



ENTRUST

Signing Automation Client 1.3

User Guide

Doc release: 1.0
Release date: February 5, 2024

© 2024, Entrust. All rights reserved

Entrust and the hexagon design are trademarks, registered trademarks and/or service marks of Entrust Corporation in Canada and the United States and in other countries. All Entrust product names and logos are trademarks, registered trademarks and/or service marks of Entrust Corporation. All other company and product names and logos are trademarks, registered trademarks and/or service marks of their respective owners in certain countries.

This information is subject to change as Entrust reserves the right to, without notice, make changes to its products as progress in engineering or manufacturing methods or circumstances may warrant. The material provided in this document is for information purposes only. It is not intended to be advice. You should not act or abstain from acting based upon such information without first consulting a professional. ENTRUST DOES NOT WARRANT THE QUALITY, ACCURACY OR COMPLETENESS OF THE INFORMATION CONTAINED IN THIS ARTICLE. SUCH INFORMATION IS PROVIDED "AS IS" WITHOUT ANY REPRESENTATIONS AND/OR WARRANTIES OF ANY KIND, WHETHER EXPRESS, IMPLIED, STATUTORY, BY USAGE OF TRADE, OR OTHERWISE, AND ENTRUST SPECIFICALLY DISCLAIMS ANY AND ALL REPRESENTATIONS, AND/OR WARRANTIES OF MERCHANTABILITY, SATISFACTORY QUALITY, NON-INFRINGEMENT, OR FITNESS FOR A SPECIFIC PURPOSE.

Contents

1 Service licensing	5
2 Service overview	6
Customer premises	6
Entrust datacenters	6
Entrust Signing Automation user	7
Service rate.....	7
Signing certificate	7
Signing REST API	8
Virtual Token	8
3 Configuring signing automation in Entrust Certificate Services	10
Creating a Virtual Token	10
Creating a Signing Client	12
Downloading the Signing Automation License	13
Creating a document signing certificate.....	14
Creating a code signing certificate.....	17
4 Downloading and installing the Signing Automation Client	20
Signing Automation Client requirements.....	20
Downloading the Signing Automation Client.....	20
Installing the Signing Automation Client.....	22
5 Signing with third-party signing applications	24
Requirements for signing with third-party signing applications.....	24
Signing PDF documents with iText	26
Signing Microsoft Authenticode files with Jsign	28
6 Signing with REST clients	33
Getting API credentials for REST clients	33
Signing data with curl commands	35
Signing data with Postman	37
7 Debugging the Signing Automation Client	41

PKCS11_LOGGER_LIBRARY_PATH	41
PKCS11_LOGGER_LOG_FILE_PATH	41
PKCS11_LOGGER_FLAGS.....	41
8 Uninstalling the Signing Automation Client	43
9 Command-line reference.....	44
signingclient completion	44
signingclient config list	45
signingclient config set	46
signingclient create key	47
signingclient credentials	49
signingclient delete certificate	50
signingclient delete key	51
signingclient help	52
signingclient import certificate	52
signingclient list certificates.....	54
signingclient list keys.....	54
signingclient process license.....	54
signingclient version	56

1 Service licensing

This section defines the licensing terms and permitted uses for the Signing Automation Service, including the Signing Client software.


In this document, when we refer to “you”, we mean the customer who has purchased the Signing Automation Service or one of that customer’s Users. In general, the Signing Automation Service is for the customer’s internal use only. However, the customer is permitted to appoint outside organizations and/or their software applications as Users and distribute the Signing Client software to such Users, solely to enable those Users to use the Signing Automation Service on the customer’s behalf (and not for the outside User’s benefit). The customer is responsible for overseeing and controlling how the Users use the Signing Automation Service.

You may deploy the Entrust Signing Client software on your company’s infrastructure and/or commercial cloud accounts. You are strongly encouraged to keep your deployments up to date with our latest product release.

You may use the Signing Automation Service only to apply for and in connection with an Entrust Signing Automation certificate that identifies you (Entrust’s customer) as the Subject by your organization name (e.g., “ABC Ltd”) or one of your corporate affiliates if that affiliate authorizes you to sign documents on its behalf. You are expressly prohibited from using the Signing Automation Service together with signing certificates that identify as the Subject any person other than yourself or an affiliate that has authorized you to sign documents on its behalf. You are expressly prohibited from using the Signing Automation Service in conjunction with any certificate not issued by Entrust.

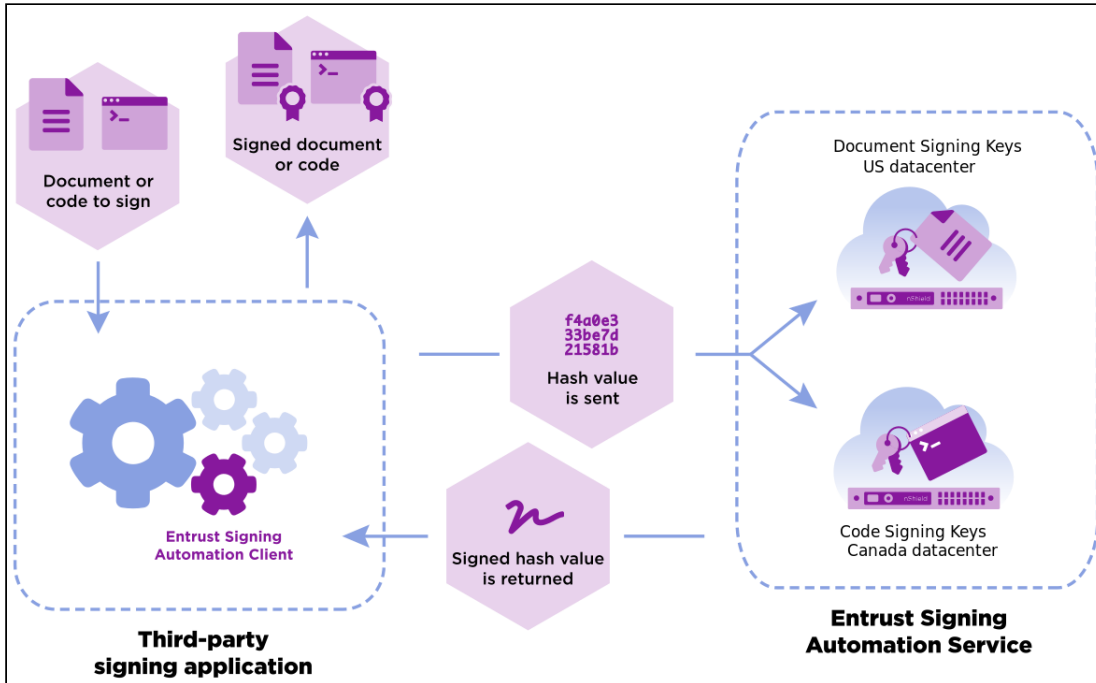
For each subscription to the Signing Automation Service, we will provide a license key that determines the number of digital signatures that you may generate using the service, the number of Signing Automation Certificates with which you may use the service, and/or for otherwise enabling or controlling certain functions within the service. You may not copy or alter your license key. You may not circumvent or attempt to circumvent the license key mechanism. You may only use a license key provided by Entrust and only in conjunction with the Signing Automation Service and Signing Client software for which it was delivered. Unless otherwise specified in your Order(s), each Entrust Signing Automation Service license key is capped to one Signing Automation certificate based on RSA 2048-bit key pairs for 10,000 signatures to be consumed within one year. Additional Entrust Signing Automation certificates or signatures can be purchased separately.

Each issuing or reissuing of a signing certificate consumes one of the 10,000 licensed signatures.

 The Entrust Signing Client includes software developed by the OpenSSL Software Foundation, and provides an interface compatible with the PKCS #11 Cryptographic Token Interface (Cryptoki) by RSA Security Inc.

2 Service overview

The Entrust Signing Automation Service is a cloud-based hash signing service, with private keys securely generated and protected by the service. Using this service, you can sign hashes generated on documents or code files.



See below for a description of the main concepts.

- [Customer premises](#)
- [Entrust datacenters](#)
- [Entrust Signing Automation user](#)
- [Service rate](#)
- [Signing certificate](#)
- [Signing REST API](#)
- [Virtual Token](#)

i The Entrust Signing Automation Service includes timestamping and OCSP validation services.

Customer premises

The customer premises run:

- The Entrust Signing Automation Client.
- The signing application: A REST client or a third-party product with signing capabilities such as the iText library, Oracle JDK, OpenJDK, or Entrust Java Toolkits.

Entrust datacenters

The cloud-based Entrust data centers host the following services.

- [Multi-tenant signing key management](#)

- [nShield HSMs](#)

i The Entrust data centers never access the documents to be signed because generating a digital signature only requires the document hash.

Multi-tenant signing key management

The multi-tenant signing key management selects the nShield HSM that will:

- Generate and wrap the keys when processing key generation requests.
- Unwrap the signing key of the signing key database and sign a document hash when processing a signature request.

nShield HSMs

The nShield HSMs are FIPS 140-2 L3 compliant devices for:

- Generating and wrapping the keys stored in the signing key database.
- Unwrapping the signing keys.
- Signing document hashes with the unwrapped keys.

Entrust Signing Automation user

In the Entrust Certificate Service portal, the users of the Signing Automation Service are referred to as "Signing Clients". All service users have the same functionality when consuming the Signing Automation Service from computers configured for that purpose. However, it will be usual to separate the following roles.

Role	Action
Administrator	Create signing keys and import certificates in the token, as explained in Configuring signing automation in Entrust Certificate Services .
Software application	Select a token key and sign documents.

Service rate

Entrust Signing Automation Service can deliver a minimum of 10 signatures per second and customer.

Signing certificate

Depending on the data you want to sign, you must either purchase:

- A Document Signing Certificate
- A Code Signing Certificate

See [Configuring signing automation in Entrust Certificate Services](#) for how to obtain these certificates.

Signing REST API

As explained [Signing with REST clients](#), you can use the Signing Automation Client to consume the Entrust Signing Automation Service. See below for a description of the main related concepts.

- [Authentication token](#)
- [Identity Provider service](#)
- [Raw Signature service](#)
- [User partition](#)

Authentication token

With the credentials described in [Getting API credentials for REST clients](#), the [Identity Provider service](#) provides authentication tokens for registered users. Each authentication token:

- Is the Base64 encoding of a JSON Web Token (JWT).
- Allows a user to consume the [Raw Signature service](#) for up to two minutes.

Identity Provider service

The Identity Provider service of the Entrust Signing Automation Service API manages users and credentials of the [Raw Signature service](#). Specifically, this service grants users [Authentication tokens](#) to consume the [Raw Signature service](#).

 See <https://api.managed.entrust.com/sas/idp-api> for the Swagger reference of this service.

Raw Signature service

The Raw Signature service of the Signing Automation Service API:

- Manages the user keys and certificates, grouped in [Virtual Tokens](#).
- Generates signatures on raw data – typically, the digest of a document.

 See <https://api.managed.entrust.com/sas/rawsigner-api> for the Swagger reference of this service.

User partition

In the Signing Automation Service API, the user *partition* refers to the user organization.

Virtual Token

The Signing Automation Service uses “virtual tokens” as per the “physical” tokens described in the PKCS #11 standard. When your signing application connects to the server, the Signing Automation Client provides a virtual token with the following standard operations.

PKCS #11 Mechanism	Supported operations
CKM_ECDSA	Sign and verify with NIST P256, NIST P384, or NIST P521 curves.

PKCS #11 Mechanism	Supported operations
CKM_ECDSA_KEY_PAIR_GEN	Generate ECDSA key pairs with NIST P256, NIST P384, or NIST P521 curves.
CKM_RSA_PKCS	Sign and verify with PKCS1.5 padding (and software-based hashing) with 2048, 3072, or 4096 key sizes.
CKM_RSA_PKCS_KEY_PAIR_GEN	Generate RSA key pairs with 2048, 3072, or 4096 key sizes.
CKM_SHA256_RSA_PKCS	Sign and verify with PKCS1.5 and SHA 256 hashing with 2048, 3072, or 4096 key sizes.
CKM_SHA384_RSA_PKCS	Sign and verify with PKCS1.5 and SHA 384 hashing with 2048, 3072, or 4096 key sizes.
CKM_SHA512_RSA_PKCS	Sign and verify with PKCS1.5 and SHA 512 hashing with 2048, 3072, or 4096 key sizes.

3 Configuring signing automation in Entrust Certificate Services

To run the signing automation client, you must perform the following operations in the Entrust Certificate Services portal.

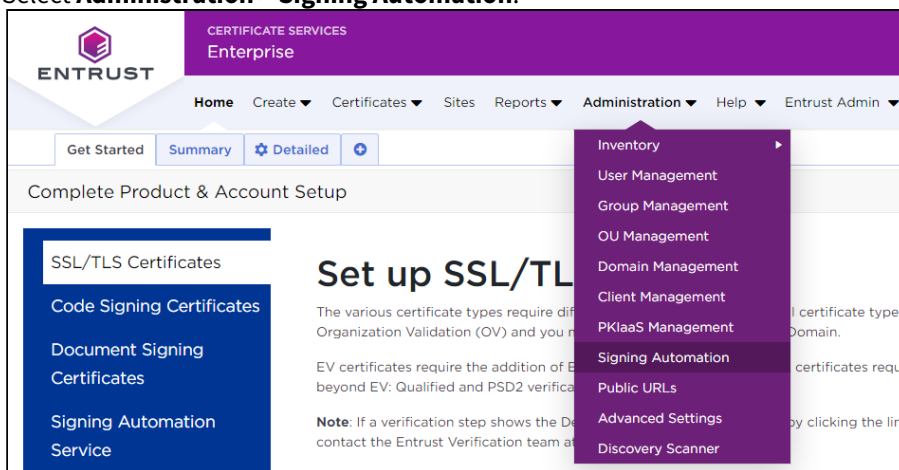
- [Creating a Virtual Token](#)
- [Creating a Signing Client](#)
- [Downloading the Signing Automation License](#)
- [Creating a document signing certificate](#)
- [Creating a code signing certificate](#)

Creating a Virtual Token

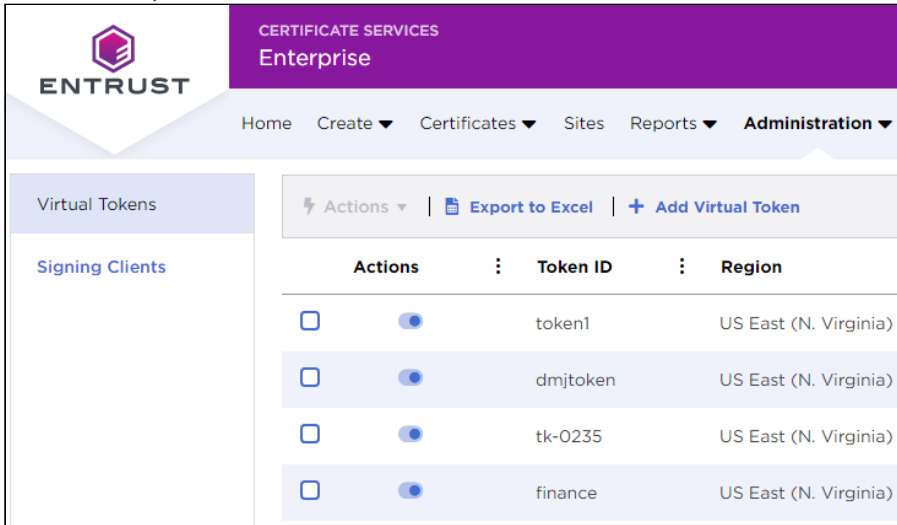
In the Entrust Certificate Services web portal, create at least one Virtual token to manage signing keys and certificates.

To create a Virtual Token in Entrust Certificate Services

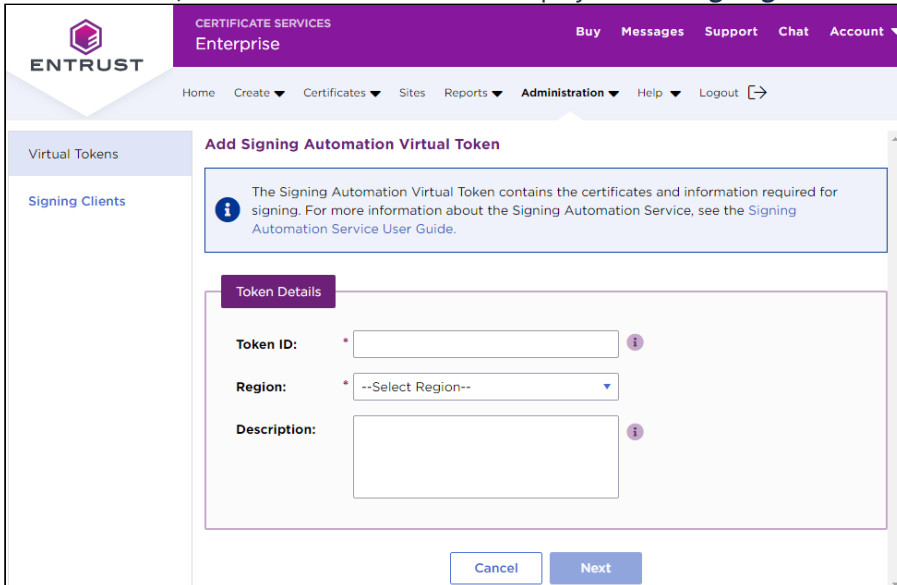
1. Log in to the Entrust Certificate Services web portal at cloud.entrust.net
2. Select **Administration > Signing Automation**.



3. In the sidebar, click on **Virtual Tokens**.



4. In the menu bar, click **+ Add Virtual Token** to display the **Add Signing Automation Virtual Token** form.



5. Enter the following values.

- [Token ID](#)
- [Region](#)
- [Description field](#)

6. Click **Next**.

7. Click **Submit** to confirm the Virtual Token creation.

Token ID

A 3-14 character name to uniquely identify the new Virtual Token.

i When processing the Signing Automation License, the Signing Client application will display this identifier in the Virtual Token selector.

Region

The region for storing the token keys. Select:

- **Code Signing Compliance (Canada)** for storing code signing keys.
- **US EAST (N. Virginia)** for storing document signing keys.

Description field

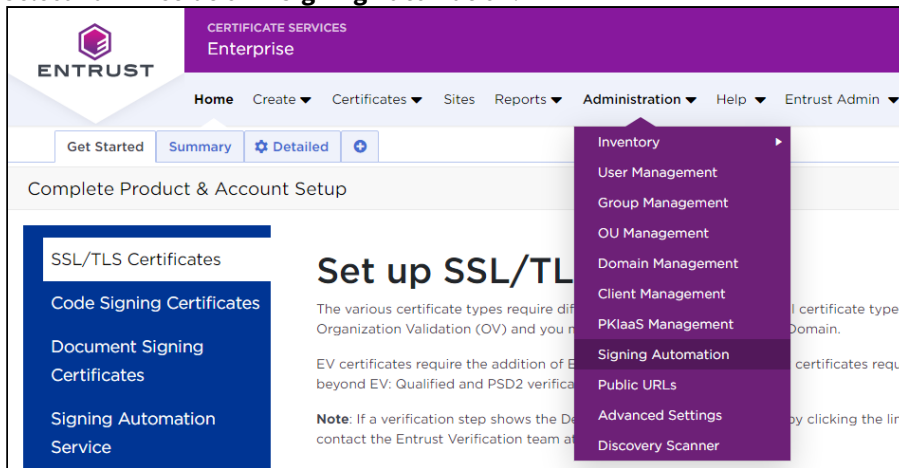
An optional description of the Virtual Token.

Creating a Signing Client

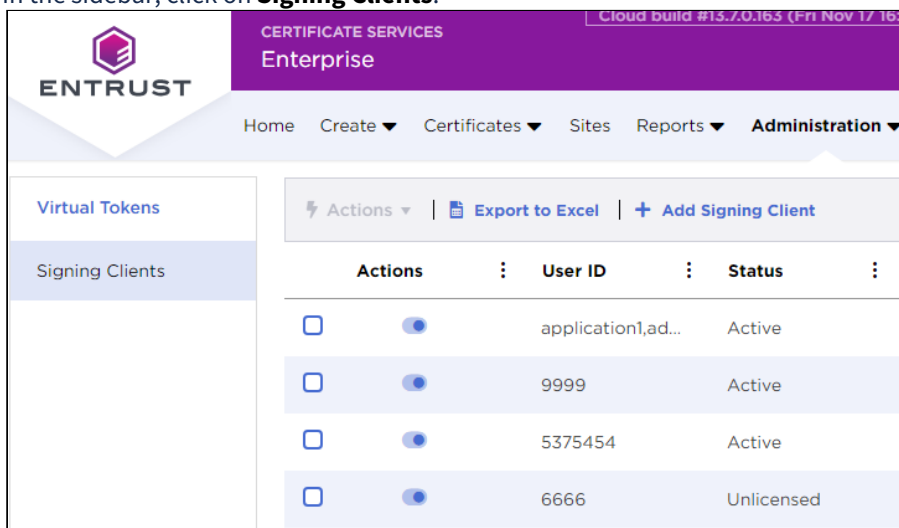
In the Entrust Certificate Services web portal, create a Signing Client with permission on the keys and certificates of a Virtual Token.

To create a Signing Client in Entrust Certificate Services

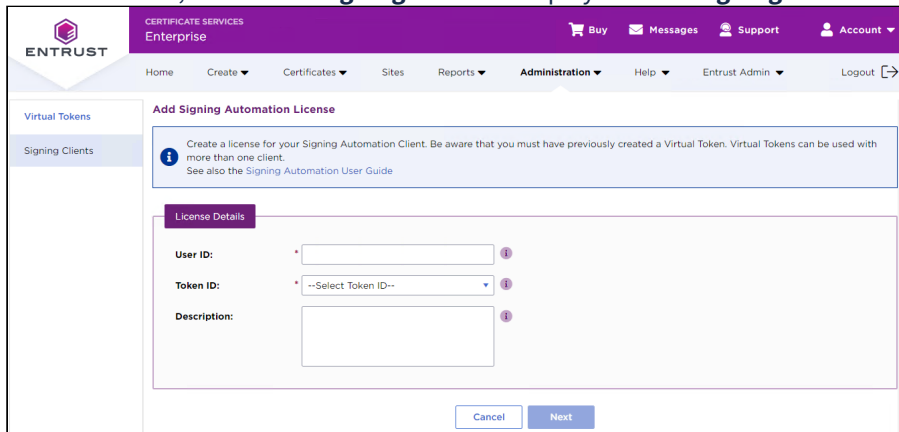
1. Log in to the Entrust Certificate Services web portal at cloud.entrust.net
2. Select **Administration > Signing Automation**.



3. In the sidebar, click on **Signing Clients**.



4. In the menu bar, click **+ Add Signing Client** to display the **Add Signing Automation License** form.



ENTRUST CERTIFICATE SERVICES Enterprise

Home Create Certificates Sites Reports Administration Help Entrust Admin Logout

Virtual Tokens

Signing Clients

Add Signing Automation License

Create a license for your Signing Automation Client. Be aware that you must have previously created a Virtual Token. Virtual Tokens can be used with more than one client. See also the Signing Automation User Guide

License Details

User ID:

Token ID: --Select Token ID--

Description:

Cancel Next

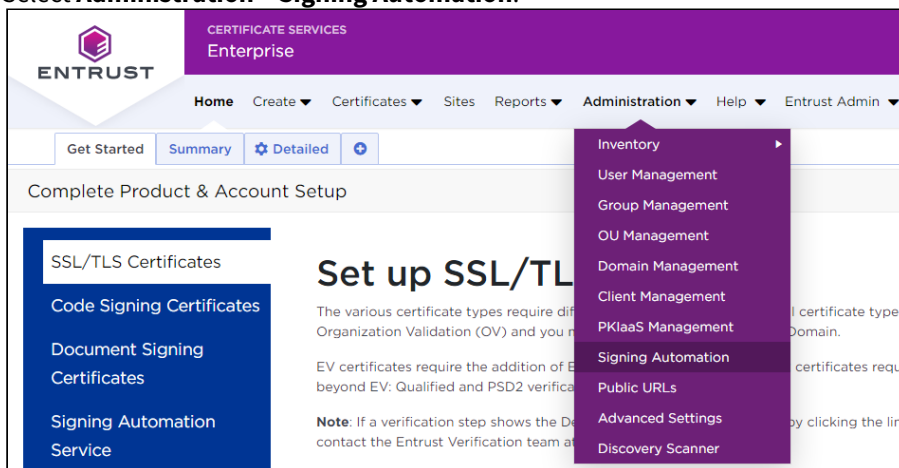
5. In the **User ID** field, enter a 3-14 character name to uniquely identify the new Signing Client.
6. In the **Token ID** list, select a Virtual Token for the new Signing Client. The Signing Client will be granted permission to sign with the keys and certificates of the selected Virtual Token.
7. In the **Description** field, enter an optional description of the new Signing Client.
8. Click **Next**.
9. Click **Submit** to confirm the Signing Client creation.

Downloading the Signing Automation License

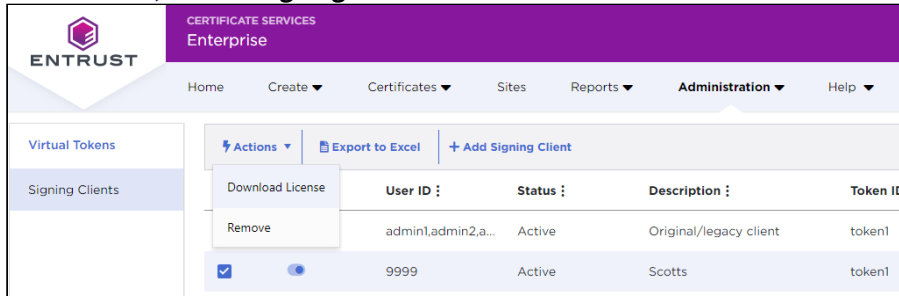
Download your Signing Automation License from the Entrust Certificate Services web portal.

To download the Signing Automation License

1. Log in to the Entrust Certificate Services web portal at cloud.entrust.net
2. Select **Administration > Signing Automation**.



- In the sidebar, click on **Signing Clients**.



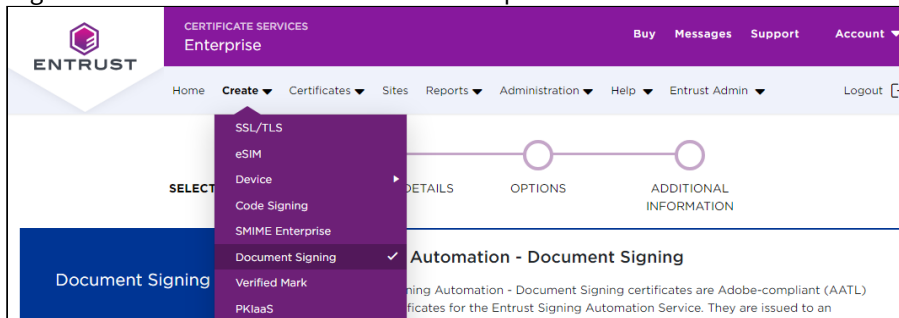
- In the Signing Clients grid, click the checkbox of a Signing Client row.
- In the menu bar, select **Actions > Download License**.
- In the confirmation dialog, click **Download**.

Creating a document signing certificate

Document signing requires a certificate specific to that purpose.

To create a document signing certificate

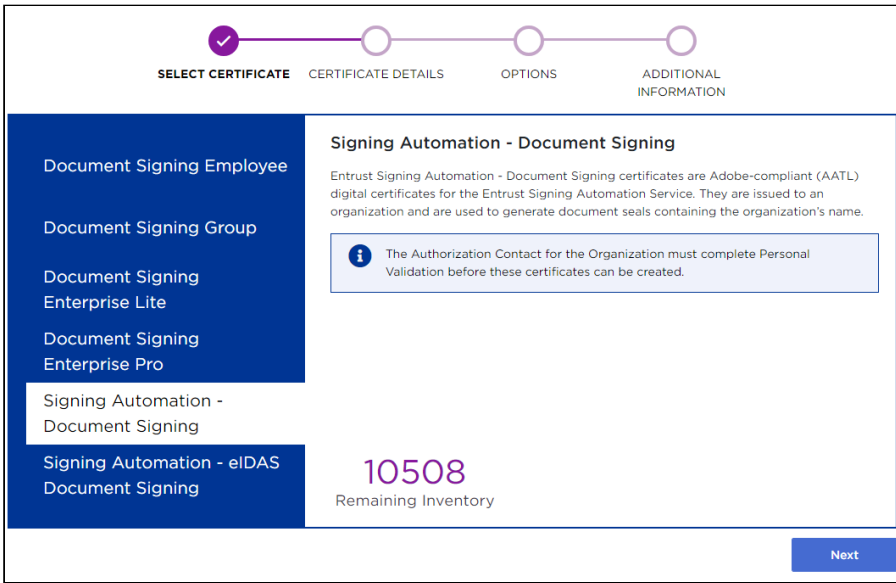
- Log in to the Entrust Certificate Services web portal at cloud.entrust.net.



- In the **Create** menu, select **Document Signing**.
- Follow the instructions on each page of the certificate creation wizard.
 - [SELECT CERTIFICATE](#)
 - [CERTIFICATE DETAILS](#)
 - [OPTIONS](#)
 - [ADDITIONAL INFORMATION](#)

SELECT CERTIFICATE

In the sidebar menu, click **Signing Automation - Document Signing**.



SELECT CERTIFICATE CERTIFICATE DETAILS OPTIONS ADDITIONAL INFORMATION

Document Signing Employee

Document Signing Group

Document Signing Enterprise Lite

Document Signing Enterprise Pro

Signing Automation - Document Signing

Signing Automation - eIDAS Document Signing

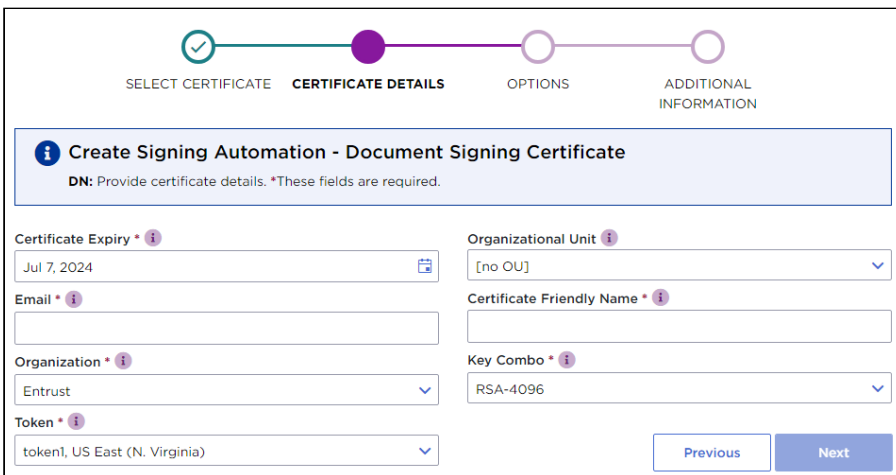
10508
Remaining Inventory

Next

In the **Remaining Inventory** section of this page, check the number of certificates you can create. Click **Next** if you still have remaining certificates.

CERTIFICATE DETAILS

Configure the new document signing certificate.



CREATE SIGNING AUTOMATION - DOCUMENT SIGNING CERTIFICATE
DN: Provide certificate details. *These fields are required.

Certificate Expiry * **i**
Jul 7, 2024

Organizational Unit **i**
[no OU]

Email * **i**

Certificate Friendly Name * **i**

Organization * **i**
Entrust

Key Combo * **i**
RSA-4096

Token * **i**
token1, US East (N. Virginia)

Previous **Next**

See the table below for a description of each field.

Field	Value	Mandatory
Certificate Expiry	The end date of the certificate validity.	✓
Email	The email address of the certificate owner. Entrust will use the selected email address to send notifications on the certificate status.	✓

Field	Value	Mandatory
Organization	The organization of the certificate owner.	✓
Organizational Unit	The unit within the organization (if any)	
Token	The Virtual Token described in Creating a Virtual Token .	✓
Certificate Friendly Name	A label for the keystores to identify the certificate when Signing with third-party signing applications . Spaces or special characters may cause errors.	✓
Key Combo	The type and length of the certificate public key.	✓

Click **Next**.

OPTIONS

This page does not display additional options for the document signing certificates. Click **Next**.

ADDITIONAL INFORMATION

Configure additional settings selected by your certificate services administrator.

✓ — ✓ — ✓ — ADDITIONAL INFORMATION

✓ **Create Signing Automation - Document Signing Certificate**
DN: email=david@dmjtest.com, cn=Entrust, o=Entrust, l=Ottawa, st=Ontario, c=CA

Provide additional information about this certificate. *These fields are required.

<p>Additional Emails ⓘ</p> <input style="width: 90%;" type="text" value="Enter additional emails, separated by commas (this field is optional)"/>	<p>Versions</p> <input style="width: 90%;" type="text"/>
<p>Count*</p> <input style="width: 90%;" type="text"/>	<p>When*</p> <input style="width: 90%;" type="text"/>
<p>Where*</p> <input style="width: 90%;" type="text" value="Select a value..."/>	<p>Who</p> <input style="width: 90%;" type="text"/>
<p>Fixing</p> <input style="width: 90%;" type="text"/>	
<p>OS category</p> <input style="width: 90%;" type="text" value="Select a value..."/>	

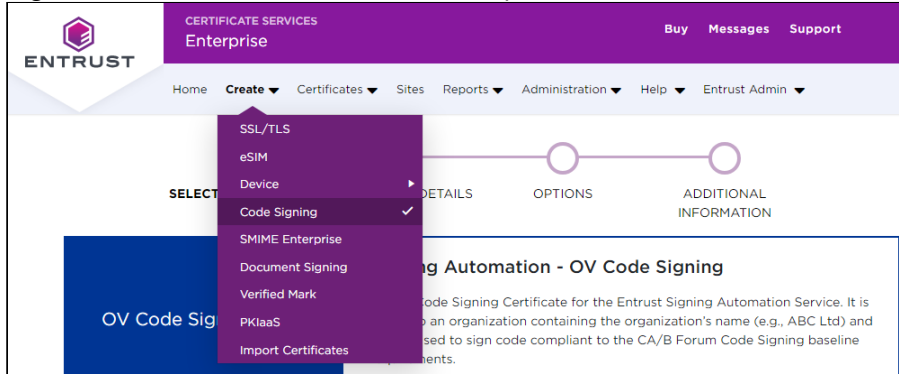
Click **Submit** and confirm the certificate creation.

Creating a code signing certificate

Code signing requires a certificate specific to that purpose.

To create a code signing certificate

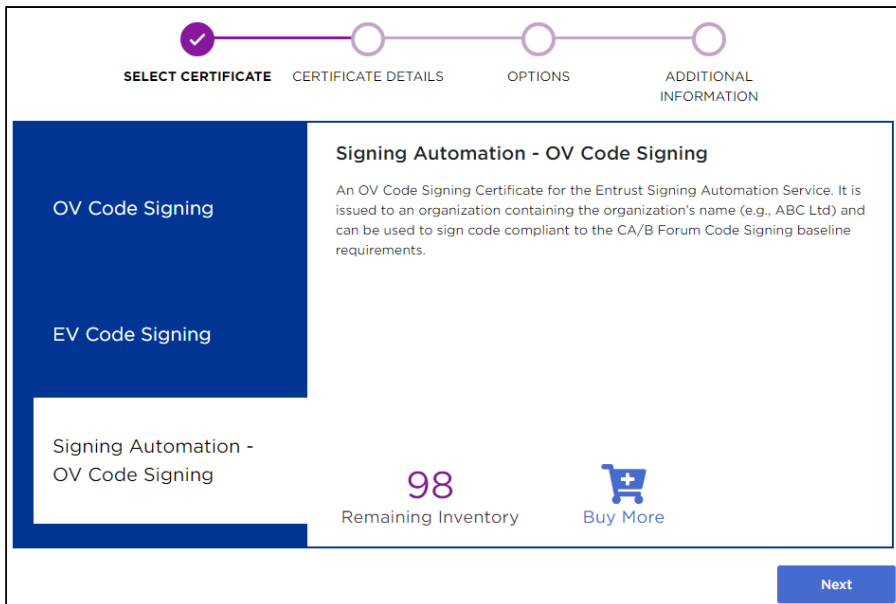
1. Log in to the Entrust Certificate Services web portal at cloud.entrust.net



2. Select **Create > Code Signing**.
3. Follow the instructions on each page of the certificate creation wizard.
 - [SELECT CERTIFICATE](#)
 - [CERTIFICATE DETAILS](#)
 - [OPTIONS](#)
 - [ADDITIONAL INFORMATION](#)

SELECT CERTIFICATE

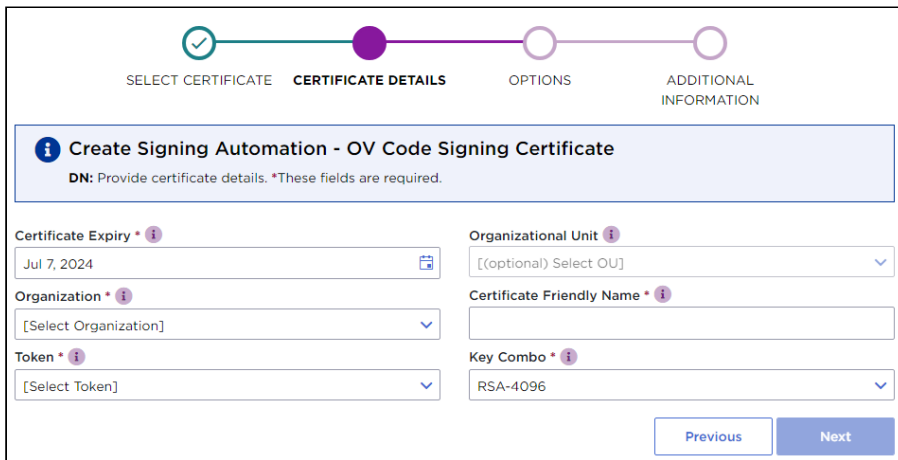
In the sidebar menu, click **Signing Automation - OV Code Signing**.



In the **Remaining Inventory** section of this page, check the number of certificates you can create. Click **Next** if you still have remaining certificates.

CERTIFICATE DETAILS

Configure the new document signing certificate.



See the table below for a description of each field.

Field	Value	Mandatory
Certificate Expiry	The end date of the certificate validity.	✓
Organization	The organization of the certificate owner.	✓
Organization Unit	The unit within the organization (if any).	
Token	The Virtual Token described in Creating a Virtual Token .	✓
Certificate Friendly Name	A label for the keystores to identify the certificate when Signing with third-party signing applications . Spaces or special characters may cause errors.	✓
Key Combo	The type and length of the certificate public key.	✓

Click **Next**.

OPTIONS

This page does not display additional options for the document signing certificates. Click **Next**.

ADDITIONAL INFORMATION

Optionally, enter **Additional Emails** for receiving notifications on the certificate status.

Progress indicator: SELECT CERTIFICATE ✓ CERTIFICATE DETAILS ✓ OPTIONS ✓ **ADDITIONAL INFORMATION** ●

✓ **Create Signing Automation - OV Code Signing Certificate**
DN: cn=Entrust QA POB, o=Entrust QA POB, l=Ottawa, st=Ontario, c=CA

Provide additional information about this certificate. *These fields are required.

Additional Emails ⓘ

Enter additional emails, separated by commas (this field is optional)

[Previous](#) [Submit](#)

Click **Submit** to confirm the certificate creation.

4 Downloading and installing the Signing Automation Client

Download and install the Signing Automation Client.

- [Signing Automation Client requirements](#)
- [Downloading the Signing Automation Client](#)
- [Installing the Signing Automation Client](#)

Signing Automation Client requirements

You need the following environment to install and run the Signing Automation Client.

- [Operating system](#)
- [Oracle/Open JDK](#)

Operating system

See the table below for the supported operating systems.

Platform	Operating System
Windows	Windows 10, version 1909 (64-bit)
Linux	Ubuntu 20.04.x (64-bit)

Oracle/Open JDK

In the supported operating systems, you can configure the Signing Automation Client into the Java platform for Java applications integrated with the Sun PKCS #11 provider. With this provider, applications using the Java Cryptography Architecture (JCA) and the Java Cryptography Extension (JCE) APIs can access native PKCS #11 tokens/HSMs as our PKCS #11 Signing Client.

- For more information about Sun PKCS #11 in Java 8 LTS, see: <https://docs.oracle.com/javase/8/docs/technotes/guides/security/p11guide.html>
- Starting with Java 9, there were slight changes in the Sun PKCS #11 integration in applications. You can find more details in the latest LTS version documentation, currently Java 11 LTS: <https://docs.oracle.com/en/java/javase/11/security/pkcs11-reference-guide1.html>

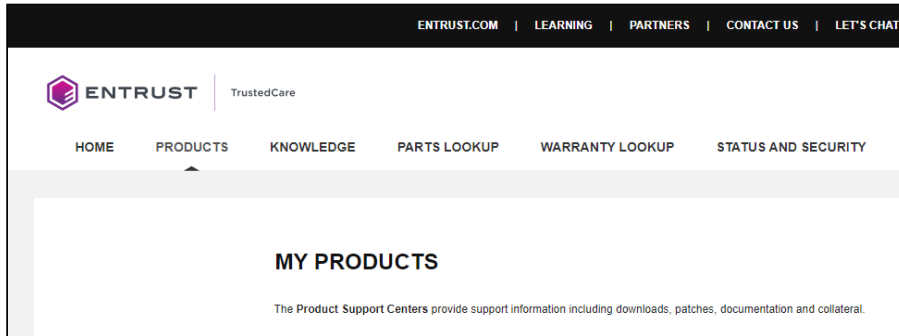
Downloading the Signing Automation Client

Download the Signing Automation Client installation file.

To download the Signing Automation Client

1. Log in to trustedcare.entrust.com

- Go to **PRODUCTS**.



The screenshot shows the Entrust website navigation bar with links for ENTRUST.COM, LEARNING, PARTNERS, CONTACT US, and LET'S CHAT. Below the navigation bar, the 'PRODUCTS' menu item is highlighted. The main content area displays 'MY PRODUCTS' with a sub-header: 'The Product Support Centers provide support information including downloads, patches, documentation and collateral.'

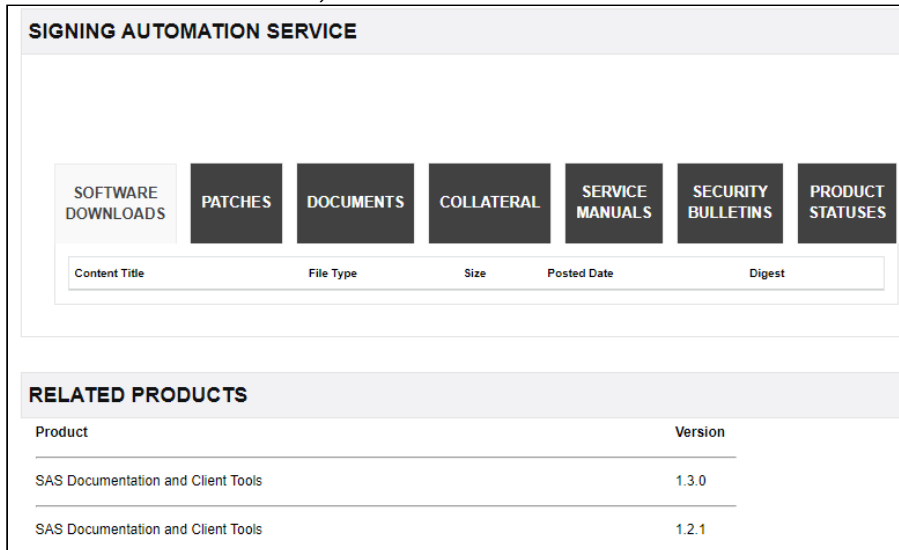
- Under **PUBLIC TRUSTED CERTIFICATES**, click **Digital Signing Services > Signing Automation Service**.



The screenshot shows the 'PUBLIC TRUST CERTIFICATES' section with a dropdown menu for 'Digital Signing Services'. Below the dropdown, there is a table with columns for Product, Version, and Supported Through.

Product	Version	Supported Through
Remote Signing Service		
Signing Automation Service		

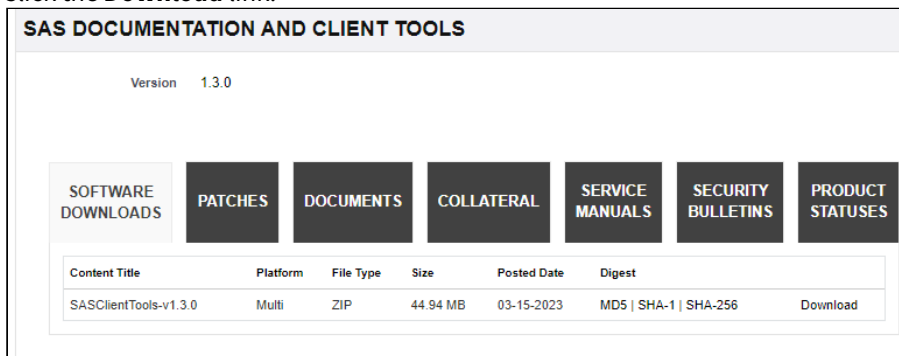
- Under **RELATED PRODUCTS**, click the latest **SAS Documentation and Client Tools** version.



The screenshot shows the 'SIGNING AUTOMATION SERVICE' section with a navigation bar for SOFTWARE DOWNLOADS, PATCHES, DOCUMENTS, COLLATERAL, SERVICE MANUALS, SECURITY BULLETINS, and PRODUCT STATUSES. Below this is a table with columns for Content Title, File Type, Size, Posted Date, and Digest. The 'RELATED PRODUCTS' section below shows a table with columns for Product and Version.

Product	Version
SAS Documentation and Client Tools	1.3.0
SAS Documentation and Client Tools	1.2.1

- Click the **Download** link.

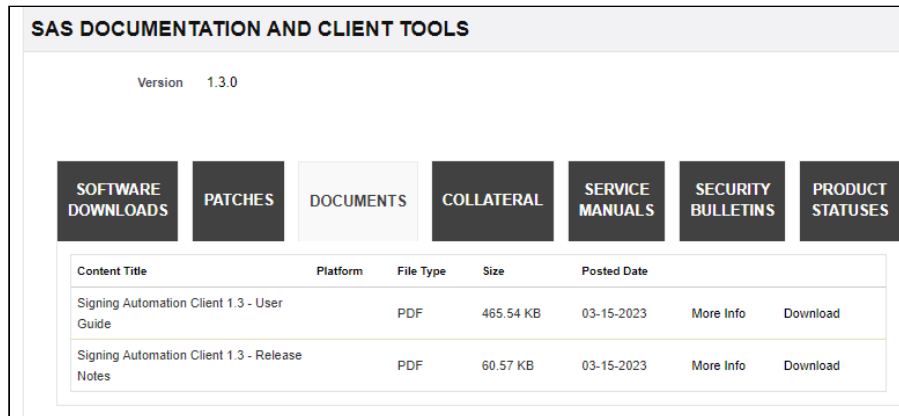


The screenshot shows the 'SAS DOCUMENTATION AND CLIENT TOOLS' section for version 1.3.0. It features a navigation bar for SOFTWARE DOWNLOADS, PATCHES, DOCUMENTS, COLLATERAL, SERVICE MANUALS, SECURITY BULLETINS, and PRODUCT STATUSES. Below this is a table with columns for Content Title, Platform, File Type, Size, Posted Date, Digest, and a Download link.

Content Title	Platform	File Type	Size	Posted Date	Digest	
SASClientTools-v1.3.0	Multi	ZIP	44.94 MB	03-15-2023	MD5 SHA-1 SHA-256	Download

- Click **ACCEPT** in the **License Agreement** to download a zipped file containing the Signing Automation Client installation files.

7. Click the **DOCUMENTS** tab.



Content Title	Platform	File Type	Size	Posted Date	More Info	Download
Signing Automation Client 1.3 - User Guide		PDF	465.54 KB	03-15-2023	More Info	Download
Signing Automation Client 1.3 - Release Notes		PDF	60.57 KB	03-15-2023	More Info	Download

8. Click the **Download** link for the latest Signing Automation Client documentation.

Installing the Signing Automation Client

Install the Signing Automation Client on your machine.

- [Installing the Signing Automation Client in Windows](#)
- [Installing the Signing Automation Client in Linux](#)

Installing the Signing Automation Client in Windows

To install the Signing Automation Client in Windows, run the installer as a user with administrator privileges.

```
SASDocAndClientTools\Software\SigningClient64.msi
```

The installer will add the application folder to the `PATH` variable.

```
C:\Program Files\Entrust\SigningClient
```

See the following table for the files in this folder.

File	Description
P11SigningClient64.dll	The application library for 64-bit Windows systems.
SigningClient.exe	The application executable.

Installing the Signing Automation Client in Linux

To install the Signing Automation Client in Linux platforms, extract the contents of the following ZIP file.

```
SASDocAndClientTools/Software/signingclient64.zip
```

This compressed file contains the following.

File	Description
libp11signingclient64.so	The application library for 64-bit systems
pkcs11-logger64.so	The logger library for 64-bit systems
signingclient	The application executable

5 Signing with third-party signing applications

Once installed, you can integrate the Signing Automation Client with a third-party signing application installed on-premises.

- [Requirements for signing with third-party signing applications](#)
- [Signing PDF documents with iText](#)
- [Signing Microsoft Authenticode files with Jsign](#)

i The Entrust Signing Automation Client on-premise software leverages the PKCS #11 Cryptographic Token Interface (Cryptoki) by RSA Security Inc. to authenticate into the Entrust Signing Automation Service and request cryptographic operations.

Requirements for signing with third-party signing applications

Before integrating a third-party signing application, you must perform the following steps.

- [Setting the P11PKIHUB_CONFIG variable](#)
- [Installing the Signing Automation License](#)

Setting the P11PKIHUB_CONFIG variable

If required, set the `P11PKIHUB_CONFIG` environment variable to modify the location for the user settings.

OS	Default user settings location
Linux	/home/\$USERNAME/.signingclient
Windows	C:\Users\%USERNAME%\AppData\Roaming\Entrust\SigningClient

Installing the Signing Automation License

Run the `signingclient process license` command to install the following types of Signing Automation License.

- Signed files with the `.lic` extension. See [Downloading the Signing Automation License](#) for how to obtain these files.
- Password-protected zip files. The signing client supports this legacy format for backward compatibility.

For example:

```
>SigningClient.exe process license intsasapi2w.lic
Organization ID: intsasapi2w
Choose the user to configure:
  1) admin1
Select number [1-1]: 1
User ID: admin1
Create a password to protect your credentials
New password:
Confirm password:
```



```

Writing credentials to: C:
\Users\romeror\AppData\Roaming\Entrust\SigningClient\credentials
Writing config to: C:\Users\romeror\AppData\Roaming\Entrust\SigningClient\config
Choose the token to configure:
  1) token82809
Select number [1-1]: 1
Token ID: token82809
Writing config to: C:\Users\romeror\AppData\Roaming\Entrust\SigningClient\config
Process license OK

```

See below for a description of the generated files.

- [config](#)
- [credentials](#)

 See [Setting the P11PKIHUB_CONFIG variable](#) for how to change the default path of these files.

config

The command-line tool saves the user settings in the following file when completing the license key installation.

OS	Default path
Windows	C:\Users\%USERNAME%\AppData\Roaming\Entrust\SigningClient\config
Linux	\$HOME/.signingclient/config

As explained in the [Command-line reference](#), you can manage these configuration settings with the following commands:

- [signingclient config list](#)
- [signingclient config set.](#)


For example, to update the library path:

```
$ signingclient config set --library /home/john/your_library_path/libp11pkihub.so
```

credentials

With the password entered by the administrator, the command-line tool encrypts the user secret in the following file.


OS	Default path
Windows	C:\Users\%USERNAME%\AppData\Roaming\Entrust\SigningClient\credentials
Linux	\$HOME/.signingclient/credentials

 We strongly recommend backing up this file.

Signing PDF documents with iText

Once installed and configured, you can integrate the Signing Automation Client with a signing application. For example, you can run the signature samples provided with the iText library for Java.

To run the iText signature samples

1. Configure a service subscription as explained in [Configuring signing automation in Entrust Certificate Services](#).
2. On your computer, install:
 - The Signing Automation Client, as explained in [Downloading and installing the Signing Automation Client](#).
 - Oracle JDK 8
 - The Eclipse IDE
3. Open a web browser in <https://github.com/itext/i7js-signatures> 
4. Click on **Code > Download ZIP**
5. Download the ZIP file on your computer.
6. Unzip the downloaded file into a working folder on your PC. For example:

```
C:\Projects\i7js-signatures-develop
```

7. Launch the Eclipse IDE.
8. Select **File > Import > General > Projects from Folder or Archive**.
9. In **Import source**, indicate the path of the unzipped iText samples.
10. Click **Finish**.
11. Add a subfolder to the project. For example:

```
C:\Projects\i7js-signatures-develop\config
```

12. In this new subfolder, create the files described in the following sections.
 - [hsm.properties](#)
 - [signkey.properties](#)
13. In the Eclipse navigation pane, browse to **i7js-signatures-develop > src/test/java > com.itextpdf.samples.signatures.chapter04**
14. In the navigation pane, double-click **C4_01_SignWithPKCS11HSM.java**
15. Update the file contents as explained in [C4_01_SignWithPKCS11HSM.java](#).
16. In the Eclipse navigation pane, right-click **C4_01_SignWithPKCS11HSM.java** and select **Run As > Java Application**.
17. Open the generated file with Adobe Acrobat Reader.

```
C:\Projects\i7js-signatures-develop\target\signatures\chapter04\hello_hsm.pdf
```

18. Check that the generated PDF file includes an embedded signature and a timestamp.

hsm.properties

Create an `hsm.properties` file with the following contents.

```
name = Entrust
library = C:\Program Files\Entrust\SigningClient\P11SigningClient64.dll
slot = 1
```

Where `library` is the path of the Signing Automation Client library. For additional variables, see:

<https://docs.oracle.com/javase/8/docs/technotes/guides/security/p11guide.html>

signkey.properties

Create a `signkey.properties` file with the following contents.

```
PASSWORD = <password>
PKCS11CFG = ./config/hsm.properties
```

Where:

- `PASSWORD` is the password of the logical token.
- `PKCS11CFG` is the path of the `hsm.properties` file.

C4_01_SignWithPKCS11HSM.java

Update the code of the `C4_01_SignWithPKCS11HSM.java` signature sample as explained in the following sections.

- [Updating the properties path](#)
- [Selecting the certificate](#)
- [Checking certificate selection](#)

Updating the properties path

The `C4_01_SignWithPKCS11HSM.java` code loads the `signkey.properties` file from the `C:/` root folder.

```
properties.load(new FileInputStream("C:/signkey.properties"));
```

Update this line to use the relative path of the `signkey.properties` file.

```
properties.load(new FileInputStream("./config/signkey.properties"));
```

Selecting the certificate

The following line of the `C4_01_SignWithPKCS11HSM.java` code selects the alias of the signing certificate.

```
String alias = ks.aliases().nextElement();
```

However, this code line may select the alias of an invalid certificate – for example, when the token includes successive renewals of the signing certificate. To update this code line, list all the certificate labels with the [signingclient list certificates](#) command.

If each certificate has a unique label, set this line as follows.

```
String alias = "<label>";
```

If different certificates share the same label, set this line as follows.

```
String alias = "<label>/<issuer>/<sn>"
```

Where:

- `<label>` is the label returned by the [signingclient list certificates](#) command.
- `<issuer>` is the DN (Distinguished Name) of the issuer's certificate.
- `<sn>` is the Serial Number of the certificate in decimal representation, as opposed to the hexadecimal representation printed by the [signingclient list certificates](#) command.

As explained in the [Oracle documentation](#):

If multiple certificates share the same CKA_LABEL, then the alias is derived from the CKA_LABEL plus the end entity certificate issuer and serial number ("MyCert/CN=foobar/1234", for example).

Checking certificate selection

Under this line of the `C4_01_SignWithPKCS11HSM.java` code.

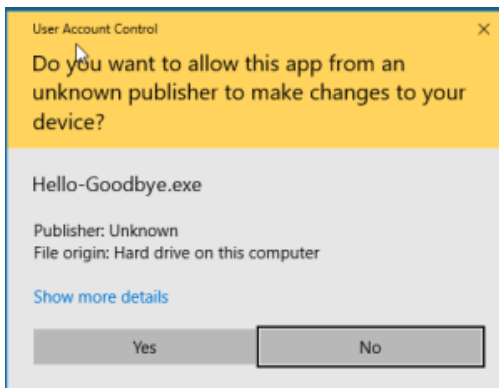
```
Certificate[] chain = ks.getCertificateChain(alias);
```

Add a clause to throw an exception when the selected alias does not correspond to a valid signing certificate.

```
if (pk == null || chain == null) {  
    throw new IllegalArgumentException("Couldn't find a certificate with the  
    specified label");  
}
```

Signing Microsoft Authenticode files with Jsign

Windows SmartScreen displays a warning like the following when an application executable is not signed.



As explained in ebourg.github.io/jsign, you can skip this warning by signing Windows executables with Jsign, a Java-based implementation of Microsoft Authenticode.

To sign Windows executables with Jsign

1. Configure a service subscription as explained in [Configuring signing automation in Entrust Certificate Services](#).
2. Install Signing Automation Client, as explained in [Downloading and installing the Signing Automation Client](#).
3. Follow the steps described below.
 - [Installing and configuring Jsign](#)
 - [Getting the token label](#)
 - [Testing the signing settings](#)
 - [Signing a Windows executable](#)
 - [Verifying the signature](#)

Installing and configuring Jsign

Install and configure Jsign on your machine.

- [Installing and configuring Jsign on Linux](#)
- [Installing and configuring Jsign on Windows](#)

Installing and configuring Jsign on Linux

Run the following commands to install Jsign on a Linux machine.

```
$ curl -fSsLL https://github.com/ebourg/jsign/releases/download/5.0/jsign-5.0-1.noarch.rpm
$ sudo dnf localinstall jsign-5.0-1.noarch.rpm -y
```

Create a configuration file – for example:

```
name = Entrust-CSaaS-PKCS11
description = Entrust CSaaS PKCS11 Driver
library = /home/eve/libp11signingclient64.so
slotListIndex = 0
```

Where the `library` is the path of the signing library described in [Installing the Signing Automation Client](#).

⚠ The examples below assume `csaas.cfg` as the name of this configuration file.

Installing and configuring Jsign on Windows

To install Jsign on a Windows machine:

1. Download the following file: <https://github.com/ebourg/jsign/releases/download/5.0/jsign-5.0.jar>
2. Move the file to a folder of your choice.
3. Press the Windows key.
4. Type **Edit the system environment variables** or **Edit environment variables for your account** depending on your user permissions.
5. Edit the **Path** environment variable.
6. Add the path of the folder containing the `jsign-5.0.jar` file.

Getting the token label

Run the `signingclient list keys` command to get the label of your code signing token.

```
$ signingclient list keys
Using token with label csaasdemo
Password:
Private Key Object; RSA 4096 bits
  Label:      csaasdemo
  ID:        876057e0fe3b0d53c822c312f4a7c76a76dd5644
  Usage:     sign

Public Key Object; RSA 4096 bits
  Label:      csaasdemo
  ID:        876057e0fe3b0d53c822c312f4a7c76a76dd5644
  Usage:     encrypt, verify, wrap
```

⚠ Your token might have more than one code signing key. Please make sure you select the right key.

Testing the signing settings

Run the `keytool -list` command to test:

- The configuration file.
- The access to the Virtual Token in Code Signing as a Service.

For example:

```
$ keytool -list -v -keystore NONE -storetype PKCS11 -providerClass
sun.security.pkcs11.SunPKCS11 -providerArg csaas.cfg
Enter keystore password:

Keystore type: PKCS11
Keystore provider: SunPKCS11-Entrust-CSaaS-PKCS11
```

```
Your keystore contains 1 entry
Alias name: csaasdemo
Entry type: PrivateKeyEntry
Certificate chain length: 3
Certificate[1]:
Owner: CN=Entrust Limited, SERIALNUMBER=1000492879, OID.2.5.4.15=Private
Organization, O=Entrust Limited, OID.1.3.6.1.4.1.311.60.2.1.2=Ontario,
OID.1.3.6.1.4.1.311.60.2.1.3=CA, L=Ottawa, ST=Ontario, C=CA
Issuer: CN=Entrust Extended Validation Code Signing CA - EVCS2, O="Entrust, Inc.",
C=US
Serial number: 60f0b081ee87f759afcc842c457fadb
```

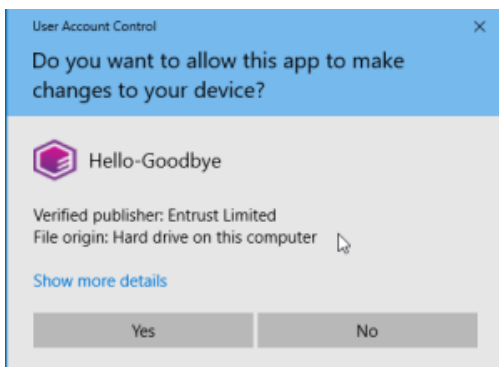
Signing a Windows executable

Run `jsign` to sign a Windows executable file – for example:

```
$ jsign --keystore csaas.cfg --alias csaasdemo -storepass xxxxxxxxxx --storetype
PKCS11 --tsurl http://timestamp.entrust.net/rfc3161ts2 --tsmode RFC3161 Hello-
Goodbye.exe
Adding Authenticode signature to Hello-Goodbye.exe
```

Verifying the signature

When running the signed executable on a Windows machine, the confirmation dialog will display information on the file signer.

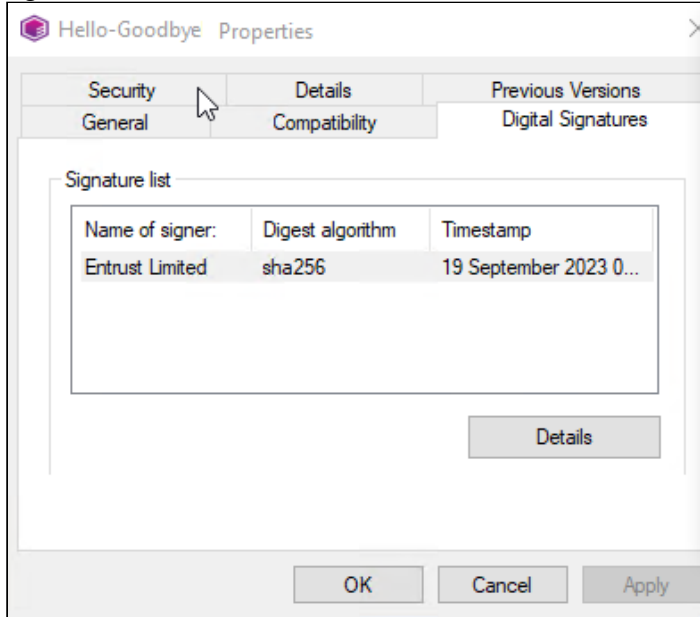


See below how to browse additional information

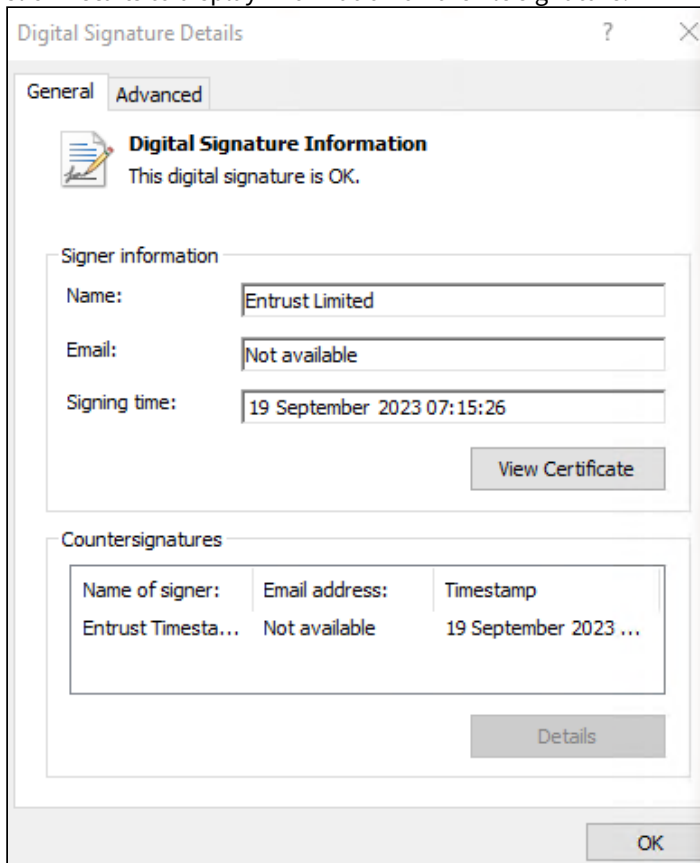
To browse information on a file signature

1. Locate the file using Windows File Explorer.

2. Right-click the file name and select the **Properties** contextual command to display the **Properties** dialog.



3. Click **Details** to display information on the file signature.



4. Click **View Certificate** to browse the signing certificate details.

6 Signing with REST clients

Once installed, you can use the Signing Automation Client to consume the REST API of the Entrust Signing Automation Service.

- [Getting API credentials for REST clients](#)
- [Signing data with curl commands](#)
- [Signing data with Postman](#)

See below the Swagger specification of each service exposed by the Signing Automation Service API.

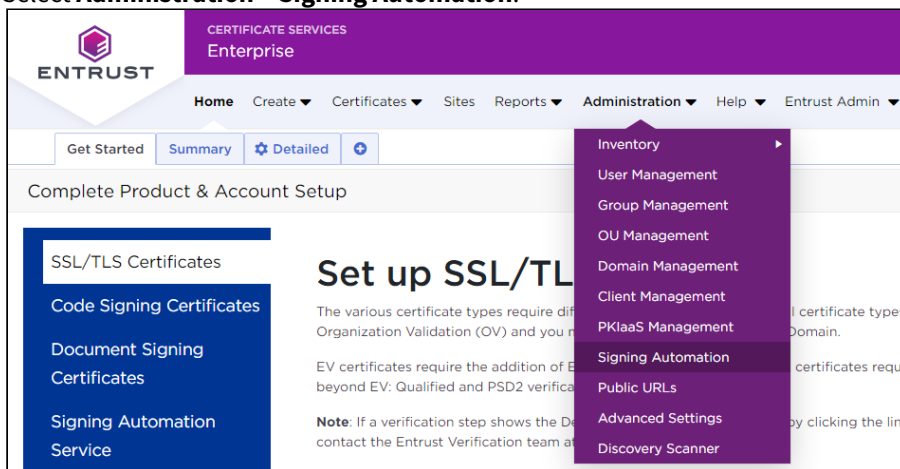
Service overview	Service Swagger reference
Identity Provider service	https://api.managed.entrust.com/sas/idp-api
Raw Signature service	https://api.managed.entrust.com/sas/rawsigner-api

Getting API credentials for REST clients

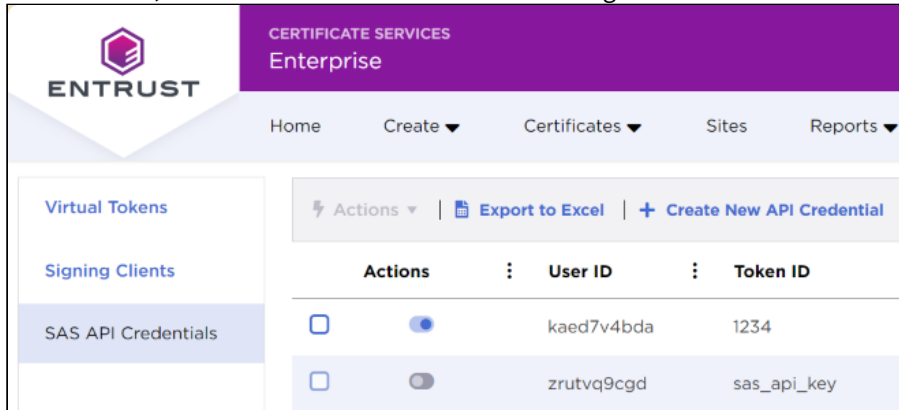
See below for obtaining API credentials for consuming the Signing Automation Service with REST clients.

To obtain API credentials for REST clients

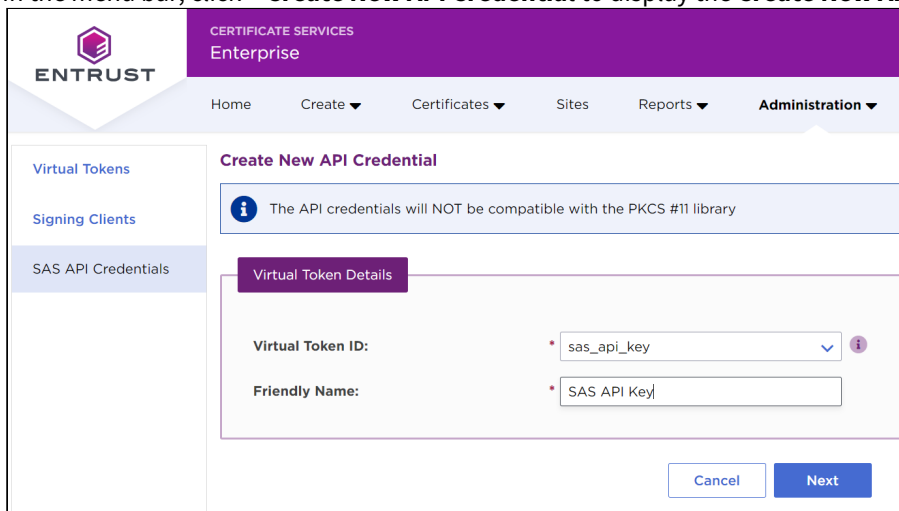
1. Log in to the Entrust Certificate Services web portal at cloud.entrust.net
2. Select **Administration > Signing Automation**.



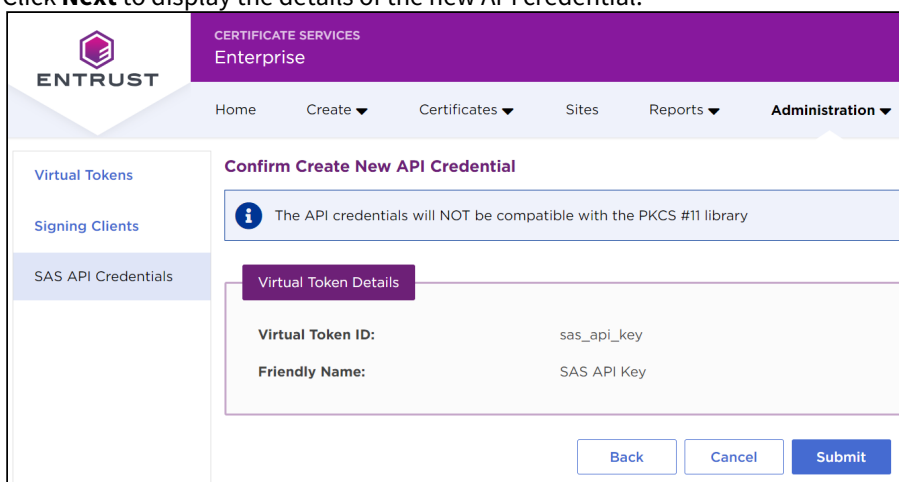
- In the sidebar, click on **SAS API Credentials** to list the generated SAS API credentials.



- In the menu bar, click **+ Create New API Credential** to display the **Create New API Credential** form.

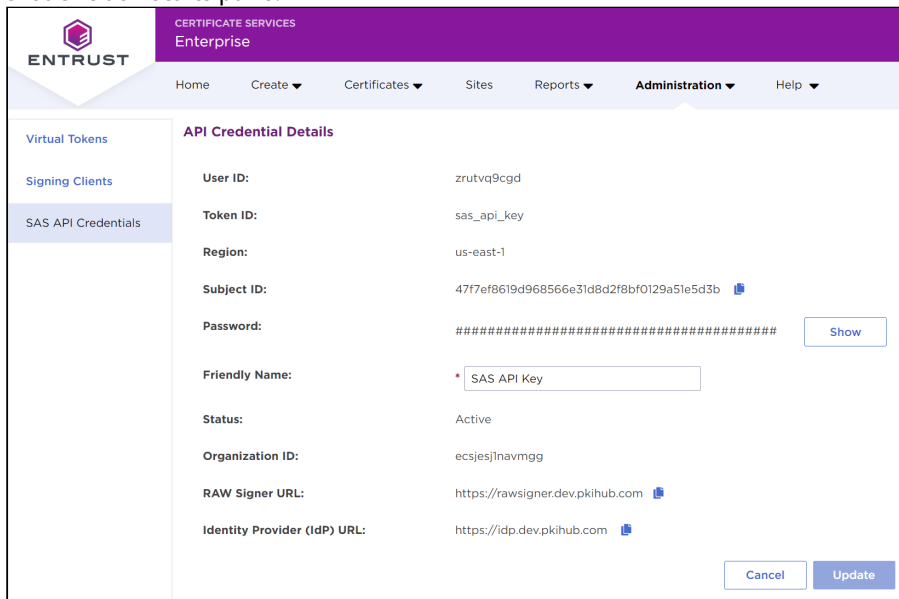


- In the **Virtual Token ID** field, enter a unique identifier for the new API credential.
- In the **Friendly Name** field, enter a descriptive name for the new API credential.
- Click **Next** to display the details of the new API credential.



- Check the API credentials details and click **Submit** to confirm the API credential creation.
- Click **OK** in the confirmation dialog and wait a few minutes while the API credential is generated.

10. Back to the list of generated API credentials, click the name of the new API credential to display the **API Credential Details** pane.



11. In the **API Credential Details** pane, copy the following credential values:

- Subject ID
- Password
- Organization ID

Signing data with curl commands

The following sections illustrate a step-by-step data signing with the `curl` command-line tool and the REST API of the Signing Automation Service.

- [Setting the environment variables](#)
- [Getting an authentication token](#)
- [Managing virtual tokens](#)
- [Managing signing keys](#)
- [Generating a data digest](#)
- [Signing a data digest](#)

 See [Identity Provider service](#) and [Raw Signature service](#) for a description of the exposed services.

Setting the environment variables

For ease of use, initialize environment variables with the license values previously obtained in [Getting API credentials for REST clients](#). Refer to the following table for the corresponding values based on the API credential generation mechanism.

Environment variable	Signing Automation Client	Entrust Certificate Services
SUBJECT_ID	subjectID	Subject ID

Environment variable	Signing Automation Client	Entrust Certificate Services
PASSWORD	secret	Password
PARTITION	Organization ID	Organization ID

For example:

```
$ export SUBJECT_ID=a08ba7436ecc63cdc09254c6768c2ddefac4e57b
$ export PASSWORD=1570b864d25c5102918066ccb2b470501b89872c
$ export PARTITION=intsasapi1
```

Getting an authentication token

Send a request to the [Identity Provider service](#) to get an [Authentication token](#).

```
$ export IDP_URL=https://idp.pkiaas.entrust.com
$ JWT="$(curl -s -X POST -u "$SUBJECT_ID:$PASSWORD" "$IDP_URL/idp/v1/tokens" | jq
-r .)" && export JWT
```

Check the token contents.

```
$ echo $JWT
```

When the token generation fails, you get an error like the following.

```
{ "code": 401, "message": "Unauthorized" }
```

Managing virtual tokens

Authenticate with the [Authentication token](#) in the [Raw Signature service](#) to list the [Virtual Tokens](#) of the user.

```
$ export SIGNING_SERVICE_URL=https://rawsigner.pkiaas.entrust.com
$ curl -s -H "Authorization: $JWT" "$SIGNING_SERVICE_URL/raw/partitions/$PARTITION/
tokens/" | jq .
```

Get information on a virtual token.

```
$ export TOKEN=token1
$ curl -s -H "Authorization: $JWT" "$SIGNING_SERVICE_URL/raw/partitions/$PARTITION/
tokens/$TOKEN" | jq .
```

Managing signing keys

Send a request to the [Raw Signature service](#) to list the identifiers of the signing keys within the Virtual Token.

```
$ curl -s -H "Authorization: $JWT" "$SIGNING_SERVICE_URL/raw/partitions/$PARTITION/tokens/$TOKEN/keys?quantity=10" | jq .
```

Get information on a signing key.

```
$ export KEY_ID=cbeeb264a4aa3db78b8abedba6dae7fcf3548c29
$ curl -s -H "Authorization: $JWT" "$SIGNING_SERVICE_URL/raw/partitions/$PARTITION/tokens/$TOKEN/keys/$KEY_ID" | jq .
```

Generating a data digest

Generate a base64-encoded hash of the data to be signed – for example, of the `data.txt` file contents:

```
$ export DIGEST_TBS=$(openssl sha256 -binary ./data.txt | base64)
```

Signing a data digest

Send a request to the [Raw Signature service](#) to sign the `$DIGEST_TBS` hash using the key with the `$KEY_ID` identifier.

```
$ curl -s -H "Authorization: $JWT" "$SIGNING_SERVICE_URL/raw/partitions/$PARTITION/tokens/$TOKEN/keys/$KEY_ID/signatures" -X POST -H "Content-type: application/json" -d '{"hashType": "'SHA256'", "hash": "'$DIGEST_TBS'"}' | jq .
```

Signing data with Postman

The following sections illustrate a step-by-step data signing with the Postman REST client the REST API of the Signing Automation Service.

- [Installing Postman](#)
- [Importing the service specifications](#)
- [Setting the environment variables](#)
- [Getting authentication tokens with pre-request scripts](#)

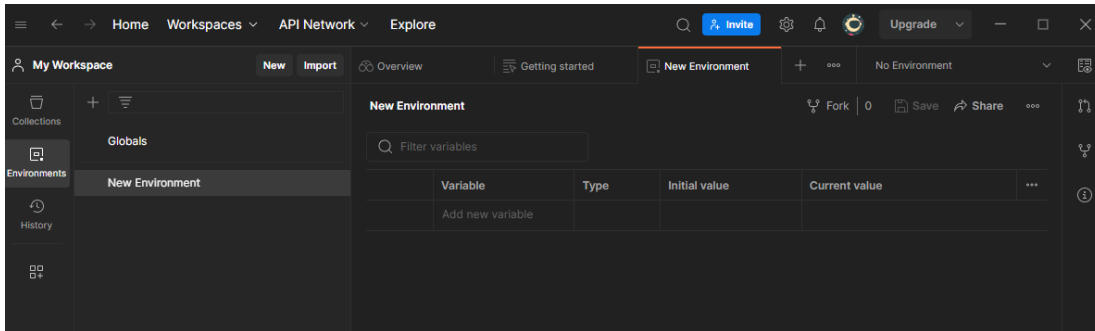
 See [Identity Provider service](#) and [Raw Signature service](#) for a description of the exposed services.

Installing Postman

Open the following URL to download Postman.

<https://www.postman.com/downloads>

Follow the installer instructions to create a user account and launch the application.



Importing the service specifications

Import the following service specifications.

OpenAPI specification of the Identity Provider service

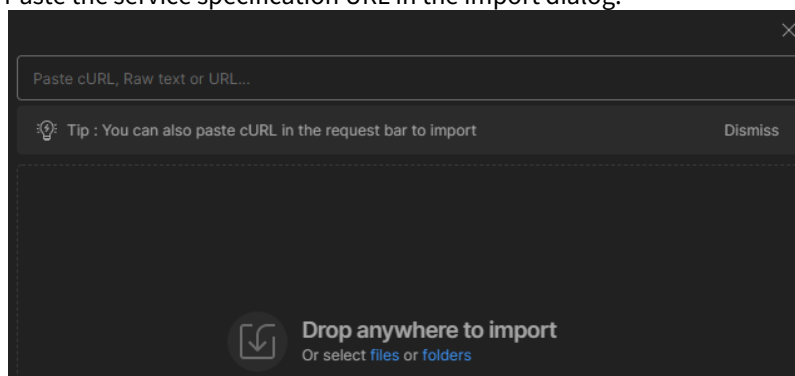
<https://api.managed.entrust.com/sas/idp-api/swagger-idp-api.json>

OpenAPI specification of the Identity Provider

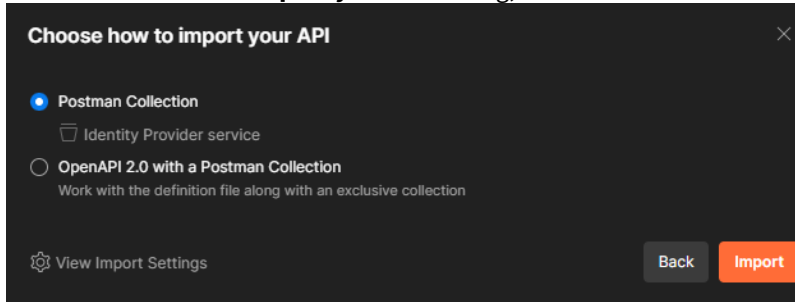
<https://api.managed.entrust.com/sas/rawsigner-api/swagger-rawsigner-api.json>

To import a service specification

1. Click **Import** in the Postman top menu bar.
2. Paste the service specification URL in the import dialog.



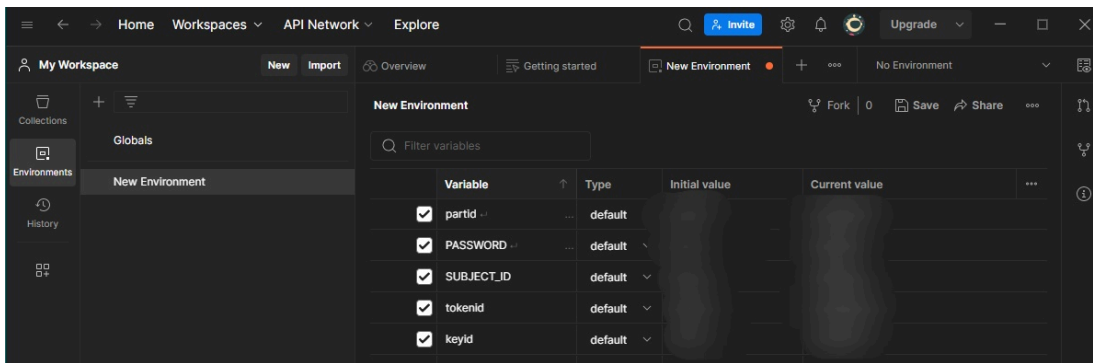
3. In the **Choose how to import your API** dialog, select **Postman Collection**.



4. Click **Import**.

Setting the environment variables

Click **Environments** in the Postman sidebar.



Initialize environment variables with the license values previously obtained in [Getting API credentials for REST clients](#). Refer to the following table for the corresponding values based on the API credential generation mechanism.

Name	Value	Entrust Certificate Services
partid	Organization ID	Subject ID
PASSWORD	Secret	Password
SUBJECT_ID	subjectID	Organization ID


Getting authentication tokens with pre-request scripts

In each request requiring an [Authentication token](#), click the **Pre-request Script** tab.



Paste the following code to generate an [Authentication token](#) before running the request.

```
// Get the username and password from Postman environment variables  
const username = pm.environment.get("SUBJECT_ID");  
const password = pm.environment.get("PASSWORD");  
  
// Construct the authorize header with a base64-encoded <username>:<password> pair  
const encodedCreds = btoa(`${username}:${password}`);  
const authHeader = `Basic ${encodedCreds}`;  
  
// Send a POST request to get JWT token  
pm.sendRequest({  
  url: 'https://idp.csaas.entrust.net/idp/v1/tokens',  
  method: 'POST',  
  header: {  
    'Accept': 'application/json',  
    'Authorization': authHeader  
  },  
  body: {}  
}, function (err, res) {  
  // Set the returned JWT value as jwt environment variable  
  pm.environment.set("jwt", res.json());  
});
```

 See <https://learning.postman.com/docs/writing-scripts/pre-request-scripts> for more details on pre-request scripts.

7 Debugging the Signing Automation Client

The out-of-the-box distribution of the Signing Client includes the `pkcs11-logger` library. In debugging mode, this library logs PKCS #11 operations by sitting between the Signing Application Client and the P11SigningClient64 library.

1. The Signing Application Client sends the PKCS #11 operation calls to the `pkcs11-logger` library.
2. The `pkcs11-logger` library redirects the calls to the P11SigningClient64 library.
3. The `pkcs11-logger` library returns the call results to the Signing Application Client.

See the project readme for a sample debugging output:

<https://github.com/Pkcs11Interop/pkcs11-logger/blob/master/README.md>

To enable the debugging mode, configure the following environment variables.

- `PKCS11_LOGGER_LIBRARY_PATH`
- `PKCS11_LOGGER_LOG_FILE_PATH`
- `PKCS11_LOGGER_FLAGS`

PKCS11_LOGGER_LIBRARY_PATH

The path to the following library, without the enclosing quotes.

```
P11SigningClient64
```

Run the `version` command to get this path.

PKCS11_LOGGER_LOG_FILE_PATH

The path to the log file, without the enclosing quotes.

PKCS11_LOGGER_FLAGS


The sum of requested logging flags.

Hex	Dec	Flag
0x01	1	Disables logging into the log file
0x02	2	Disables logging process identifiers.
0x04	4	Disables logging thread identifiers.
0x08	8	Enables logging personal identification numbers (PINs).
0x10	16	Enables logging to the standard output (stdout).

Hex	Dec	Flag
0x20	32	Enables logging to the standard error destination (stderr).
0x40	64	Enables reopening of the log file. This feature decreases performance but allows deleting the log file when needed.

For example, a value of 6 disables logging:

- The process identifier.
- The thread identifier.

 When not set, this variable defaults to 0.

8 Uninstalling the Signing Automation Client

To uninstall the Signing Automation Client in Windows, you can:


- Run the installer, and select **Remove**.
- Use the Windows **Add or remove programs** dialog

To uninstall the Signing Automation Client in Linux, remove the installation files.

9 Command-line reference

The `signingclient` executable of the Signing Client supports the commands described below.

- [signingclient completion](#)
- [signingclient config list](#)
- [signingclient config set](#)
- [signingclient create key](#)
- [signingclient credentials](#)
- [signingclient delete certificate](#)
- [signingclient delete key](#)
- [signingclient help](#)
- [signingclient import certificate](#)
- [signingclient list certificates](#)
- [signingclient list keys](#)
- [signingclient process license](#)
- [signingclient version](#)

 See [Installing the Signing Automation Client](#) for the path of the `signingclient` executable.

signingclient completion

Enables tab completion for `signingclient` commands.

```
signingclient completion <shell> [--log <file>] [--verbose]
```

See below for the supported options.

- `<shell>`
- `--log <file>`
- `--verbose`

<shell>

Generate a completion script for the `<shell>` shell. Where `<shell>` is one of the following:

- `bash`
- `fish`
- `powershell`
- `zsh`

Run the following command for instructions on how to install the generated script.

```
signingclient completion <shell> -h
```

For example:

```
$ signingclient completion bash -h
Generate the autocompletion script for the bash shell.
```

```
This script depends on the 'bash-completion' package.  
If it is not installed already, you can install it via your OS's package manager.
```

```
To load completions in your current shell session:
```

```
source <(signingclient completion bash)
```

```
To load completions for every new session, execute once:
```

```
#### Linux:
```

```
signingclient completion bash > /etc/bash_completion.d/signingclient
```

```
#### macOS:
```

```
signingclient completion bash > $(brew --prefix)/etc/bash_completion.d/  
signingclient
```

```
You will need to start a new shell for this setup to take effect.
```

Mandatory: Yes.

--log <file>

Record the command execution in the `<file>` log file.

- If the file does not exist, the command creates it.
- If the file exists, the command appends the execution log.

Mandatory: No. When omitting this option, the command does not record a log.

--verbose

Print additional error information (if any).

Mandatory: No.

signingclient config list

Lists the user settings.

```
signingclient config list [--from <date>] [--to <date>]
```

For example:

```
>signingclient config list  
User ID: application1  
Subject ID: 0123456789abcdef0123456789abcdef01234567  
Organization ID: dsaas999999  
Token ID: token1
```

```
PKCS #11 library: P11SigningClient64.dll
Signing server: https://rawsigner.dev.pkihub.com
IdP server: https://idp.dev.pkihub.com
Proxy server: <not set>
Proxy auth: <not set>
```

--log <file>

Record the command execution in the `<file>` log file.

- If the file does not exist, the command creates it.
- If the file exists, the command appends the execution log.

Mandatory: No. When omitting this option, the command does not record a log.

--verbose

Print additional error information (if any).

Mandatory: No.

signingclient config set

Modifies the user settings.

```
signingclient config set [--library <library>] [--log <file>] [--proxy-auth
<usr>:<pwd>] [--proxy-host <host>:<port>] [--token] [--verbose]
```

For example, to delete the proxy configuration:

```
signingclient config set --proxy-host "" --proxy-auth ""
```

See below for the supported options.

- `--library <library>`
- `--log <file>`
- `--proxy-auth <usr>:<pwd>`
- `--proxy-host <host>:<port>`
- `--token`
- `--verbose`

--library <library>

Select `<library>` as the PKCS #11 library.

Mandatory: No.

--log <file>

Record the command execution in the `<file>` log file.

- If the file does not exist, the command creates it.
- If the file exists, the command appends the execution log.

Mandatory: No. When omitting this option, the command does not record a log.

`--proxy-auth <usr>:<pwd>`

Authenticate in the proxy as the `<usr>` user with the `<pwd>` password.

Mandatory: No.

`--proxy-host <host>:<port>`

Select the `<host>:<port>` proxy.

Mandatory: No.

`--token`

List the available tokens for the user to select one.


Mandatory: No.

`--verbose`

Print additional error information (if any).

Mandatory: No.

signingclient create key

 This command is for administrator users only. In a normal scenario, Entrust Certificate Services automatically manages your keys and certificates.

Generates a key pair and the corresponding CSR (Certificate Signing Request).

```
signingclient create key --key-type <key_type> [--csr-out <csr>] [--csr-subject <subject>] [--key-id <id>] [--key-label <label>] [--log <file>] [--password <pwd>] [--verbose]
```

See below for the supported options.

- `--csr-out <csr>`
- `--csr-subject <subject>`
- `--key-id <id>`
- `--key-label <label>`
- `--key-type <key_type>`
- `--log <file>`
- `--password <pwd>`
- `--verbose`

i The command signs the CSR and, therefore, consumes one of the 10,000 licensed signatures.

--csr-out <csr>

Save the generated CSR in the <csr> file path.

Mandatory: No. When omitting this option, the command skips the CSR generation.

--csr-subject <subject>

Use <subject> as the Subject of the certificate request. Where <subject> is a full Distinguished Name (DN) or Relative Distinguished Name (RDN).

⚠ For Entrust Validation Authority to recognize the Subject, the DN attributes must be in capital letters.

For example:

```
CN=Example User,O=Example,C=US
```

```
CN=Example User
```

Mandatory: No. When omitting this option, the Subject in the generated certificate request defaults to the following:

```
CN=<key_id>
```

Where <key_id> is the key identifier.

--key-id <id>

Set <id> as the hexadecimal key identifier.

Mandatory: No. When omitting this option, the identifier is the public key's SHA1.

--key-label <label>

Set <label> as the key label.

Mandatory: No. When omitting this option, the label is the key identifier.

--key-type <key_type>

Create a key of the <key_type> type, where <key_type> is one of the following.

- RSA2048
- RSA3072
- RSA4096

- ECDSAP256
- ECDSAP384
- ECDSAP521

Mandatory: Yes.

--log <file>

Record the command execution in the <file> log file.

- If the file does not exist, the command creates it.
- If the file exists, the command appends the execution log.

Mandatory: No. When omitting this option, the command does not record a log.

--password <pwd>

Set <pwd> as the token password.

Mandatory: No. When omitting this option, the command prompts for the password value.

--verbose

Print additional error information (if any).

Mandatory: No.

signingclient credentials

Manages the user credentials

```
signingclient credentials --change-password [--log <file>] [--verbose]
```

See below for the supported options.

- [--change-password](#)
- [--log <file>](#)
- [--verbose](#)

--change-password

Change the user password - for example:

```
>signingclient credentials --change-password
Password:
Enter the new password to protect your credentials
New password:
Confirm password:
Writing credentials to: /home/user/.signingclient/credentials
Password changed
```

Mandatory: Yes. This is currently the only supported credential management option.

--log <file>

Record the command execution in the `<file>` log file.

- If the file does not exist, the command creates it.
- If the file exists, the command appends the execution log.


Mandatory: No. When omitting this option, the command does not record a log.

--verbose

Print additional error information (if any).

Mandatory: No.

signingclient delete certificate


 This command is for administrator users only. In a normal scenario, Entrust Certificate Services automatically manages your keys and certificates.

Deletes a certificate.

```
signingclient delete certificate --cert-id <cert_id> [--force] [--log <file>] [--verbose]
```

See below for the supported options.

- `--cert-id <cert_id>`
- `--force`
- `--log <file>`
- `--verbose`

 Deleting the certificates obtained when [Creating a document signing certificate](#) makes the Signing Automation Service unusable.

--cert-id <cert_id>

Delete the certificate with the `<id>` identifier.

Mandatory: Yes.

--force

Skip the confirmation before deleting the certificate.

Mandatory: No. When omitting this option, the command prompts for confirmation.

--log <file>

Record the command execution in the `<file>` log file.

- If the file does not exist, the command creates it.
- If the file exists, the command appends the execution log.


Mandatory: No. When omitting this option, the command does not record a log.

--verbose

Print additional error information (if any).

Mandatory: No.

signingclient delete key


 This command is for administrator users only. In a normal scenario, Entrust Certificate Services automatically manages your keys and certificates.

Deletes a key.

```
signingclient delete key --key-id <key_id> [--force] [--log <file>] [--verbose]
```

See below for the supported options.

- `--force`
- `--key-id <key_id>`
- `--log <file>`
- `--verbose`

 Deleting the key obtained in [Creating a document signing certificate](#) makes the Signing Automation Service unusable.

--force

Skip the confirmation before deleting the key.

Mandatory: No. When omitting this option, the command prompts for confirmation.

--key-id <key_id>

Delete the key with the `<id>` identifier.

Mandatory: Yes.

--log <file>

Record the command execution in the `<file>` log file.

- If the file does not exist, the command creates it.
- If the file exists, the command appends the execution log.

Mandatory: No. When omitting this option, the command does not record a log.

--verbose

Print additional error information (if any).

Mandatory: No.

signingclient help

Prints help information on `signingclient` subcommands. You can use any of the following syntaxes.

```
signingclient help <subcommands>
```

```
signingclient <subcommands> -h
```

```
signingclient <subcommands> --help
```


Where `<subcommands>` is the list of subcommands. For example, the following commands display the same help information on the `signingclient list certificates` command.

```
signingclient help list certificates
```

```
signingclient list certificates -h
```

```
signingclient list certificates --help
```

signingclient import certificate

 This command is for administrator users only. In a normal scenario, Entrust Certificate Services automatically manages your keys and certificates.

Imports a certificate.

```
signingclient import certificate <cert_file> [--cert-id <id>] [--cert-label <label>]  
[--no-trusted] [--log <file>] [--verbose]
```

See below for the supported options.

- `<cert_file>`
- `--cert-id <id>`
- `--cert-label <label>`
- `--log <file>`
- `--no-trusted`

- `--verbose`

`<cert_file>`

Import the certificate in the `<cert_file>` file path.

Mandatory: Yes.

`--cert-id <id>`


Set `<id>` as the hexadecimal identifier of the certificate.

Mandatory: No. When omitting this option, the identifier is:

- The public key identifier, if the certificate public key is in the token.
- The public key SHA1 otherwise.

`--cert-label <label>`

Set `<label>` as the certificate label.

 When [Signing with third-party signing applications](#), the keystore of the third-party application must use the same `<label>` label to identify the certificate.

Mandatory: No. When omitting this option, the certificate label is the certificate subject.

`--log <file>`

Record the command execution in the `<file>` log file.

- If the file does not exist, the command creates it.
- If the file exists, the command appends the execution log.

Mandatory: No. When omitting this option, the command does not record a log.

`--no-trusted`

Set `CKA_TRUSTED` to `false` in the certificate flags.

 Setting `CKA_TRUSTED` to `false` can prevent Java signing applications from working.

Mandatory: No. When omitting this option, `CKA_TRUSTED` is `true`.

`--verbose`

Print additional error information (if any).

Mandatory: No.

signingclient list certificates

List the token certificates.

```
signingclient list certificates [--log <file>] [--verbose]
```

--log <file>

Record the command execution in the `<file>` log file.

- If the file does not exist, the command creates it.
- If the file exists, the command appends the execution log.

Mandatory: No. When omitting this option, the command does not record a log.

--verbose

Print additional error information (if any).

Mandatory: No.

signingclient list keys

Lists the token keys.

```
signingclient list keys [--log <file>] [--verbose]
```

--log <file>

Record the command execution in the `<file>` log file.

- If the file does not exist, the command creates it.
- If the file exists, the command appends the execution log.

Mandatory: No. When omitting this option, the command does not record a log.

--verbose

Print additional error information (if any).

Mandatory: No.

signingclient process license

Imports a license key.

```
signingclient process license <license> [--all-available-users] [--api-credentials] [--force] [--log <file>] [--verbose]
```

See below for the supported options.

- <license>
- --all-available-users
- --api-credentials
- --force
- --log <file>
- --verbose


<license>

Process the <license> license, where <license> is the path license file. See [Downloading the Signing Automation License](#) for how to obtain this file.

Mandatory: Yes.

--all-available-users

Print the list of licensed users.

 This command is maintained for backward compatibility with legacy multi-user licenses.

Mandatory: No.

--api-credentials

Print credentials for the REST API instead of generating the [config](#) and [credentials](#) files described in [Installing the Signing Automation License](#). For example:

```
>signingclient process license intsasapi1.lic --api-credentials
Organization ID: intsasapi1
Choose the user to configure:
  1) admin1
Select number [1-1]: 1
User ID: admin1
WARNING: If you continue the license will be consumed and your credentials to access
the API will be printed on the screen.
      This will be the only time they will be displayed, so make sure you save
them securely.
NOTE: The API credentials will NOT be compatible with the PKCS #11 library
Continue? (y/n): y
{
  "subjectID": "a08ba7436ecc63cdc09254c6768c2ddeffac4e57b",
  "secret": "1570b864d25c5102918066ccb2b470501b89872c"
}
Process license OK
```

 The printed credentials are not valid for running the operations provided by the PKCS#11 library.

Mandatory: No. When omitting this flag, the command saves the [config](#) and [credentials](#) files described in [Installing the Signing Automation License](#).

--force

Overwrite the [config](#) and [credentials](#) files described in [Installing the license key](#).

 As explained in [Setting the P11PKIHUB_CONFIG variable](#), you can select the path of these files.

Mandatory: No. When omitting this flag, the application raises an exception if existing user files already exist.

--log <file>

Record the command execution in the `<file>` log file.

- If the file does not exist, the command creates it.
- If the file exists, the command appends the execution log.

Mandatory: No. When omitting this option, the command does not record a log.

--verbose

Print additional error information (if any).

Mandatory: No.

signingclient version

Prints the version of the Entrust Automation Signing Client.

```
signingclient version [--log <file>] [--verbose]
```

For example:

```
>signingclient version
CLI Version: v202103090152
Library: C:\Program Files\Entrust\SigningClient\P11SigningClient64.dll
Library Version: 1.0
Library Description: Signing Client v202102181732
```

--log <file>

Record the command execution in the `<file>` log file.

- If the file does not exist, the command creates it.
- If the file exists, the command appends the execution log.

Mandatory: No. When omitting this option, the command does not record a log.

--verbose

Print additional error information (if any).

Mandatory: No.