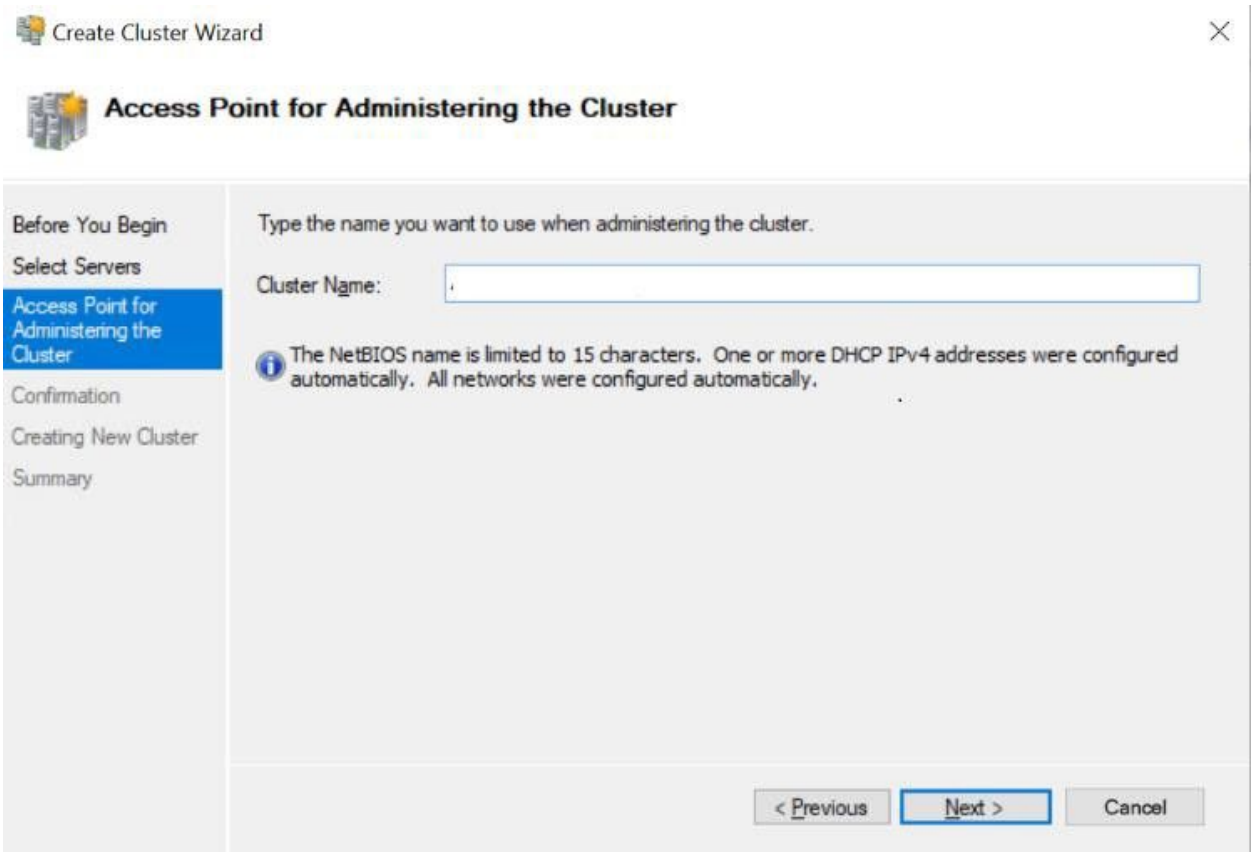


Figure 6 : "Create Cluster Wizard" Window

8. To perform the validation tests, choose **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 that the cluster was configured successfully. Check the status of the nodes, disk, and network. It should be in green.

6.1.4.2.1 Installation of SQL Server on Cluster Nodes

1. Go to the Cluster Node, where the Shared drive is mounted automatically by the 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** until the user gets the **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 the 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.**
13. Select the **Mix Mode** radio button, enter the password for the SQL Admin account, and select the username for SQL Admin account.
14. Select the **Data Directories** Tab and verify the appropriate shared drive directory paths.
15. Click **Next** for the next two wizards; the installation for SQL Status starts.
16. After the 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.**

6.1.4.3 Verify Failover Cluster Configuration

1. Open **Failover Cluster Manager,** then select **Roles.**
2. Verify that the status of the cluster is up and running.
3. To verify the Failover, right-click on the cluster instance, click **Move,** 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 the previous node to the preferred node.

6.1.4.4 SQL Server Clustering with Utimaco HSM

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

To configure EKM Provider on the cluster nodes, refer to the Enable Extensible Key Management chapter.

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



The RSA algorithm is not supported in FIPS mode.

1. On the Primary Cluster Node, use the following query for creating keys using the Utimaco HSM.

```
USE Testdb
GO
CREATE ASYMMETRIC KEY 'RSA2048Key1'
FROM PROVIDER utimaco
WITH ALGORITHM = RSA_2048,
PROVIDER_KEY_NAME = 'RSA2048Key1',
CREATION_DISPOSITION=CREATE_NEW;
GO
```

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

```
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'));
GO
```

3. Follow the steps from 8.3 Verify Failover Cluster Configuration to perform the Failover Scenario.

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

```
USE Testdb
GO
SELECT FirstName, SecondName, CONVERT (varchar, DECRYPTBYASYMKEY
(ASYMKEY_ID('RSA2048Key1'), CardNumber))
AS 'CardNumber' FROM Customers;
GO
```

6.1.5 Database Mirroring

This feature is a high-availability and disaster recovery solution provided by Microsoft SQL Server. In this type of configuration, there are three entities: Principal SQL Server, Mirror SQL Server, and Witness SQL Server, wherein Principal SQL Server serves as the primary/production instance, and Mirror SQL Server serves as a backup or high-availability instance. Both Principal and Mirror SQL Servers are in real-time synchronization, and changes are replicated on the Mirror SQL Server as per the changes made on the Principal SQL Server.

With the Utimaco HSM integration, data encrypted on the Principal SQL Server database can be decrypted from the Mirror SQL Server by using the same configuration and environment when the Failover Scenario occurs.

6.1.5.1 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 the **Search** button to select the Domain Administrator Account.

6.1.5.2 Configure Database Mirroring

1. On the Principal SQL Server, open **SMSS**, right-click the database from the **Object Explorer**, select **Tasks,** then the **Backup Database Window** appears. Select the 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 the `<file-name>.bak` extension and click **OK.**

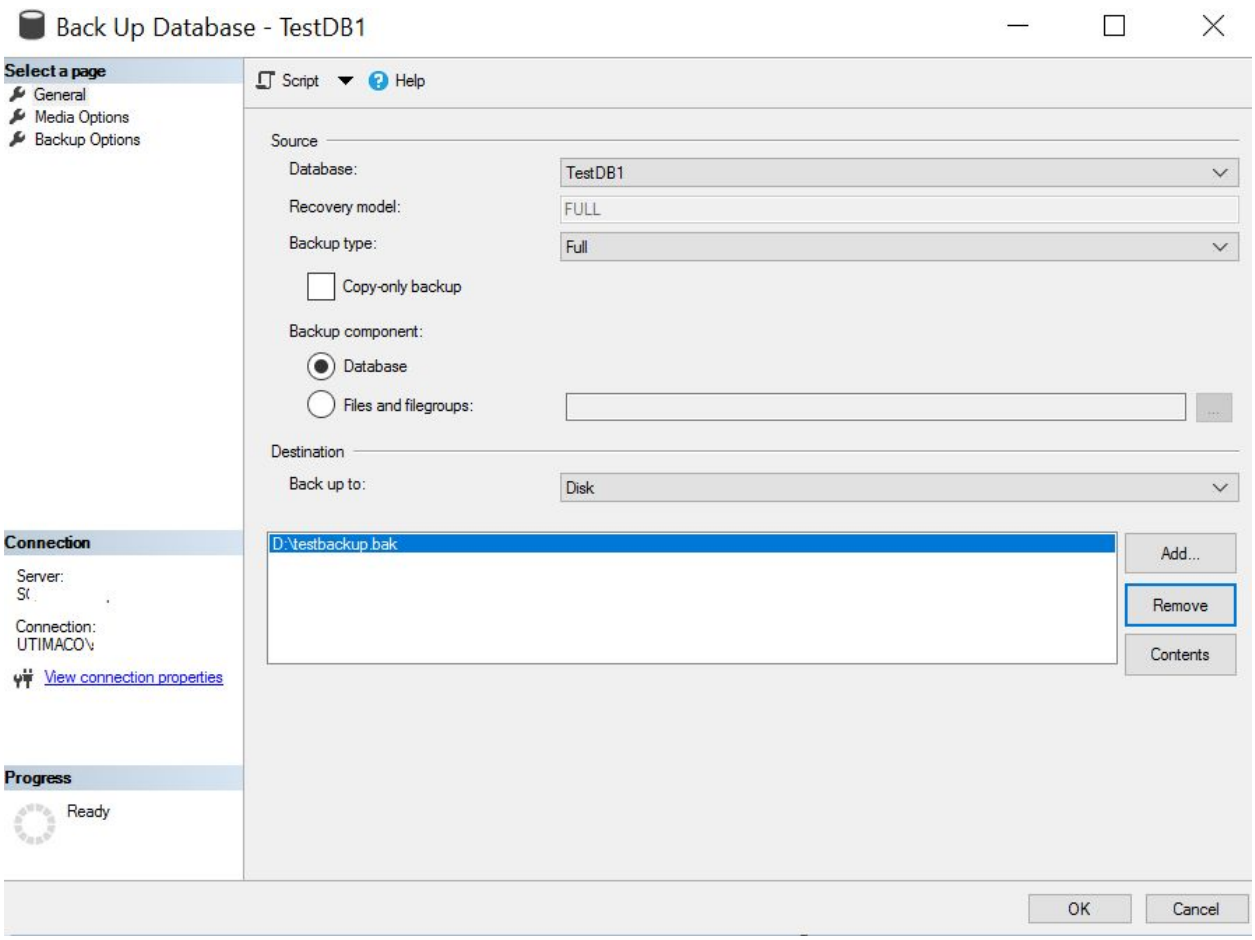


Figure 7 : "Back Up Database" Window

4. Similarly, select the same database. Right-click and select **Tasks,** 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. 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 newly created Backup Files.
10. Select the **Options** and choose **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. The **Configure Database Mirroring Security** wizard appears. Click **Next** for another two wizards.
13. Select the **Witness Server** checkbox.
14. Click **Next** on the Principal SQL Server wizard.

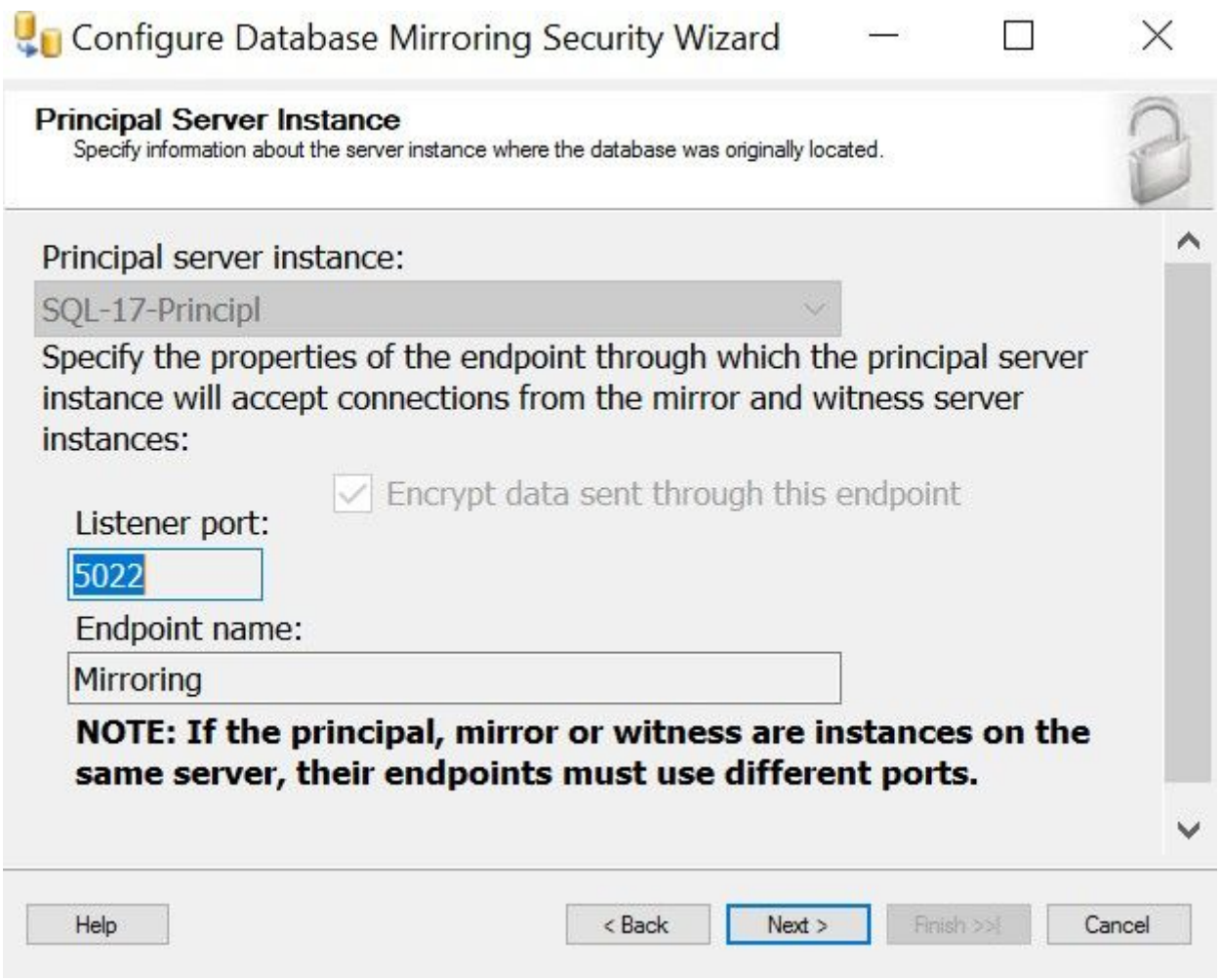


Figure 8 : Configuring Database Mirroring Security Wizard

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

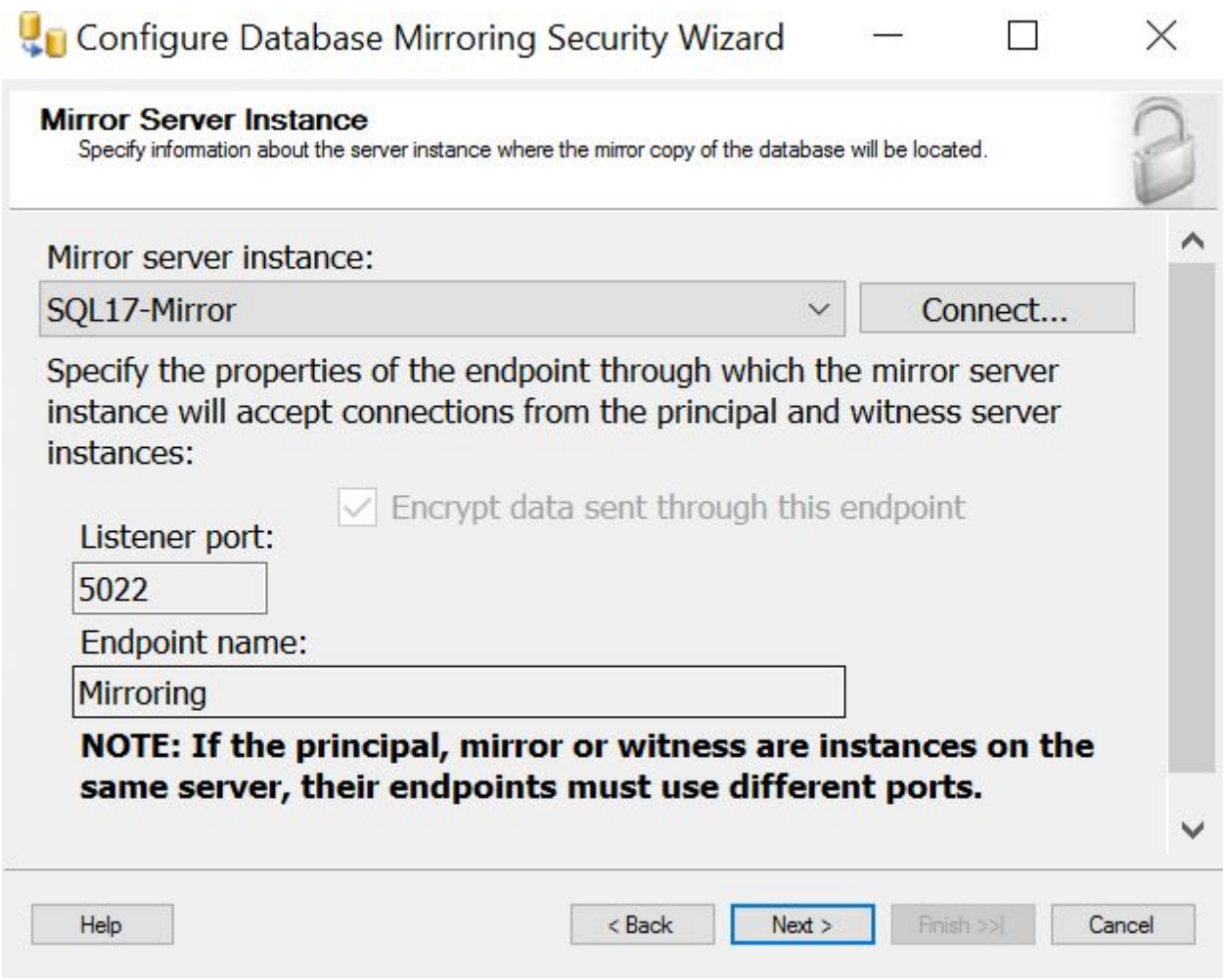


Figure 9 : Configuring Database Mirroring Security Wizard

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

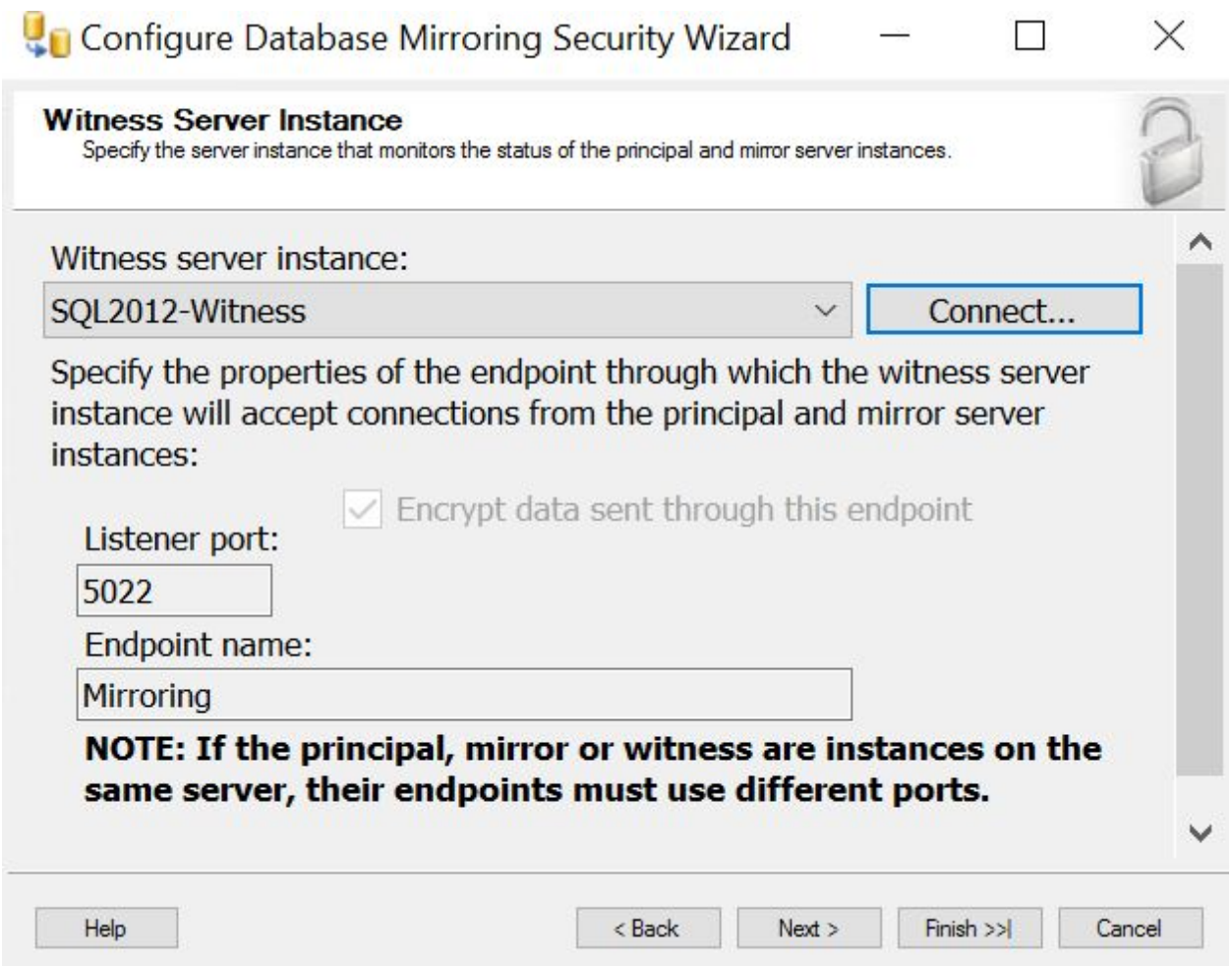


Figure 10 : Configuring Database Mirroring Security Wizard

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

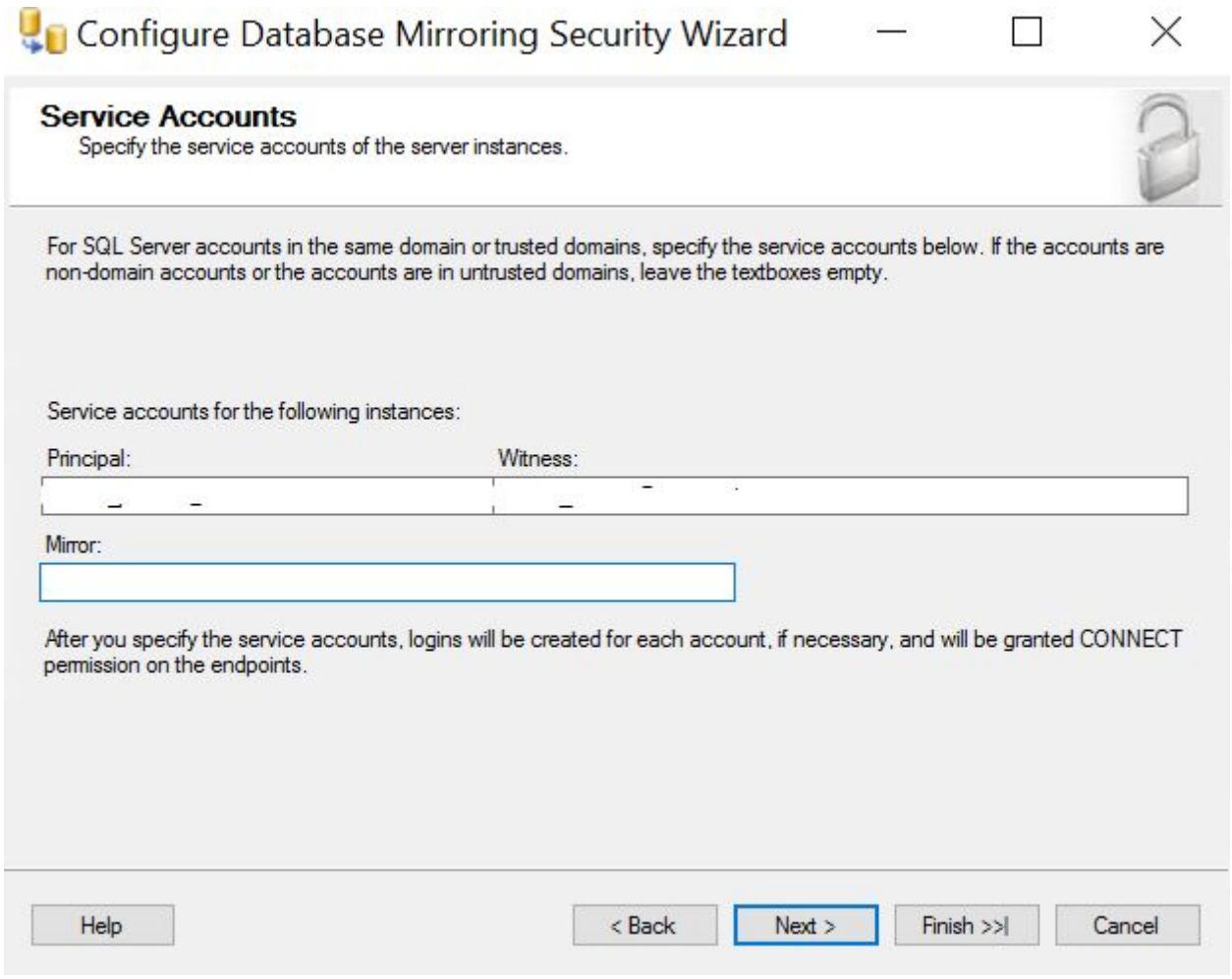


Figure 11 : Configuring Database Mirroring Security Wizard

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

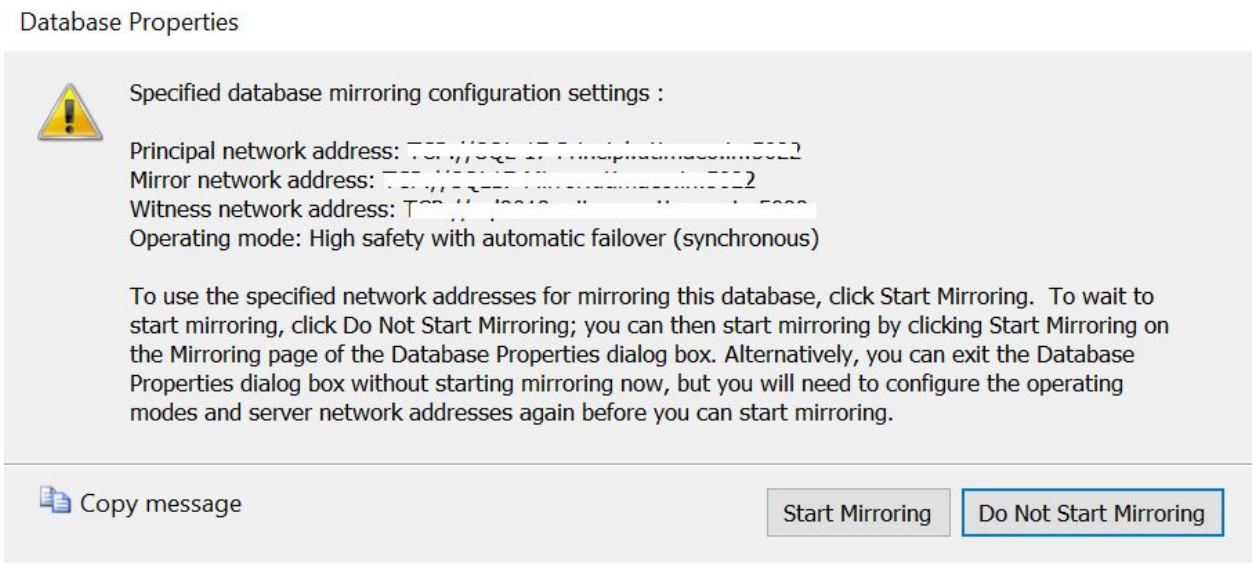


Figure 12 : Database Properties Wizard

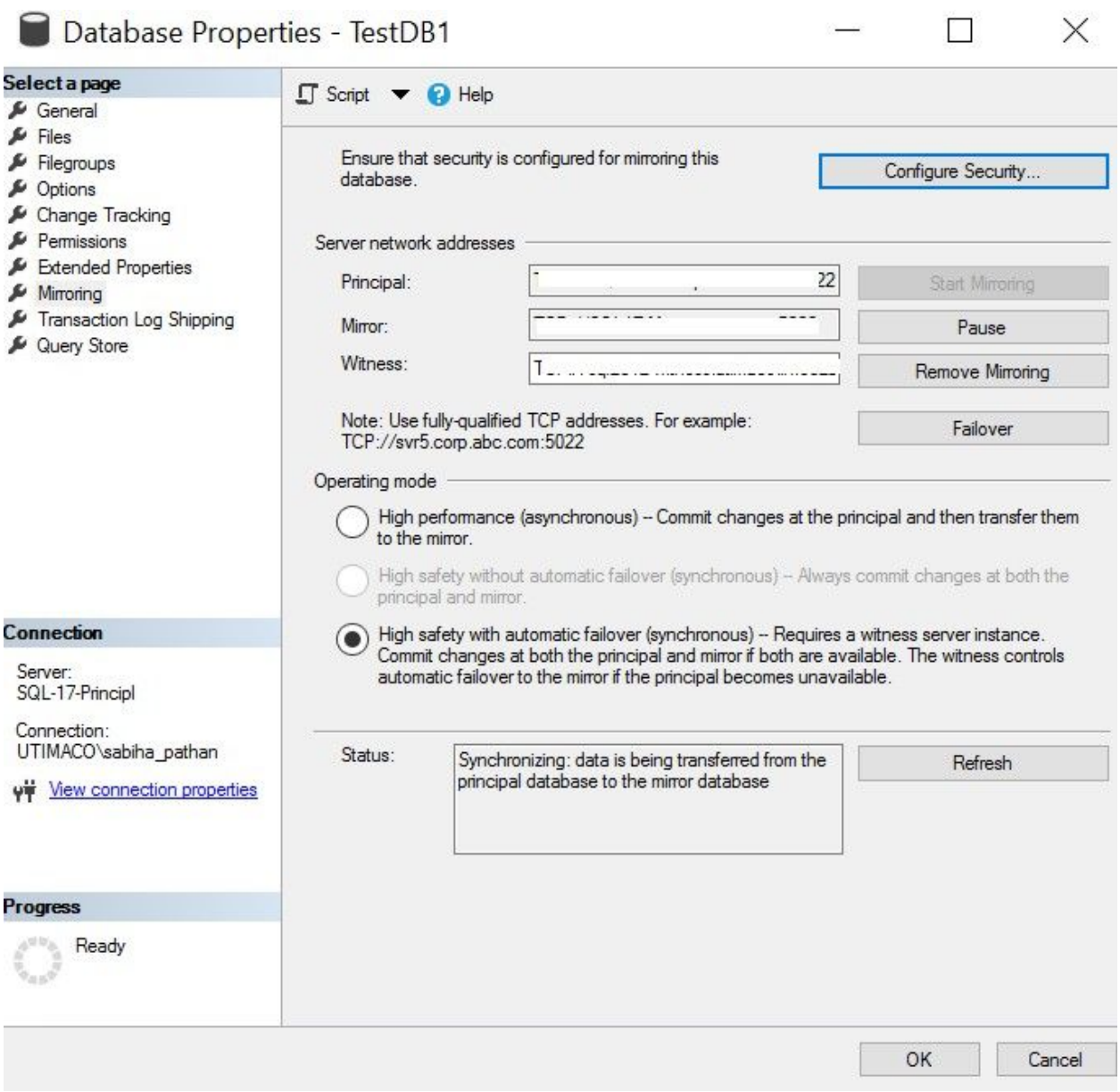


Figure 13 : Database Properties Wizard

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

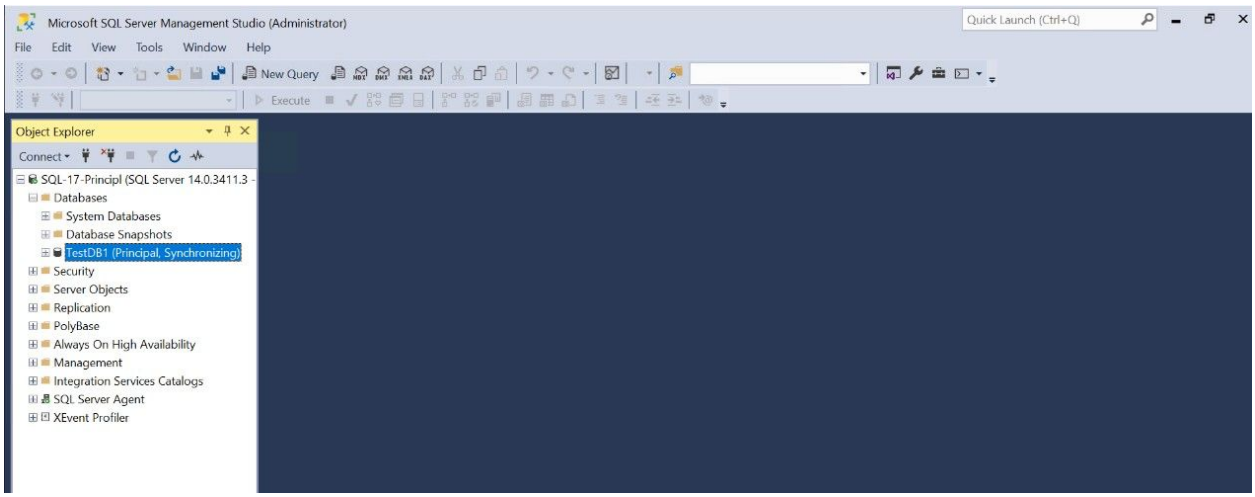


Figure 14 : Database Properties Wizard

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

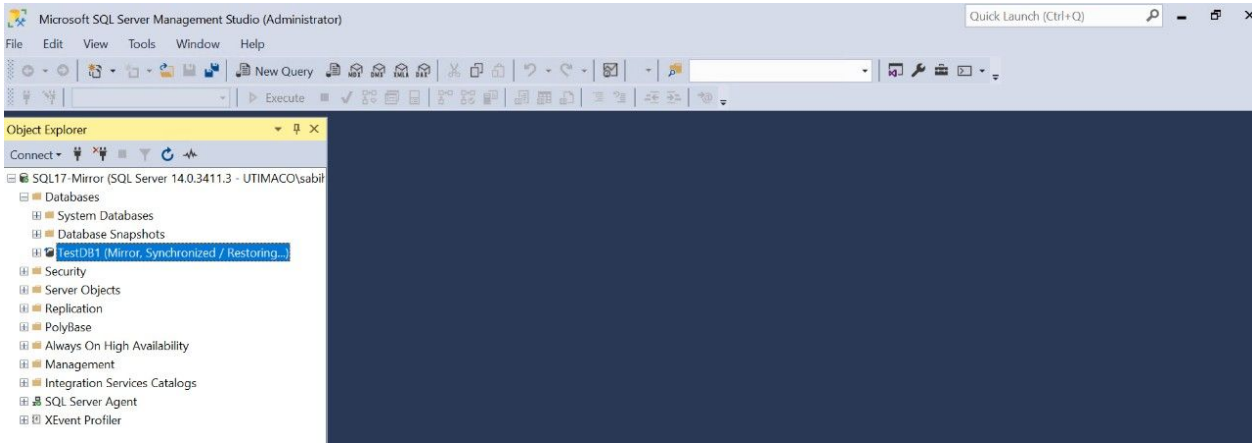


Figure 15 : Database Properties Wizard

6.1.5.3 Database Mirroring with Utimaco HSM

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

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

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



RSA algorithm is not supported in FIPS mode.

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

```
USE TestDB1;
GO
CREATE ASYMMETRIC KEY tdekey
FROM PROVIDER utimaco
WITH ALGORITHM = RSA_2048, PROVIDER_KEY_NAME = 'tdekey',
CREATION_DISPOSITION=CREATE_NEW;
GO
```

3. Insert the data into the table.

```
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'));
GO
```

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

6.1.5.3.1 To Test Failover Scenario

1. Select the database. Right-click and select **Tasks**, then select **Mirror**.

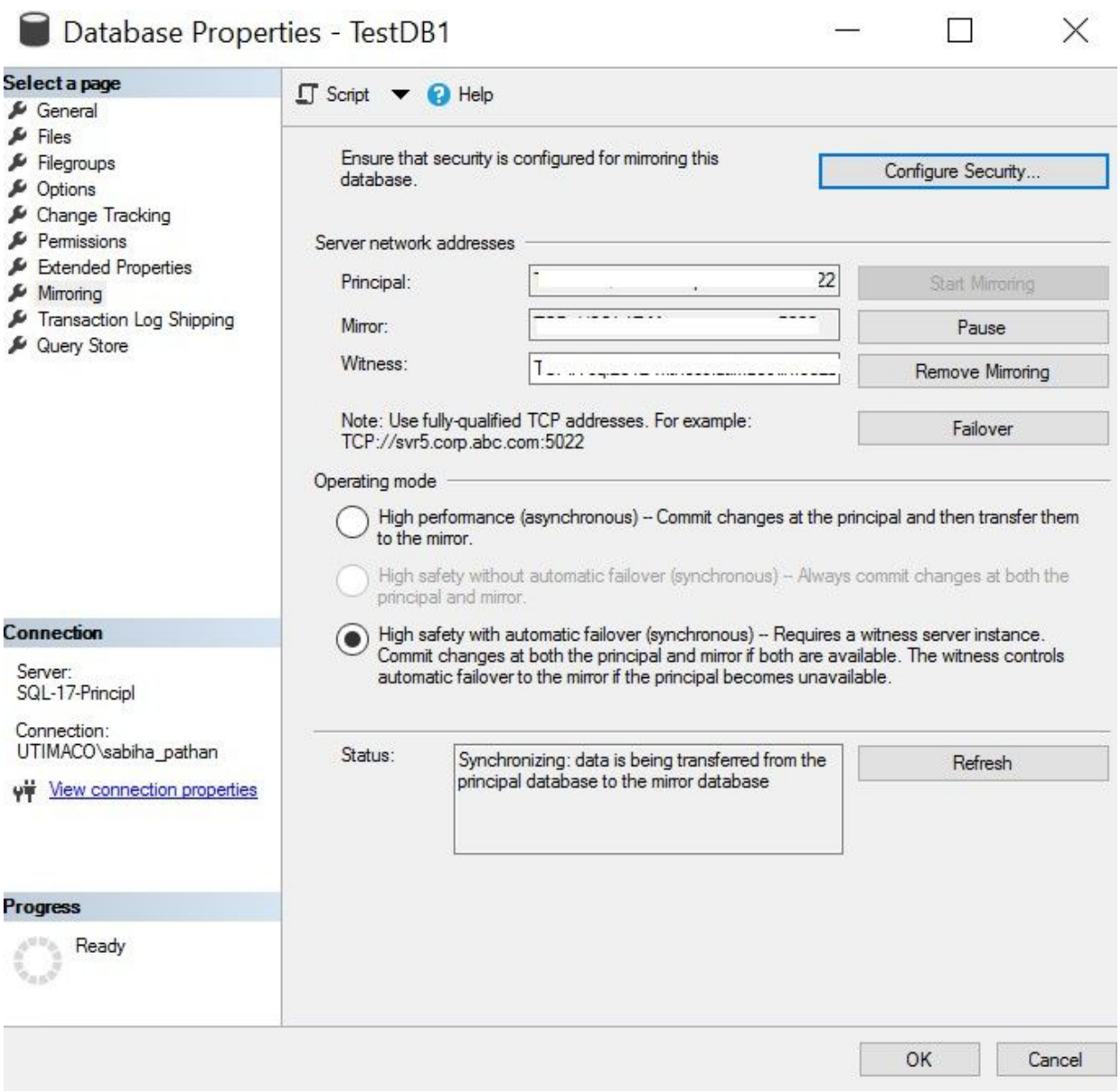


Figure 16 : Database Properties Wizard

2. Select **Failover** and then select **Yes** on the next Dialogue box that appears.
3. This will make the current Principal Server a Mirror Server, and vice versa.
4. Click **Refresh** to reflect the changes.
5. On the new Principal Server, run the following query to verify that Database Mirroring with Utimaco HSM is working as expected.

```
USE TestDB1
GO
SELECT FirstName, SecondName, CONVERT (varchar, DECRYPTBYASYMKEY
(ASYMKEY_ID('RSA2048Key'), CardNumber))
AS 'CardNumber' FROM Customers;
GO
```

7 Troubleshooting

7.1 Common Issues and How to Resolve Them

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. Check that `C:\Program Files\Utimaco\SecurityServer\Lib` is on system PATH. 4. 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 the SecurityServer Application is up and running and connected to the HSM. 2. Check if the Configuration File is pointing towards the IP Address of the 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 the Configuration File is **Disabled**. 2. Verify that the correct Credentials are mapped to the Microsoft SQL Server login.

Error	*Diagnosis*
Server principal 'test/admin' has no credential associated with cryptographic provider 'utimaco'.	Map a Credential to the Microsoft SQL Server login.
Key Type property of the key returned by the EKM provider doesn't match the expected value.	1. This error occurs when the user is trying to create an Asymmetric key or a Symmetric key with an unsupported algorithm. 2. The RSA Algorithm is not supported in FIPS Mode.
When viewing the data from the table, the output displays the NULL value instead of encrypted/decrypted data.	Check if you are using the correct encrypt/ decrypt key, or the user may not have permission to access the key.

Table 8: List of Errors and their Diagnoses

7.2 Log Locations and Interpretation

The following logs can be reviewed to check for errors:

- Utimaco EKM provider log: `C:\ProgramData\Utimaco\EKM\cssqlekm.log`
    - This log contains errors related to the EKM provider. These include errors with the connection to the HSM, with the Keystore, or during the execution of cryptographic operations.

Error	Diagnosis
`Utimaco::HSM::ODBCConnection::connect | E: Database Connection Failed: -1 > (-1) IM002:0 [Microsoft][ODBC Driver Manager] Data source name not found and no default driver specified`	Error connecting to the ODBC external Keystore. Check the ODBC connector using the ODBC Data Source Administrator or modify the `cssqlekm.cfg` configuration file.
`EkmDB::generateKey | E: error> key generate failed. errorcode is 3187671048. error is ExtStorageException(Error::INVALID_PARAMETER = 0xbe000008) thrown in Utimaco::HSM::KeyStorePluginODBC::find_keys_intern (-1) 42S02:208 [Microsoft][ODBC Driver 17 for SQL Server][SQL Server]Invalid object name 'hsm_keys'.`	The ODBC External Keystore has not been initialized, please refer to Configuration of the External Keystore.
`EkmDB::generateKey | E: key already exists`	A key with the same name already exists.

Table 9: List of Common Errors on the EKM Provider Log

- Microsoft SQL Server Error Log: `C:\Program Files\Microsoft SQL Server\MSSQL16.MSSQLSERVER\MSSQL\Log\ERRORLOG`
    - This log contains errors during the operation of the Microsoft SQL Server.

8 Appendices

8.1 References

Reference	*Title*	*Document Number*
[CSADM]	u.trust Anchor – csadm Manual / Utimaco IS GmbH	2021-0037
[UTAADMIN]	u.trust Anchor - Administration Manual / Utimaco IS GmbH	2020-0035

Table 10: List of References

8.2 Command Summary

Task	*Command*
Connect to MSSQL Database using Windows Login	`sqlcmd -S <IP> -E`
Connect to MSSQL Database using User and Password	`sqlcmd -S <IP> -U <user>`
Create DB Schema for External Key Storage in MSSQL	`cxitool dbConnString="DSN=<ODBC>;Uid=<Username>;Pwd=<Password>" CreateDBSchema=mssql`
Create Key using an SDB external keystore	`cxitool Dev=<port@IP> LogonPass=<user>,<password> keystoretype=SDB keystoreparam="<SDB_PATH>" group="" | Name=<KEY_NAME> Usage=ENCRYPT,DECRYPT,SIGN,VERIFY spec=0 generatekey=AES,256`

Table 11: List of Commands