MICROSOFT SQL SERVER

Integration Guide

Applicable Devices:
KMES Series 3
Applicable Versions:
6.3.1.x

TABLE OF CONTENTS

[1] DOCUMENT INFORMATION 3
[1.1] Document Overview 3
[1.2] Application Description 3
[2] PREREQUISITES 5
[3] KMES SERIES 3 CONFIGURATION 6
[3.1] Configure TLS communication between the KMES Series 3 and the Microsoft SQL Server instance 6
[3.2] Create an identity for MS SQL Server and grant it the required permissions 10
[3.3] Enable the Host API commands required for the Microsoft SQL Server operation 11
[3.4] Grant the MS SQL Server role "Use" permissions on the CA tree 12
[4] INSTALL AND CONFIGURE FUTUREX CLIENT LIBRARY (FXCL) EKM 13
[4.1] Installing FXCL EKM 13
[4.2] Configuring FXCL EKM 13
[5] CONFIGURING EKM IN MICROSOFT SQL SERVER 15
[5.1] Enable the EKM provider option 15
[5.2] Register the FXCL EKM provider 15
[6] ENABLING TDE IN MICROSOFT SQL SERVER USING EKM 16
APPENDIX A: XCEPTIONAL SUPPORT 19

[1] DOCUMENT INFORMATION

[1.1] DOCUMENT OVERVIEW

The purpose of this document is to provide information regarding the configuration of the Futurex KMES Series 3 with Microsoft SQL Server Transparent Data Encryption (TDE) using EKM libraries. For additional questions related to your KMES Series 3 device, see the relevant user guide.

[1.2] APPLICATION DESCRIPTION

[1.2.1] About Microsoft SQL Server

Microsoft SQL Server is a relational database management system (RDBMS) used for large-scale online transaction processing (OLTP), data warehousing, and e-commerce applications. It is also a business intelligence platform for data integration, analysis, and reporting solutions.

[1.2.2] About Transparent Data Encryption (TDE)

From Microsoft's documentation website: "Transparent Data Encryption (TDE) encrypts SQL Server data files. This encryption is known as encrypting data at rest.

To help secure a database, you can take precautions like:

- Designing a secure system.
- Encrypting confidential assets.
- Building a firewall around the database servers.

But a malicious party who steals physical media like drives or backup tapes can restore or attach the database and browse its data.

One solution is to encrypt sensitive data in a database and use a certificate to protect the keys that encrypt the data. This solution prevents anyone without the keys from using the data. But you must plan this kind of protection in advance.

TDE does real-time I/O encryption and decryption of data and log files. The encryption uses a database encryption key (DEK). The database boot record stores the key for availability during recovery. The DEK is a symmetric key. It's secured by a certificate that the server's master database stores or by an asymmetric key that an EKM module protects.

TDE protects data at rest, which is the data and log files. It lets you follow many laws, regulations, and guidelines established in various industries. This ability lets software developers encrypt data by using AES and 3DES encryption algorithms without changing existing applications."

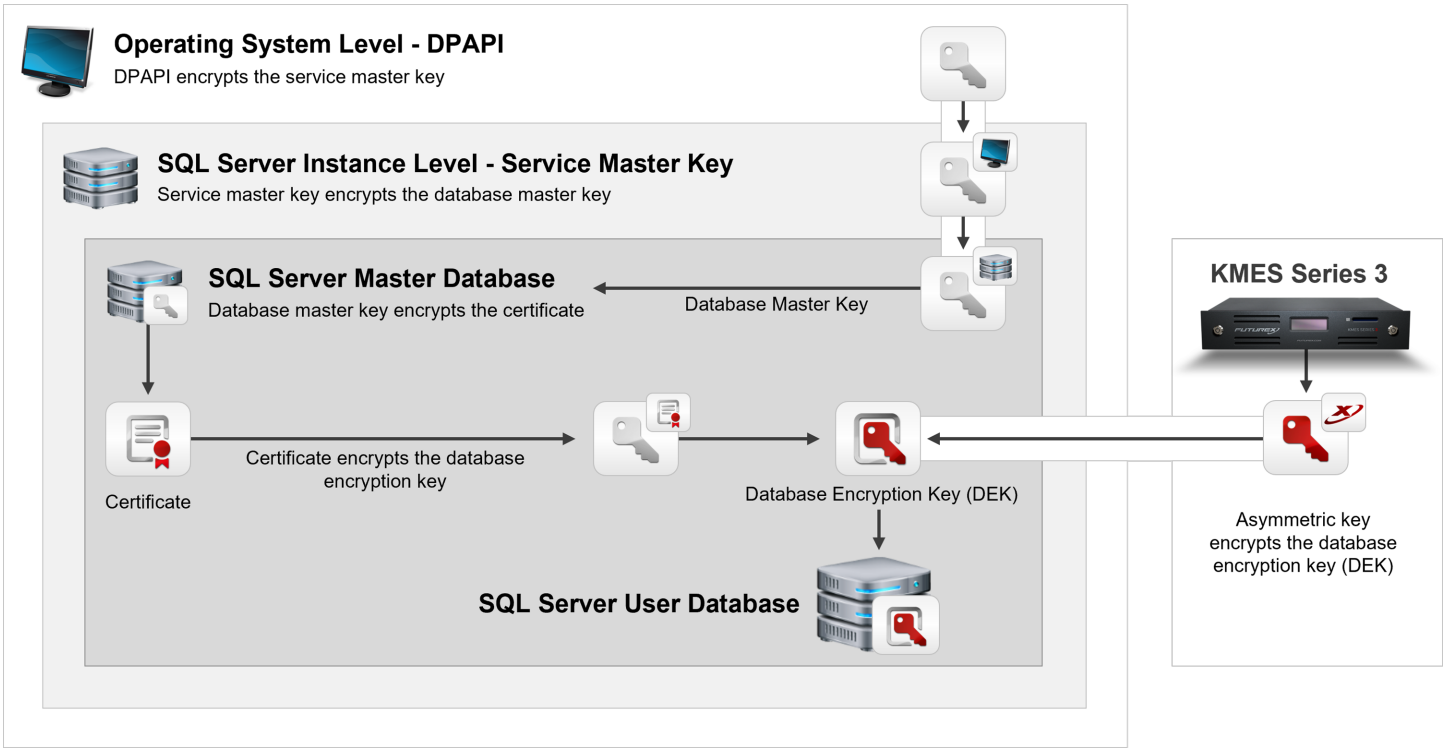
[1.2.3] Encryption Hierarchy and Integration with the KMES Series 3

Through Extensible Key Management (EKM), Microsoft SQL Server can utilize a Futurex KMES Series 3 for key management and encryption acceleration.

In this configuration, data can be encrypted using encryption keys that only the database user has access to on the external EKM/HSM module.

The figure below shows the architecture of TDE encryption, as well as the relationship between the SQL Server database master key and the KMES Series 3.

NOTE: Only the database-level items (i.e., the database encryption key) are user-configurable when you use TDE on SQL Database.



[2] PREREQUISITES

Supported Hardware:

- KMES Series 3, 6.3.1.x and above

Supported Operating Systems:

- Windows 7 and above

Other:

- OpenSSL
- Microsoft SQL Server
- Microsoft SQL Server Management Studio

[3] KMES SERIES 3 CONFIGURATION

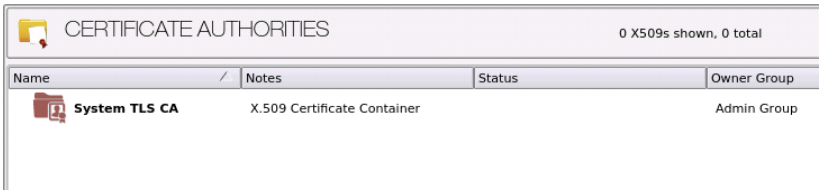
The first half of this section will cover the steps needed to configure TLS communication between the KMES and the Microsoft SQL Server instance. The second half of this section will cover general configurations that need to be made on the KMES Series 3 to allow Microsoft SQL Server to integrate with the KMES, via the FXCL EKM library, for Transparent Data Encryption.

Every step in this section requires being logged in to the KMES Series application interface with the default Admin identities.

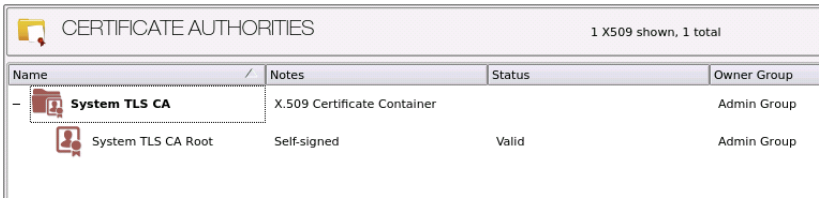
[3.1] CONFIGURE TLS COMMUNICATION BETWEEN THE KMES SERIES 3 AND THE MICROSOFT SQL SERVER INSTANCE

[3.1.1] Create a Certificate Authority (CA)

1. Navigate to *PKI* -> *Certificate Authorities*, then click the **Add CA...** button at the bottom of the page.
2. In the *Certificate Authority* dialog, enter a name for the Certificate Container, leave all other fields as the default values, then click **OK**.
3. The Certificate Container that was just created will be listed now in the Certificate Authorities menu.



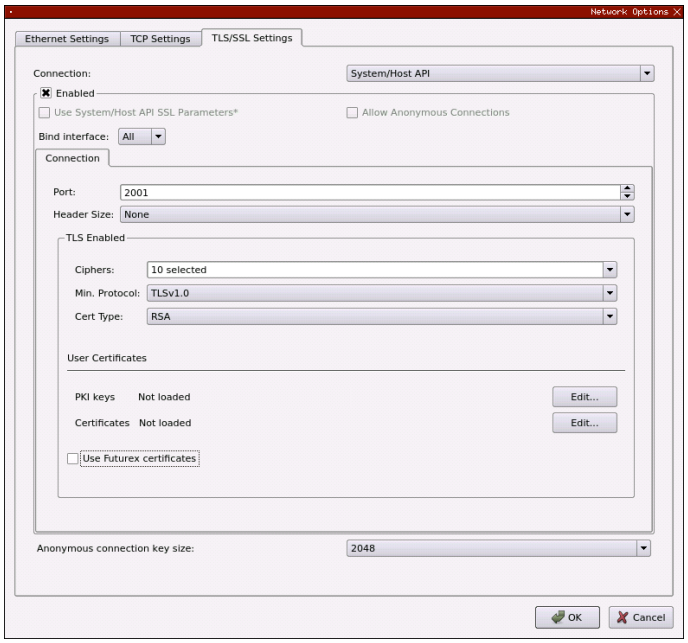
4. Right-click on the Certificate Container again and select **Add Certificate -> New Certificate...**
5. In the *Subject DN* tab, change the Preset dropdown to Classic and specify a Common Name for the certificate, such as "System TLS CA Root".
6. In the *Basic Info* tab, leave all fields set to the default values.
7. In the *V3 Extensions* tab, select the "Certificate Authority" profile, then click **OK**.
8. The root CA certificate will be listed now under the previously created Certificate Container.



[3.1.2] Generate a CSR for the System/Host API connection pair

1. Navigate to Administration -> *Configuration* -> *Network Options*.
2. In the *Network Options* dialog, select the *TLS/SSL Settings* tab.

3. Under the **System/Host API** connection pair, uncheck "Use Futurex certificates", then click **Edit...** next to PKI keys in the User Certificates section.

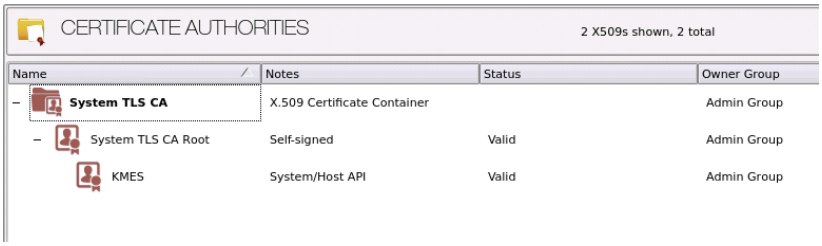


4. In the *Application Public Keys* dialog, click **Generate...**
5. There will be a warning stating that SSL will not be functional until new certificates are imported. Select **Yes** if you wish to continue.
6. In the *PKI Parameters* dialog, leave the default values set and click **OK**.
7. It should show that an HSM trusted asymmetric key is loaded now in the *Application Public Keys* dialog. If this is the case, click **Request...**
8. In the *Subject DN* tab, set a Common Name for the certificate, such as "KMES".
9. In the *V3 Extensions* tab, select the "TLS Server Certificate" profile.
10. In the *PKCS #10 Info* tab, select a save location for the CSR, then click **OK**.
11. There should be a message stating that the certificate signing request was successfully written to the file location that was selected. Click **OK**.
12. Click **OK** again to save the *Application Public Keys* settings.
13. In the main *Network Options* dialog, it should now say "Loaded" next to **PKI keys** for the System/Host API connection pair. Click **OK** to save.

[3.1.3] Sign the System/Host API CSR

1. Navigate to the *PKI -> Certificate Authorities* menu.
2. Right-click on the root CA certificate created in section 3.1.1, then select **Add Certificate** -> **From Request...**
3. In the file browser, find and select the CSR that was generated for the System/Host API connection pair.

4. Once loaded, none of the settings need to be modified for the certificate. Click **OK**.
5. The signed System/Host API certificate should now show under the root CA certificate on the *Certificate Authorities* page.



[3.1.4] Export the Root CA certificate

1. Navigate to the *PKI -> Certificate Authorities* menu.
2. Right-click on the "System TLS CA Root" certificate, then select **Export** -> **Certificate(s)...**
3. In the *Export Certificate* dialog, change the encoding to "PEM", then click **Browse...**
4. In the file browser, navigate to the location where you want to save the Root CA certificate. Specify a unique name for the file, such as "root_cert.pem", then click **Open**.
5. Click **OK**. A message box will pop up stating that the PEM file was successfully written to the location that you specified.

NOTE: The Root CA certificate will be configured later inside of the FXCL EKM configuration file.

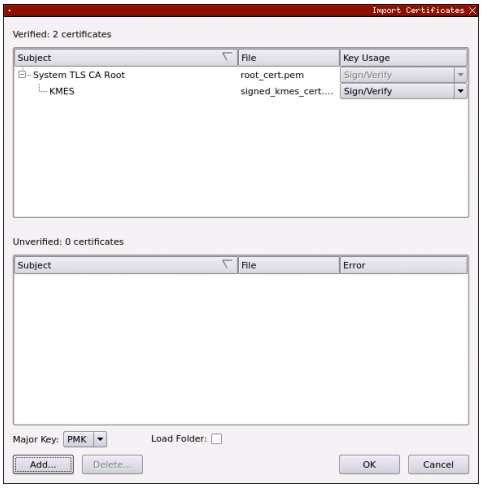
[3.1.5] Export the signed System/Host API certificate

1. Navigate to the *PKI -> Certificate Authorities* menu.
2. Right-click on the "KMES" certificate, then select **Export** -> **Certificate(s)...**
3. In the *Export Certificate* dialog, change the encoding to "PEM", then click **Browse...**
4. In the file browser, navigate to the location where you want to save the signed System/Host API certificate. Specify a unique name for the file, such as "signed_kmes_cert.pem", then click **Open**.
5. Click **OK**. A message box will pop up stating that the PEM file was successfully written to the location that you specified.

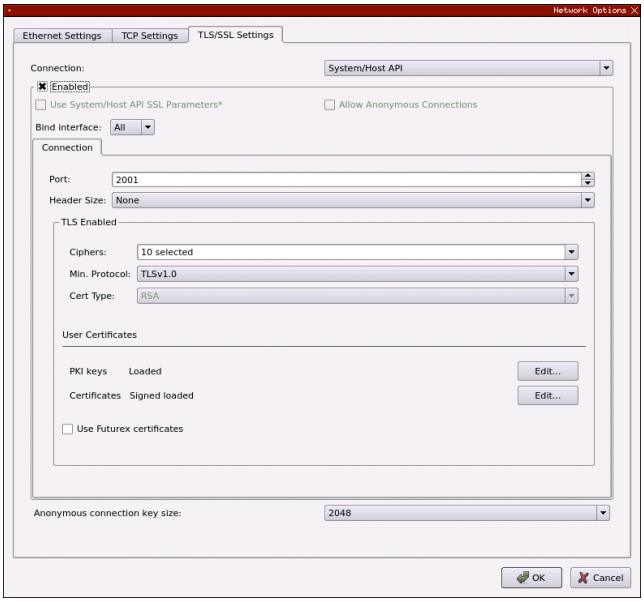
[3.1.6] Load the exported certificates into the System/Host API connection pair

1. Navigate to *Administration -> Configuration -> Network Options*.
2. In the *Network Options* dialog, select the *TLS/SSL Settings* tab.
3. Click **Edit...** next to Certificates in the User Certificates section.
4. Right-click on the **System/Host API SSL CA** X.509 Certificate Container, then select **Import...**
5. Click **Add...** at the bottom of the *Import Certificates* dialog.

6. In the file browser, find and select both the root CA certificate and the signed System/Host API certificate, then click **Open**. The certificate chain should appear as shown below. Click **OK** in two consecutive dialogs to save the changes.



7. In the *Network Options* dialog, the System/Host API connection pair should show "Signed loaded" next to Certificates in the User Certificates section, as shown below:



8. Click **OK** to save and exit the Network Options dialog.

[3.1.7] Issue a client certificate for Microsoft SQL Server

NOTE: The client certificate that is being created for Microsoft SQL Server will be configured later inside of the FXCL EKM configuration file.

1. Navigate to the *PKI -> Certificate Authorities* menu.
2. Right-click on the **System TLS CA Root** certificate and select **Add Certificate** -> **New Certificate...**.

3. In the *Subject DN* tab, change the Preset dropdown to Classic and set "SqlServer" as the Common Name for the certificate.
4. In the *Basic Info* tab, leave all fields set to the default values.
5. In the *V3 Extensions* tab, select the "TLS Client Certificate" profile, then click **OK**.
6. The Microsoft SQL Server client certificate will be listed now under the **System TLS CA Root** certificate.

[3.1.8] Export the Microsoft SQL Server client certificate as PKCS #12 file

NOTE: To be able to perform the steps below you must go to *Administration -> Configuration -> Options* and enable the "Allow export of certificates using passwords" option.

1. Navigate to the *PKI -> Certificate Authorities* menu.
2. Right-click on the "SqlServer" certificate, then select **Export** -> **PKCS12...**
3. Make sure that the "Export Selected" option is selected, specify a unique name for the export file, such as "PKI.p12", then click **Next**.
4. Input a file password of your choosing, then click **Next**.

   NOTE: The P12 file password will be configured later inside of the FXCL EKM configuration file.
5. Click **Finish** to initate the export.

NOTE: The Microsoft SQL Server client certificate and the Root CA certificate that was exported in section 3.1.4 both need to be moved to the computer where the SQL Server is running. In the next section, they will be configured and used for TLS communication with the KMES Series 3.

[3.2] CREATE AN IDENTITY FOR MS SQL SERVER AND GRANT IT THE REQUIRED PERMISSIONS

A new role and identity need to be created for MS SQL Server on the KMES Series 3.

Create a new role

1. Navigate to the *Identity Management -> Roles* menu and add a new role. This will open the *Role Editor* dialog.
2. Name the role "MS SQL Server" and change the number of logins required to **1**. Leave all other fields under the *Info* tab set as the default values.
3. Under the *Permissions* tab, select the following permissions:
   - Certificate Authority -> Export
   - Cryptographic Operations -> Encrypt, Decrypt
   - Keys -> Add, Delete
4. Under the *Advanced* tab, set Allowed Ports to only **Host API**. Leave the other settings set to the default

values.

5. Click **OK** to finish creating the role.

Create a new identity and assign it the MS SQL Server role and Password authentication credentials

1. Navigate to the *Identity Management -> Identities* menu. Right-click and select *Add -> Client Application* to add a new identity. This will open the *Identity Editor* dialog.
2. Specify "SqlServer" in the Name field. Leave all other fields under the *Info* tab set as the default values.

   NOTE: The name that is set for this identity must match the Common Name that was set for the Microsoft SQL Server client certificate in section 3.1.7.
3. Under the *Assigned Roles* tab, select the **MS SQL Server** role.
4. Under the *Authentication* tab, click the **Add** button to add a new credential.
5. In the *Configure Credential* dialog, select **Password** in the **Type** dropdown. The **Provider** and **Mechanism** fields should populate with **Local Application** and **Password**, respectively. Select the **Change** and set a password for the credential, then click **Save**. Click **OK** to finish configuring the credential.
6. Remove the default API Key mechanism, leaving only the Password credential, and click **OK** to save.

[3.3] ENABLE THE HOST API COMMANDS REQUIRED FOR THE MICROSOFT SQL SERVER OPERATION

Because FXCL EKM will be connecting to the Host API port on the KMES, users must define which Host API commands will be enabled for execution by FXCL EKM. To set the enabled commands, complete the following steps:

1. Navigate to *Administration -> Configuration -> Host API Options*, enable the commands listed below, then click **Save**.

- **RKGP**: Export Asymmetric Key
- **RKLN**: Lookup Objects
- **RKDP**: Delete Asymmetric Key
- **RKLO**: Login User
- **RKCK**: Create Asymmetric Key
- **RKRE**: RSA Encrypt
- **RKRD**: RSA Decrypt
- **RKPK**: Pop Generated Key
- **CLKY -> get**: Retrieve HSM protected key

[3.4] GRANT THE MS SQL SERVER ROLE "USE" PERMISSIONS ON THE CA TREE

1. Navigate to the *PKI -> Certificate Authorities* menu.
2. Right-click on the CA container that was created in section 3.1.1, then select **Permission...**.
3. Grant the **MS SQL Server** role the "Use" permission, select "Apply to children recursively", then click **OK** to save.

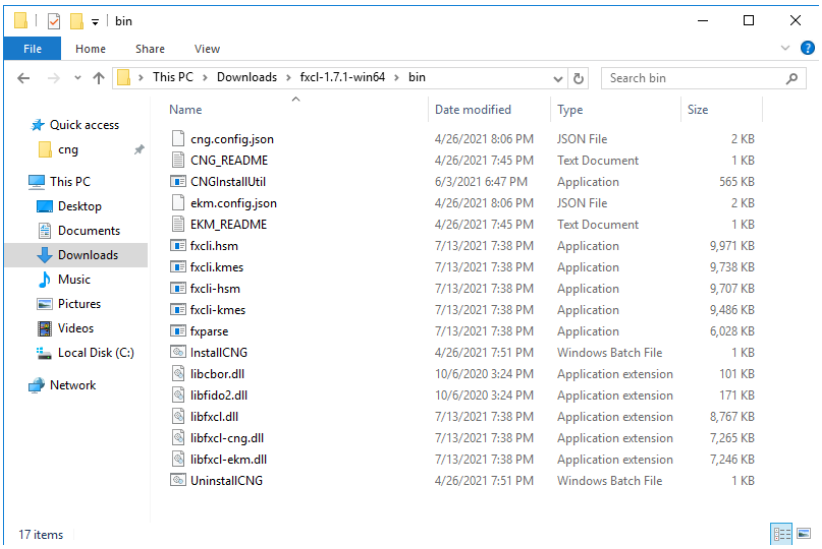
[4] INSTALL AND CONFIGURE FUTUREX CLIENT LIBRARY (FXCL) EKM

The Futurex Client Library, or FXCL, is a set of functions, offered using either Java (Java Native Interface) or C++, used by applications to access cryptographic processing and key management functionality.

[4.1] INSTALLING FXCL EKM

NOTE: To maintain system security, it is important to only install and operate copies of FXCL that are obtained directly from Futurex. These files will either be provided directly by a member of the Solutions Architect team or made available for download on the Futurex Portal or equivalent Futurex-operated file distribution platform.

1. Download or copy the `fxcl-x.x.x-win64.zip` file to the computer/server that will be running the Microsoft SQL Server instance.
2. Unzip the file in any directory, then navigate into the `fxcl-x.x.x-win64\bin` folder. There you will see the following files:



3. Copy the `ekm.config.json` and `libfxcl-ekm.dll` files to `C:\Program Files\Futurex\fxcl\kmes\ekm\`, then change the name of the `ekm.config.json` file to `config.json`.

[4.2] CONFIGURING FXCL EKM

1. Create a `Certs\` directory in `C:\` (i.e., `C:\Certs`). Then, copy the Root CA certificate PEM file (exported from the KMES in section 3.1.4) and the Microsoft SQL Server client certificate PKCS #12 file (exported from the KMES in section 3.1.8) into the `Certs\` folder.
2. Create a `Fxcl_Logs\` directory in `C:\` (i.e., `C:\Fxcl_Logs`). The FXCL EKM configuration file will be configured to output the FXCL EKM logs to the `Fxcl_Logs\` directory.
3. Edit the `config.json` file to point to the TLS connection certificates and network-connected KMES Series 3 device. An example `config.json` file is shown below:

```
{
    // Enables output via DebugOutputString
    // (default: false)
    // Note that regardless of this setting, output is
    // placed in the debug view while loading the config.
    "enable_debug_view": false,

    // A file to place logs into. Optional.
    // If not provided, no log file is made.
    "log_file": "C:\\Fxcl_Logs\\fxcl.log",

    // Level of logging to emit. Case insensitive.
    // possible values: None, Error, Info, Debug, Traffic (default: Info)
    "log_level": "traffic",

    // What kind of key storage unit is this?
    // possible values: kmes (default: kmes)
    // Not currently used, it always uses kmes.
    "driver": "kmes",

    // The host to connect to. Required.
    "host": "10.0.8.22:2001",

    // A PEM file containing a list of trusted CA certificates. Required.
    "ca": "C:\\Certs\\root_cert.pem",

    // A P12 file containing leaf certificate and key. Required.
    "p12": "C:\\Certs\\PKI.p12",

    // Password to unlock the P12 file. Optional.
    // If not given, assumes it doesn't need a password.
    "p12_pass": "safest"
}
```

NOTE: The `root_cert.pem` file is the Root CA certificate exported in section 3.1.4 and the `PKI.p12` file is the Microsoft SQL Server client certificate exported as a PKCS #12 file in section 3.1.8.

[5] CONFIGURING EKM IN MICROSOFT SQL SERVER

This section will cover the steps needed to enable EKM in Microsoft SQL Server and register the FXCL EKM provider.

[5.1] ENABLE THE EKM PROVIDER OPTION

In order to use the FXCL EKM provider, the EKM provider option must first be enabled on the SQL Server.

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

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

[5.2] REGISTER THE FXCL EKM PROVIDER

1. Open a new Query window in SQL Server Management Studio and execute the following:

```
CREATE CRYPTOGRAPHIC PROVIDER FxclEkmProvider
    FROM FILE = 'C:\Program Files\Futurex\fxcl\kmes\ekm\libfxcl-ekm.dll';
GO
```

[6] ENABLING TDE IN MICROSOFT SQL SERVER USING EKM

This section will cover the steps needed to enable transparent data encryption (TDE) in Microsoft SQL Server to protect a database encryption key by using an asymmetric key stored on the KMES Series 3.

NOTE: All of the following commands need to be run inside a Query window in SQL Server Management Studio.

1. Create a credential that will be used by system administrators.

```
CREATE CREDENTIAL EkmCredential
       WITH
               IDENTITY = 'SqlServer',
               SECRET = 'safest'
       FOR CRYPTOGRAPHIC PROVIDER FxclEkmProvider;
GO
```

   NOTE: The values set in the `IDENTITY` and `SECRET` fields should be the name and password of the user created on the KMES that is specified in the FXCL EKM configuration file.

2. Add the credential to a high privileged user such as your own domain login in the format [DOMAIN\login[.

```
ALTER LOGIN [WIN-Q4E6RG9BSOL\Administrator]
       ADD CREDENTIAL EkmCredential;
GO
```

3. Create an asymmetric key stored inside the FXCL EKM provider.

```
USE master;
GO
CREATE ASYMMETRIC KEY EkmAsym
       FROM PROVIDER FxclEkmProvider
       WITH
               ALGORITHM = RSA_1024,
               PROVIDER_KEY_NAME = 'EkmAsym';
GO
```

4. Create a credential that will be used by the Database Engine.

```
CREATE CREDENTIAL EkmEngineCredential
       WITH
               IDENTITY = 'SqlServer',
               SECRET = 'safest'
       FOR CRYPTOGRAPHIC PROVIDER FxclEkmProvider;
GO
```

   NOTE: The values set in the `IDENTITY` and `SECRET` fields should be the name and password of the user created on the KMES that is specified in the FXCL EKM configuration file.

5. Create a login that will use the asymmetric key stored inside the FXCL EKM provider.

```
CREATE LOGIN EkmLogin
    FROM ASYMMETRIC KEY EkmAsym;
GO
```

6. Set the login to be able to use the database engine credential.

```
ALTER LOGIN EkmLogin
    ADD CREDENTIAL EkmEngineCredential;
GO
```

7. Create a new example database, add a table to it, then insert information into the table.

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

```
CREATE DATABASE exampleDB;
GO
USE exampleDB;
GO
CREATE TABLE users (
       id INT NOT NULL PRIMARY KEY,
       name VARCHAR(64),
       password VARCHAR(128),
       ssn VARCHAR(32)
);
GO
INSERT INTO users (id, name, password, ssn) VALUES (1, 'SomeGuy', 'blah', '000-00-0000'), (2,
'SomeGal', 'password', '000-00-0000'), (3, 'TestUser', 'test123', '000-00-0000');
GO
```

8. Create a database encryption key for the 'exampleDB' database.

```
USE exampleDB;
GO
CREATE DATABASE ENCRYPTION KEY
       WITH
               ALGORITHM = AES_128
       ENCRYPTION BY SERVER ASYMMETRIC KEY EkmAsym;
GO
```

9. Enable transparent data encryption on the 'exampleDB' database.

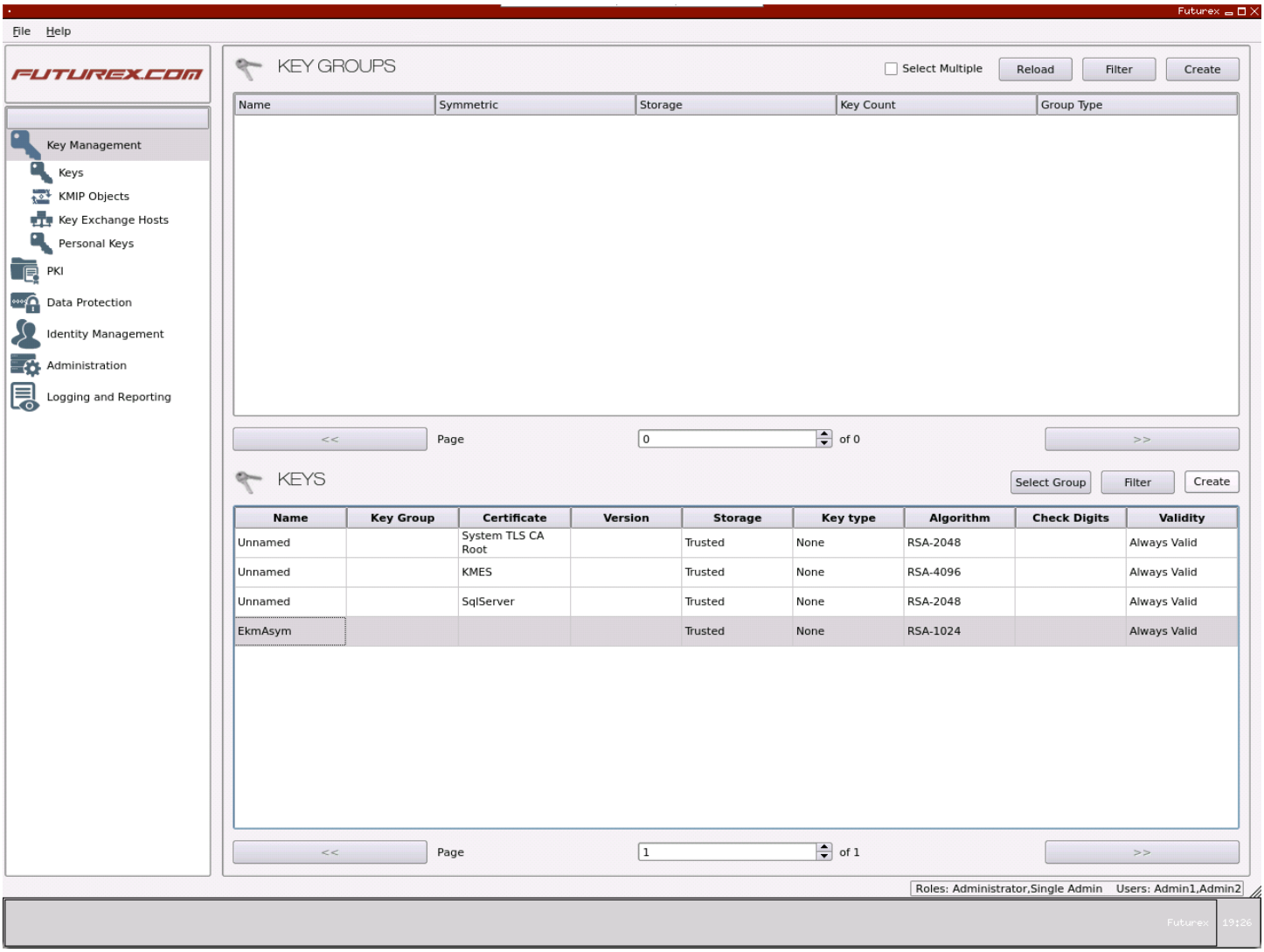
```
ALTER DATABASE exampleDB
       SET ENCRYPTION ON;
GO
```

10. Check if data can be decrypted.

    NOTE: Restart SQL Server service with the KMES Series 3 offline, then check if the following command fails. If it does, then TDE is set up correctly. If the KMES is online, the command should succeed.

```
USE exampleDB;
SELECT * FROM users;
```

NOTE: The asymmetric key that is created on the KMES and used for encrypting the Database Encryption Key (DEK) can be viewed on the *Key Management -> Keys* menu in the KMES application interface.



APPENDIX A: XCEPTIONAL SUPPORT

In today's high-paced environment, we know you are looking for timely and effective resolutions for your mission-critical needs. That is why our Xceptional Support Team does whatever it takes to ensure you have the best experience and support possible. Every time. Guaranteed.

- 24x7x365 mission critical support
- Level 1 to level 3 support
- Extremely knowledgeable subject matter experts

At Futurex, we strive to supply you with the latest data encryption innovations as well as our best-in-class support services. Our Xceptional Support Team goes above and beyond to meet your needs and provide you with exclusive services that you cannot find anywhere else in the industry.

- Technical Services
- Onsite Training
- Virtual Training
- Customized Consulting
- Customized Software Solutions
- Secure Key Generation, Printing, and Mailing
- Remote Key Injection
- Certificate Authority Services

Toll-Free: 1-800-251-5112

E-mail: support@futurex.com

ENGINEERING CAMPUS

864 Old Boerne Road

Bulverde, Texas, USA 78163

Phone: +1 830-980-9782

+1 830-438-8782

E-mail: info@futurex.com

XCEPTIONAL SUPPORT

24x7x365

Toll-Free: 1-800-251-5112

E-mail: support@futurex.com

SOLUTIONS ARCHITECT

E-mail: solutions@futurex.com