Microsoft SQL Server  
SSL Setup Guide

Author: Eitan Blumin

Madeira Data Solutions

[www.madeiradata.com](http://www.madeiradata.com)

Contents

[Produce CA Certificates 2](#_Toc118097773)

[Import the CA Certificate on the SQL Server machine 3](#_Toc118097774)

[Enable the SSL Setting in SQL Server 3](#_Toc118097775)

[Restart the SQL Server service 3](#_Toc118097776)

[Configuring the SQL Server clients to use encrypted connections 3](#_Toc118097777)

[Check if connections are encrypted 4](#_Toc118097778)

[Try to connect using the Fully Qualified Domain Name 4](#_Toc118097779)

[Check the connection string of your application 4](#_Toc118097780)

[SSL encryption for failover clustering in SQL Server 4](#_Toc118097781)

[Rollback: Disabling the SSL Setting in SQL Server 5](#_Toc118097782)

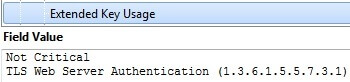
[Troubleshooting 5](#_Toc118097783)

[Resources 6](#_Toc118097784)

# Produce CA Certificates

The first step to secure the connections is to obtain a security certificate. These certificates need to be generated by IT, or by a trusted CA (Certification Authority). There are several requirements which should be fulfilled by the certificate:

* It must be valid thus the current system date and time should be between the **Valid From** and **Valid To** properties of the certificate.
* The **Common Name (CN)** in the **Subject** property of the certificate must be the same as the fully qualified domain name (FQDN) of the server computer(s).
* It must be issued for server authentication so the **Extended Key Usage** property of the certificate should include '*Server Authentication (1.3.6.1.5.5.7.3.1)*' (see below).



* It must be created by using the **KeySpec** option of '*AT\_KEYEXCHANGE*'.

It is possible to use self-signed certificates, but this should be done for test purposes only, and must be avoided in production environments.

# Import the CA Certificate on the SQL Server machine

To import a certificate, follow these steps on each relevant machine:

1. Run **certlm.msc** (Certificates – Local Machine) as Administrator
2. Right-click on the **Personal** folder, point to **All Tasks**, and then click **Request New Certificate...**
3. Click **Next** in the **Certificate Request Wizard** dialog box. Select certificate type 'Computer'.
4. You can enter a friendly name in text box if you want or leave it blank, then complete the wizard.
5. Now you should see the certificate in the folder with the fully qualified computer domain name.

# Enable the SSL Setting in SQL Server

You can configure SSL using the SQL Server Configuration Manager. First, you should run SQL Server Configuration Manager under the SQL Server service account. The only exception is if the service is running as LocalSystem, NetworkService, or LocalService, in this case you can use an administrative account.

1. Expand **SQL Server Network Configuration** and right-click on **Protocols for <YourMSSQLServer>**, then click **Properties**.
2. On the **Certificate** tab, select the certificate you would like to use.
3. On the **Flags** tab, select **No** in the **ForceEncryption** box, then click **OK**.

# Restart the SQL Server service

Restarting the SQL Server service in production environment obviously must be done very carefully and thoughtfully.

# Configuring the SQL Server clients to use encrypted connections

You should export the certificate from your SQL Server and install it on the client computer to establish the encryption.

1. Open the MMC Certificates Snap-in as described above.
2. Right-click the **Certificate**, point to **All Tasks**, and then click **Export**.
3. Complete the **Certificate Export Wizard**, storing the certificate file in a selected location.
4. Copy the certificate to the client computer.
5. Use the MMC Certificates Snap-in on the client computer to install the exported certificate file: Right-click the **Trusted Root Certification Authorities** folder, point to **All Tasks**, and then click **Import** and follow the steps.
6. In the SQL Server Configuration Manager right-click **SQL Server Native Client Configuration**, and then click **Properties**.
7. On the **Flags** tab, select **Yes** in the **ForceEncryption** box, then click **OK**.

The client machine should trust the certificate so there are two options:

* The SQL Server's certificate should be installed on the client machine to establish a direct trust.
* The certificate of the root certificate authority and the intermediate/chain certificates should all be trusted. This way you can take advantage of the chain of trust, the core principle of SSL certificate hierarchy.

You can also encrypt the connection from SQL Server Management Studio (mainly for testing purposes):

1. Click **Options** in the **Connect to Server** dialog.
2. On the **Connection Properties** tab, tick the **Encrypt connection** checkbox.

# Check if connections are encrypted

You can query the *sys.dm\_exec\_connections* [dynamic management view](https://www.mssqltips.com/sqlservertutorial/273/dynamic-management-views/) (DMV) to see if the connections to your SQL Server are encrypted or not. If the value of *encrypt\_option* is "TRUE" then your connection is encrypted.

**SELECT session\_id, encrypt\_option  
FROM sys.dm\_exec\_connections**

# Try to connect using the Fully Qualified Domain Name

It can cause an issue if you use only the computer name in the connection string. It is better to use the Fully Qualified Domain Name (FQDN) e.g. YourSQLServer.YourCompany.internal \YourSQLServerInstance

This FQDN should match the CN that’s registered in your server certificate.

# Check the connection string of your application

Pay attention to the following properties of the connection string:

* *encrypt*
* *trustServerCertificate*

The value of the *encrypt* property should be 'true' to enable SSL encryption. If *trustServerCertificate*=true then it is possible to connect to the SQL Server using a self-signed certificate or one whose CN does not match the server name in the connection string, but this scenario is recommended only in test environments.

# SSL encryption for failover clustering in SQL Server

If you would like to use encrypted connections in a clustered environment (including AlwaysOn) then you should have a certificate issued to the fully qualified DNS name (**FQDN**) of the failover clustered instance (i.e. the AlwaysOn **listener**) and this certificate should be installed on **all** of the nodes in the failover cluster. Additionally, you will have to edit the thumbprint of the certificate in the registry because it is set to Null in clustered environment.

The following steps should be performed on all of the nodes in the cluster:

1. Navigate to the certificate in the MMC Certificates Snap-in and double click to open the certificate.
2. Copy the hex value from the **Thumbprint** property on the **Details** tab to Notepad and remove the spaces.
3. Start **Regedit** and copy the hex value to this key: **HKLM\SOFTWARE\Microsoft\Microsoft SQL Server\<YourSQLServerInstance>\MSSQLServer\SuperSocketNetLib\Certificate**
4. You will have to reboot your node, so it is recommended to failover to another node first.

**Note:** Encryption of the direct communication between AlwaysOn replicas is enabled by default, and therefore additional SSL encryption is not required.

# Rollback: Disabling the SSL Setting in SQL Server

You can configure SSL using the SQL Server Configuration Manager. First, you should run SQL Server Configuration Manager under the SQL Server service account. The only exception is if the service is running as LocalSystem, NetworkService, or LocalService, in this case you can use an administrative account.

1. Expand **SQL Server Network Configuration** and right-click on **Protocols for <YourMSSQLServer>**, then click **Properties**.
2. On the **Certificate** tab, select **None** instead of the certificate that was previously selected, then click **OK**.
3. The SQL Server service will need to be restarted again to take effect.

To undo the SSL setting in client machines:

1. Use the MMC Certificates Snap-in on the client computer to install the exported certificate file.
2. In the SQL Server Configuration Manager right-click **SQL Server Native Client Configuration**, and then click **Properties**.
3. On the **Flags** tab, select **No** in the **ForceEncryption** box, then click **OK**.

# Troubleshooting

After you successfully install the certificate, the certificate does not appear in the**Certificate** list on the**Certificate** tab.  
  
**Note:**The**Certificate**tab is in the**Protocols for <InstanceName> Properties**dialog box that is opened from SQL Server Configuration Manager.  
  
This issue occurs because you may have installed an invalid certificate. If the certificate is invalid, it will not be listed on the**Certificate**tab. To determine whether the certificate that you installed is valid, follow these steps:

* 1. Open the **Certificates** snap-in. To do this, see step 1 in the "How to Configure the MMC Snap-in" section.
  2. In the **Certificates** snap-in, expand**Personal**, and then expand Certificates.
  3. In the right pane, locate the certificate that you installed.
  4. Determine whether the certificate meets the following requirements:
     + In the right pane, the value in the**Intended Purpose**column for this certificate must be **Server Authentication.**
     + In the right pane, the value in the **Issued To**column must be the server name.
  5. Double-click the certificate, and then determine whether the certificate meets the following requirements:
     + On the **General**tab, you receive the following message:  
         You have a private key that corresponds to this certificate.
     + On the**Details**tab, the value for the**Subject**field must be server name.
     + The value for the**Enhanced Key Usage** field must be **Server Authentication (<number>)**.
     + On the **Certification Path**tab, the server name must appear under **Certification path**.

If any one of these requirements is not met, the certificate is invalid.

# Resources

* <https://support.microsoft.com/en-us/help/316898/how-to-enable-ssl-encryption-for-an-instance-of-sql-server-by-using-mi>
* [https://docs.microsoft.com/en-us/sql/database-engine/database-mirroring/transport-security-database-mirroring-always-on-availability?view=sql-server-2017](https://docs.microsoft.com/en-us/sql/database-engine/database-mirroring/transport-security-database-mirroring-always-on-availability?view=sql-server-2017#targetText=The%20type%20of%20authentication%20used,of%20the%20database%20mirroring%20endpoint.)
* <https://www.mssqltips.com/sqlservertip/3299/how-to-configure-ssl-encryption-in-sql-server/>
* <https://www.mssqltips.com/sqlservertip/3408/how-to-troubleshoot-ssl-encryption-issues-in-sql-server/>