M-FILES CORPORATION

GENERATING A TLS CERTIFICATE FOR SELF-HOSTED M-FILES

LAST UPDATED 12 MARCH 2026

VERSION 1.0

Contents

1. Introduction and Basics

To secure a network traffic endpoint, a TLS certificate is necessary. A TLS certificate lets systems verify the identity of an endpoint and establish an encrypted network connection to it. They can be considered digital identity cards given to the network endpoints.

This document gives general information about TLS certificates and how to generate them or to convert existing certificates for M-Files server to use. For more information about digital certificates, refer to information given by well-known certificate authorities (CA). For example, Verisign, IdenTrust, or DigiCert.

> **Disclaimer:** M-Files support scope does not include configuring and maintaining certificates or anything else configured outside of the M-Files Server or not directly related to the M-Files application. Each organization has its unique IT landscape, and the organizations themselves are solely responsible for configuring and maintaining it as they see fit.
>
> The purpose of this document is to provide a generalized example as a reference. It is possible that the setup and steps illustrated in this document are not directly applicable in your environment and some adjustments may be required.

1.1 Private and Public Keys in Certificates

Each TLS certificate always has two cryptographic parts: a **private** and **public** key. The **private** key is only known by the entity to whom the TLS certificate belongs. The **public** key is, as the name suggests, public and shown to any entity who wants to initiate a secure connection with the entity who owns the certificate.

The detailed process of how these keys are used when securing a connection is rooted in advanced mathematics and cryptography and is beyond the scope of this document.

1.2 Cryptographic Algorithms

There are various cryptographic algorithms used in TLS certificates. They all have their strengths and weaknesses. Elliptic Curve (EC) and Rivest-Shamir-Adleman (RSA) are the two most commonly used algorithms. They can both be used in the certificates used in this document.

When purchasing a TLS certificate from a certificate authority, both RSA and EC certificates are commonly available, though RSA has historically been the more widely used option.

1.3 Certificate Formats

Cryptographic keys can be stored in various file formats. The two formats relevant in the context of this document are Privacy Enhanced Mail (PEM) and Personal Exchange Format (PFX, sometimes also referred as PKCS#12). Both containers can include just the public key, both the private and public keys, or even the entire certificate chain with any intermediate certificates. In practical use, PEM is more often used to store individual certificate components, while PFX is normally used to package everything together.

Certificates in the PEM format are plain text files, which can be opened and viewed with any text editor. PEM-formatted private keys must be kept secure as they are directly readable by anyone with access to them. Due to its plain-text nature, transferring PEM files between systems requires the use of secure transfer methods to protect the private key from being exposed.

PFX is the certificate format used by Microsoft Windows and normally contains both the private and public key. PFX format supports password protection, which makes it suitable for importing and exporting certificates between different systems and applications. Access to the PFX file alone is not sufficient without the password.

1.4 Certificate Trust Scope

Certificates can be split into two categories based on their trust scope: publicly trusted certificates and self-signed certificates.

Publicly trusted certificates can be purchased from a well-known and trusted certificate authority (CA). These certificates are signed and validated by the CA, which chains up to a trusted root certificate. This makes most devices and applications automatically trust these certificates. It is recommended to use trusted certificates in endpoints where all entities that initiate the connection to the secured endpoint are not known beforehand. In these scenarios, it is not possible to provide the certificate beforehand for the connecting entities and establish trust in it. Thus, it is important that they can trust the certificate automatically without having seen it before. For example, endpoints that directly face the public internet typically use publicly trusted certificates.

Self-signed certificates can be generated by anyone and do not cost anything. There is no innate trust in self-signed certificates, but trust can be established explicitly by the local organization. This can be done by providing the certificate to the connecting entity beforehand through a trusted medium and instructing them to trust it.

Technically, trust can be established in multiple ways. On Windows machines, many applications automatically trust any certificate that is imported into the Microsoft Certificate Store, while some applications may trust certificates found at a certain folder path.

Self-signed certificates can be used to secure connections where all the entities that initiate the communication to the secured endpoint are known beforehand. Endpoints that only face the organization's internal network can normally use self-signed certificates, although trusted certificates can be used as well.

When your organization implements such trust for self-signed certificates, they begin acting as trusted certificates for the purpose of connection encryption and authentication. Therefore, it is important to understand and distinguish the terms and the current situation you have.

2. Creating a Self-Signed Certificate

There are various tools that can be used to create self-signed certificates. In this document, the widely used free tool OpenSSL (https://www.openssl.org) is used. One of the easiest ways to install it to a Windows machine is to install Cygwin terminal with the *Base* set of packages (it includes OpenSSL).

You must execute OpenSSL commands in your prompt to complete the rest of these instructions. A certificate created with the instructions below is in the correct format to be installed for both M-Files server and Traefik reverse proxy running on DMZ machine, if that is used in your M-Files environment.

1. Download and install OpenSSL.
   - For example, Cygwin for Windows includes OpenSSL.
   - Any computer can be used. It is not necessary for the tool to be on any of the servers.
   - You can use the default values and selections for all installation settings.
   - Restart your machine after the installation.
2. Generate a private and public key with the OpenSSL command below.
   Please read the considerations first, as you may need to adjust the command for your situation:
   - Replace 365 with the preferred number of days. It specifies the time of validity for the certificate.
   - Replace the **phrases written in accent color** to use the domain and hostnames of the server that the certificate is for.
   - The certificate must include all names (public and internal), which any entity uses when connecting to the server, for example:
     - The internal name of the M-Files server (mfilesserver.local) can be used as *Subject Distinguished Name*.
     - All the DNS names what are used to connect to the M-Files server, via M-Files Desktop or other clients, including proxy servers, (mfilesserver.company.com) should be set to *Subject Alternative Name.*
   - In this example, the certificate is generated so that it is valid for both the public and internal name of the application.
     - In case you use wildcard name, you may not need to list many alternative DNS names, so you can use the same string in both CN and a single SAN with the same wildcard name.
   - In case the server has multiple vaults which use vault-specific names, you should add separate subjectAltName entry for each vault or generate a wildcard certificate which matches all vault names.
     - You can find a more detailed example of certificate generation through a configuration file here.
   - The example command to generate self-signed certificate is:

```
openssl req -x509 -newkey rsa:4096 -keyout mfilesserver.key -out mfilesserver.crt -days 365 -nodes \
  -subj "/C=FI/ST=/L=Tampere/O=M-Files test/OU=DEV/CN=mfilesserver.local" \
  -addext "subjectAltName=DNS:mfilesserver.local,DNS:mfilesserver.company.com"
```

As a result, two files are created, mfilesserver.key and mfilesserver.crt. These files are in a format that both M-Files server and Traefik can use.

> **Important:** Remember to implement the necessary process and reminders to **update the certificate well before it expires**. If the certificate expires, connections to that endpoint will not operate correctly, and this can make your M-Files system unusable.

3. Converting an Existing Certificate to a Suitable Format for M-Files

This chapter explains the requirements M-Files has for the TLS certificate, then offers some examples of how to perform the most common conversion and to ensure that the intermediate certificates are included, if needed.

3.1 Requirements for the Certificate

To be used with M-Files and Traefik, the certificate must fulfill these requirements:

- All the DNS names what are used to connect to the M-Files server, via M-Files Desktop or other clients, including proxy servers, should be set to the *Subject Alternative Name* (**SAN**) attribute of the certificate.
    - In case wildcard certificate is used, you may not need to list many alternative DNS names. Instead, you can use the same string in both CN and a single SAN with the same wildcard name that will match to the DNS names what are used to connect to M-Files server.
- The certificate and private key files must be in PEM format (Base64 encoding).
    - Both EC and RSA algorithms are supported.
        - EC private keys must be in the PKCS#8 PEM format.
          (-----BEGIN PRIVATE KEY----)
        - RSA certificates are also supported in the PKCS#1 format.
          (-----BEGIN RSA PRIVATE KEY----)
- Certificates purchased from certificate authorities may come in PFX format. For M-Files or Traefik to use the certificate, it must be converted to PEM format (see section 3.2), which usually has .crt or .pem file extension.
- The certificate must be trusted by a trusted root certificate on the client computers.
    - When you purchase the certificate from a well-known CA, this is automatically fulfilled and it is not necessary to install the certificate to the connecting client devices.
    - **If the certificate is not trusted directly by the root certificate, the certificate file must also contain all the certificates of the trust chain**: the server certificate, all intermediate certificates, and the root certificate.

- In the PEM format, the certificate contents can be written into the file one after another.
- For example:

```
-----BEGIN CERTIFICATE-----
<Your primary certificate content>
-----END CERTIFICATE-----
-----BEGIN CERTIFICATE-----
<Intermediate certificate content>
-----END CERTIFICATE-----
-----BEGIN CERTIFICATE-----
<Another intermediate certificate as necessary>
-----END CERTIFICATE-----
-----BEGIN CERTIFICATE-----
<Root certificate content>
-----END CERTIFICATE-----
```

3.2 Converting Certificate from PFX to PEM Format

If your certificate files are in the PFX format, these instructions tell you how to convert them into PEM format (CRT files). If you already have the certificate in PEM format (.CRT file) you can skip these steps.

1. Place your PFX certificate in a suitable location.

   In this example, the file is certificate.pfx and the password for the certificate is 12345
2. Execute this OpenSSL command to extract the certificate from the pfx file including all intermediate certificates:

   ```
   openssl pkcs12 -in certificate.pfx -nokeys -out certificate.crt -passin pass:12345
   ```
3. Execute this OpenSSL command to extract the key from the .pfx file:

   ```
   openssl pkcs12 -in certificate.pfx -nocerts -nodes -out certificate.key -passin pass:12345
   ```

As a result, you have one certificate file certificate.crt and one key certificate.key which is the unencrypted private key. If your PFX certificate contained the full chain of intermediate and root certificates, the resulting certificate.crt should contain the intermediate and root certificates also. However, we recommend that you still double check the output to ensure all certificates are present as necessary.

> **Note:** depending on where you run the OpenSSL command, you may need to put full path in single quotes to both -in and -out parameters for the certificate file. Also, if password contains special characters, use single quotes around it.

3.3 Exporting Intermediate Certificates

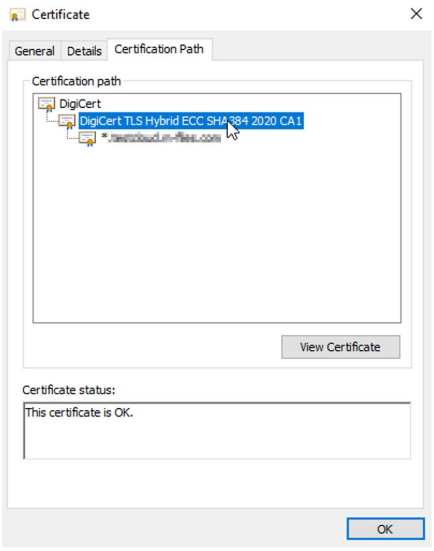
This chapter instructs on how to check if the PEM certificate file contains all the needed certificates and how to export the missing ones if necessary.

3.3.1 Verifying the Certificate Chain in the PEM File

The PEM (.crt) certificate file installed on M-Files server must have all the intermediate certificates from the whole trust chain. If you exported the certificate with the above instructions from .pfx file, the resulting certificate.crt should already include all intermediate certificates that were present in the .pfx file.

If you received the certificate already in PEM format, use the following steps to verify that it contains all necessary intermediate certificates:

1. Double-click the .crt file to open it.
2. Verify that the certificate includes all intermediate certificates, as shown in the screenshot. Intermediate certificates are those between the root certificate (first level) and your certificate (last level).



3. If intermediate certificates are present, open certificate.crt or the received PEM certificate in a text editor.
4. Verify that the file contains multiple BEGIN CERTIFICATE — END CERTIFICATE blocks and that all intermediate certificates are represented among them. Including the root certificate is not required but is recommended.

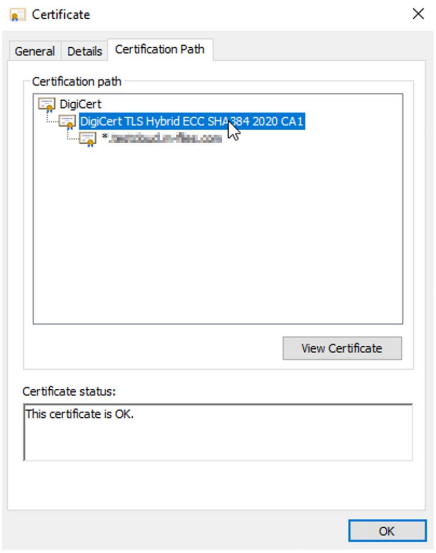
If the .crt file already contains all intermediate certificates, the manual steps below are not necessary.

3.3.2 Exporting Intermediate Certificates Manually

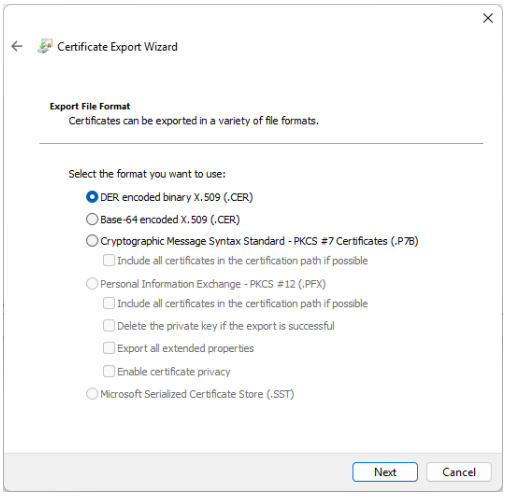
If the PEM certificate file does not contain all intermediate certificates, you need to export them from their source and append them to the certificate file you will provide to M-Files Server.

1. Open the certificate file: double-click a .crt file, or right-click a .pfx file and select Open to open it in Windows Certificate Manager (certmgr).
    - If you opened the .pfx file in Certificate Manager, expand the certificates in the window and double-click the certificate to open it.

2. On the **Certification Path** tab, select the intermediate certificate you want to export.



3. Select **View Certificate** and in the new dialog, select **Details** > **Copy to File**.
4. In the Copy to File dialog, select **DER format (.CER)** and save the file as intermediate.cer. Use the default settings in further dialogs.



5. Run this command in Open SSL to convert the der certificate into correct format:

```
openssl x509 -inform der -in intermediate.cer -out intermediate.crt
```

6. Open the PEM certificate file certificate.crt and intermediate.crt in a text editor.
7. Copy the contents of intermediate.crt.
8. Paste the content from your clipboard to the end of the certificate.crt file.
   The end result in certificate.crt looks similar to the example in section 3.1.1. Note how the order in the text file is reversed compared to the order in which you see the hierarchy in the Windows Certificate Manager (certmgr) — the root certificate is now at the bottom, the leaf certificate is at the top.
9. If the certificate chain contains additional intermediate certificates, repeat steps 2–8 for each one. The root certificate does not need to be included, but it is recommended — particularly if it is a self-signed certificate generated and distributed to client computers by your organization.

Tip: If you have issues with a certificate, you can use third-party services to analyze it. For example, https://decoder.link/result. This can help you detect problems, for example, with the certificate's chain of trust.

4. Testing PEM Certificate

To make sure that the certificate chain is correct, you can use OpenSSL to test it. Replace parts written in **accent color** to match your environment.

```
openssl s_client -connect yourserver:443 -CAfile "path/to/filename.pem"
```

5. Change History

The table describes the changes by document version.

VERSION	DATE	ESSENTIAL CHANGES
1.0	2026/03/12	The first published version.

6. Reference Documents

- Using a Reverse Proxy on DMZ with Self-Hosted M-Files