

# Managing certificates

- [Introduction](#)
- [Provided certificates and keystores](#)
- [Using certificates using the provided CA or your own CA](#)
  - [Step 1: set the CA to use when signing your certificates](#)
    - [Use the provided CA](#)
    - [Let the tool generate a new CA](#)
    - [Use your own CA](#)
  - [Step 2: generate the certificates and keystores](#)
  - [Step 3: copy the generated files to the user-content folder](#)
  - [Step 4: continue with your installation](#)
- [Using certificates signed by a trusted CA](#)
  - [Step 1: create a CSR \(Certificate Signing Request\)](#)
  - [Step 2: create the java keystore](#)
  - [Step 3: continue with your installation](#)
- [Using certificates provided by your organization](#)
  - [Step 1: verify files format](#)
  - [Step 2: backup provided files](#)
  - [Step 3: generate java keystore and truststore](#)
  - [Step 4: copy the generated files to the user-content folder](#)
  - [Step 5: continue with your installation](#)
- [Adding the provided or a custom CA to Kiuwan On-Premises' clients](#)

## Introduction

Kiuwan On-Premises fosters secure connections by providing a default installation environment where most communications are done under a secure protocol.

By default, Kiuwan On-Premises service connections use:

Communication between		Protocol	Secure connection
Any client (browser, KLA, K4D, custom REST API client, etc.)	Kiuwan apache load balancer	HTTPS	Yes
Kiuwan apache load balancer	Kiuwan (frontal)	HTTPS	Yes
Kiuwan (frontal, analyzer, scheduler, updater)	MySQL database	MySQL protocol (SSL can be optionally enabled)	Optional
Kiuwan (frontal, analyzer, scheduler, updater)	Redis cluster node	RESP (Redis Serialization Protocol) + SSL	Optional (only supported when using AWS elasticsearch)
Redis cluster node	Redis cluster node	RESP (Redis Serialization Protocol)	Optional (only supported when using AWS elasticsearch)

Every time a client connects to a server using a secure protocol, it needs to make sure that the contacted server is who it claims to be. This is usually done by the server returning a certificate (signed by a Certification Authority, CA) that the client can check for authenticity.

As the client needs a way to identify if the server's certificate is trustable, all secure transmission enabled clients have or rely on a dictionary of trustable CAs.

In order to provide a default installation configuration that enables secure protocols on most communications channels, Kiuwan On-Premises comes with a set of certificates and keystores for the default configured domain (kiuwan.onpremise.local).



Note that the previous statement means that, if you rely on the default installation configuration, all your certificates will be the same as other Kiuwan customers certificates. We encourage you to create your own CA for signing your own domain certificates or sending a CSR to a trusted CA. See the following sections for more information on this topic.

## Provided certificates and keystores

Kiuwan On-Premises installation tool (kiuwan-cluster) provides a number of files to allow secure communications between containers. These files are located in kiuwan-cluster distributions under the ssl folder.

The following table shows the provided certificate files:

Location	File	Public key algorithm	Key format	File encoding	Signature algorithm	Content	Purpose	Expiration date
ssl/ca	cacert.pem	RSA 4096 bits	X.509	PEM	SHA256 with RSA	The CA certificate that signed Kiuwan On-Premises domain certificate	Allows Kiuwan servers to provide the CA that signed their certificates	2029/10/13
ssl/kiuwan.onpremise.local	domain cert.pem	RSA 4096 bits	X.509	PEM	SHA256 with RSA	The Kiuwan On-Premises domain certificate	Allows Kiuwan servers to identify themselves	2029/10/13

The following table shows the provided private key files:

Location	File	Private key algorithm	Key format	File encoding	Content	Purpose
ssl/ca	cakey.pem	RSA 4096 bits	PKCS#1	PEM	The provided CA private key (password protected)	Allows signing certificates with the provided CA
ssl/kiuwan.onpremise.local	domain key.pem	RSA 4096 bits	PKCS#1	PEM	The Kiuwan On-Premises domain private key	Allows encrypting traffic for the provided domain

The following table shows the provided Java keystore files:

Location	File	Content	Purpose
ssl/kiuwan.onpremise.local	domainkeystore.jks	This keystore contains cacert.pem, domaincert.pem and domainkey.pem files. Its password is the one provided in the default installation configuration (see java.keystore.password property).	Allows Kiuwan instances to identify themselves and encrypt traffic to enable secure connections.
ssl/kiuwan.onpremise.local	truststore.jks	This keystore contains all the CA certificates included in the OpenJDK default truststore (see next row in this table) plus the provided CA certificate. Its password is the one provided in the default installation configuration (see java.truststore.password property).	Allows Kiuwan instances to communicate to external servers that offer certificates signed by trusted CAs (needed both for AWS based installations and Kiuwan central servers communications).
ssl/truststore	truststore.jks	This keystore contains the OpenJDK 13 trusted CAs as of 2019/10/16. Its password is the one provided by OpenJDK for its cacerts file.	Allows generating a custom truststore that includes most needed trusted CAs certificates plus the one provided by the installation tool.

## Using certificates using the provided CA or your own CA

Kiuwan On-Premises installer (kiuwan-cluster) contains a handy tool to create certificates both with the provided CA or your own CA.

The tool is a bash script located here:

- [INSTALLER\_DIR]/ssl/kiuwan-certtool.sh


Remember that, as stated in [Installation guide - Installation requirements](#), you will need the specified versions of a JRE and OpenSSL in order to be able to generate certificates using the provided tool.

When generating custom certificates, it is recommended that you change the default properties in the configuration file located here:

- [INSTALLER\_DIR]/ssl/config/certs.properties

This is what the customizable properties of the previous file (default passwords are omitted) mean:

Property	Default value	Meaning
----------	---------------	---------

java. keystore. password		The password to set to the generated Java keystore.
java. truststore. password		The password to set to the generated Java truststore.
ssl.ca. password		The password to set to the generated CA (only applies when generating a new custom CA). The set password will be used when signing certificates as well.
ssl.country	US	Country, state, locality, organization or organization unit to set both to the subject of the CA certificate (in case of you are generating a new custom CA) and to the subject of the specified domain signing request.
ssl.state	mystate	
ssl.locality	mylocality	
ssl. organization	mycompany	
ssl. organization. unit	myorganization. unit	
ssl. company. domain	mycompany. com	Company domain to set to the subject's Common Name (CN) of the CA certificate (in case of you are generating a new custom CA).
ssl.subject. alt.names	DNS:kiuwan. onpremise. local[:443,: 3306,:6379]  DNS: wildflykiuwan- f[1-2][:8143,: 8443]  DNS: wildflykiuwan Container-f[1- 2][:8143,: 8443]  DNS: mysqlkiuwan[: 3306]  DNS: mysqlkiuwan Container[: 3306]  DNS: redis_0000[1- 6][:6379]	Subject Alternative Names (SANs) that will be set to the specified domain certificate. These are needed in order to be able to share the same certificate between different services of the Kiuwan On-Premises infrastructure.  <div>  <p>If you have modified the default ports, you must set the new ports to these properties and create new certificates. See <a href="#">Modifying exposed ports</a> section for more details.</p> </div>

## Step 1: set the CA to use when signing your certificates

The provided tool will use the CA files located here:

- [INSTALLER\_DIR]/ssl/ca/cacert.pem
- [INSTALLER\_DIR]/ssl/ca/akey.pem

You can either:

- Use the provided CA.
- Let the tool generate a new CA.
- Use your own CA.

### Use the provided CA

Just continue to [Step 2: generate the certificates and keystores](#).

### Let the tool generate a new CA

Just backup the provided CA files and a new CA will be automatically generated:

```
cd [INSTALLER_DIR]/ssl/ca
mv cacert.pem cacert.pem.bak
mv cakey.pem cakey.pem.bak
```

## Use your own CA

Just replace the provided files with your own CA's ([INSTALLER\_DIR]/ssl/ca/cacert.pem and [INSTALLER\_DIR]/ssl/ca/cakey.pem).

We recommend backing up the provided CA files just in case you want to get back to the provided defaults (see [Use your own CA](#)).

## Step 2: generate the certificates and keystores

To generate all the needed files using the provided CA and the default configuration, just run the following commands:

```
cd [INSTALLER_DIR]/ssl
./kiuwan-certtool.sh [DOMAIN_NAME]
```

This will create the following files under the ssl/[DOMAIN\_NAME] folder:

- domaincert.pem
- domainkey.pem
- domainkeystore.jks
- truststore.jks

## Step 3: copy the generated files to the user-content folder

You can run the following commands to automatically copy the needed files to the user-content folder, where the installer tool deploy-user-content.sh will read from when deploying the user content to the persistent volumes locations:

```
cd [INSTALLER_DIR]/ssl
./kiuwan-cercopy.sh [DOMAIN_NAME]
```

## Step 4: continue with your installation

The following step is to run the deploy-user-content.sh script to let the installer deploy your certificates to the persistent volumes. Note that once this is done and depending on your installation needs, the following steps may change. Please refer to the [Kiuwan On-Premises Distributed Installation Guide](#) page for more information.

## Using certificates signed by a trusted CA

Note that the Kiuwan On-Premises installation tool does not automate this process as it may be different between organizations based on their security policies.

The following table shows the files that Kiuwan On-Premises needs:

File	Where does it come from?	How can I get it?
domainkey.pem	You have to generate this file	Use a SSL tool to generate it
cacert.pem	Your CA will provide this file	Your CA will send this file to you after a CSR (Certificate Signing Request) has been fulfilled
domaincert.pem	Your CA will provide this file	Your CA will send this file to you after a CSR (Certificate Signing Request) has been fulfilled

domainkeystore.jks	You have to generate this file	Use your JRE's keytool program to generate it
truststore.jks	Provided by the installation tool	It is stored in [INSTALLER_DIR]/ssl/truststore/truststore.jks

Here are the usual steps to follow when requesting a CA to create a certificate for your domain. Most of them can be carried out using OpenSSL (but there are other alternative tools available).

## Step 1: create a CSR (Certificate Signing Request)

Follow these steps:

1. Create a private key file (**domainkey.pem**).
2. Create a certificate signing request (CSR) file using the keyfile from the previous step.
3. Send the CSR file to your trusted CA for signing.
4. Store the files sent by your trusted CA:
  - a. The CA certificate (**cacert.pem**).
  - b. Your domain certificate (**domaincert.pem**).

## Step 2: create the java keystore

At this point you will need three files. Your private key and those sent by your CA:

1. Create a pkcs12 file using the CA certificate, your private key file and your domain certificate.
2. Create a Java keystore (**domainkeystore.jks**) that:
  - a. Contains both the CA certificate and your domain's certificate.
  - b. Its format is pkcs12.
  - c. Has an alias "domainp12" for your domain's certificate.

Once you have all the needed files (remember that you can use the provided **truststore.jks** file), copy them to:

- [INSTALLER\_DIR]/user-content/certs

## Step 3: continue with your installation

The next step is to run the deploy-user-content.sh script to let the installer deploy your certificates to the persistent volumes. Note that once this is done, depending on your installation needs, the following steps may change. Please refer to the [Kiuwan On-Premises Distributed Installation Guide](#) page for more information.

## Using certificates provided by your organization

If your organization manages their own certificates, you should send a requirement for the domain where your Kiuwan On-Premises installation will be accessible from. You should indicate that this domain should match the CN set in the generated certificate. You should receive these files from the department responsible for generating the certificate files:

- The public certificate of your organization's CA (cacert.pem).
- The public certificate for Kiuwan On-Premises domain (domaincert.pem).
- The private key of your Kiuwan On-Premises domain certificate (domainkey.pem).

Note that the previous files may have different names depending on your organization's naming policies.



In case your company generates generic certificates whose CN do not match the required domain, you should update your front-end server configuration to avoid validating the CN of the certificate against the accessed hostname. For the provided Apache load balancer, just set the directive "SSLProxyCheckPeerCN" to "off" in the httpd.conf file located under [VOLUMES\_DIR]/shared-conf/ApacheLoadBalancer/conf.

## Step 1: verify files format

Depending on how your organization creates their certificates, your files may have a format not supported by the installation tool.

All provided certificates **MUST** be in PEM format. Please refer to [Provided certificates and keystores](#) section for more details.

In case you have received the certificate files in a different format, you should convert the files to PEM format.

Once the files have been converted and renamed, you should end up with three files:

- cacert.pem
- domaincert.pem
- domainkey.pem

## Step 2: backup provided files

In order to avoid confusion, rename folder where the provided files reside to keep them separated from the ones you have already prepared:

```
cd [INSTALL_DIR]/ssl
mv ca kiwanca
```

## Step 3: generate java keystore and truststore

Copy your cacert.pem file to the ca folder:

```
cd [INSTALL_DIR]/ssl
mkdir ca
cp cacert.pem [INSTALL_DIR]/ssl/ca
```

Then put the domain files into a folder named as your domain:

```
cd [INSTALL_DIR]/ssl
mkdir [DOMAIN_NAME]
cp domaincert.pem [INSTALL_DIR]/ssl/[DOMAIN_NAME]
cp domainkey.pem [INSTALL_DIR]/ssl/[DOMAIN_NAME]
```

Invoke the kiuwan-certool.sh script to generate the needed java stores:

```
cd [INSTALL_DIR]/ssl
./kiwan-certool.sh [DOMAIN_NAME]
```

This will create the missing jks files into the [DOMAIN\_NAME] folder.

## Step 4: copy the generated files to the user-content folder

You can run the following commands to automatically copy the needed files to the user-content folder, where the installer tool deploy-user-content.sh will read from when deploying the user content to the persistent volumes locations:

```
cd [INSTALLER_DIR]/ssl
./kiwan-cercopy.sh [DOMAIN_NAME]
```

## Step 5: continue with your installation

The following step is to run the deploy-user-content.sh script to let the installer deploy your certificates to the persistent volumes. Note that once this is done and depending on your installation needs, the following steps may change. Please refer to the [Kiuwan On-Premises Distributed Installation Guide](#) page for more information.

## Adding the provided or a custom CA to Kiuwan On-Premises' clients

The Kiuwan On-Premises installer provides default certificates for the default host name, signed by a supplied CA (Certificate Authority).

The CA public certificate is provided in this file:

- [INSTALLER\_DIR]/ssl/ca/cacert.pem

If you choose to sign your domain's certificate with the provided CA, a new CA created using kiuwan-certtool.sh or your own CA, internet browsers and other clients accessing your Kiuwan On-Premises installation will not recognize it as a trusted CA by default. You will get error messages like this one:

```
Your connection is not private
Attackers might be trying to steal your information from kiuwan.onpremise.
local (for example, passwords, messages, or credit cards).
NET::ERR_CERT_AUTHORITY_INVALID
```

This is the expected behavior as the CA store that your browser or client uses will not contain your own CA or the one supplied along with kiuwan-cluster.

In order to make your browser trust the supplied certificates, you will need to add this CA to your browser, and Java clients that access your Kiuwan On-Premises installation:

- Firefox, Chrome, Edge: import cacert.pem by using the tools provided by the browsers.
- Java clients (Kiuwan for developers Eclipse, Kiuwan for developers JetBrains, Jenkins plugin, KLA, etc): add the provided cacert.pem to the JRE keystore used by the client. Please refer to the official documentation of your JRE distribution about the Java keytool program.
- Windows clients (Kiuwan for developers VisualStudio): import cacert.pem by using the tools provided by Windows (certmgr.msc).
- Multiplatform clients (Kiuwan for developers VisualStudioCode): import cacert.pem by using the tools provided by your OS.