Code Signing Cert Setup

Each time you receive a new code signing certificate on a USB eToken, there are steps to perform and information to collect, so it can be utilized in an automated build pipeline.

There is little comprehensive documentation available on the web, for how to easily utilize a USB eToken for code signing.
This may simply be that such information needs to be obscured.
Or, that it is just not commonly used enough because it's not something "mortals" and casual programmers do.

What information is available is scattered and inconsistent.
So, this page attempts to consolidate those steps and list the data required to collect for a build signing tool (to access and use the token).

> NOTE: The certificate private key of your code-signing certificate cannot be exported from the USB token.
> So, all code-signing operations will require the USB eToken be attached to the build machine, and the SafeNet Authentication Client software be installed and active.

The code-signing certificate does with within a VSphere virtual machine, and additional steps are here: Adding a HSM Token to a VM

Precautions

1. The SafeNet Authentication Client software is designed for use by token creators (certificate vendors), not token users (certificate users).
   And, there is no functionality available to backup or restore the private key on a USB eToken.
   So, plenty of care and caution are necessary as the client software contains many functionalities that can reset and render a token inert, or overwrite a certificate, without any safeguards.
2. Above all else, do NOT execute a function called, Initialize Token. This will erase the certificate contents from your eToken, and you will have to buy a new certificate.
3. The USB eToken is configured to enter a lockdown state after a number of failed password attempts.
   And, SafeNet Authentication Client software has no reliable means to verify that a token password is correct.
   So, you need to verify that the token password is good using a signing tool.

Recommendations

There are not many actions (in the client software) required to properly use the USB eToken in automated signing.
All necessary interactions are listed below.

And since there is no means to backup/restore a code-signing certificate, it is both unnecessary and NOT recommended to explore the SafeNet Authentication Client software outside of the steps below.

Results

The below steps will create/retrieve all data elements required by an automated build signing tool. Here’s the list of data elements that will be recovered:

- Certificate File Path - location of the certificate’s public key file
- Token Password - required to access the token and use it for signing purposes
- Cryptographic Service Provider (CSP) - used by the build tool to identify the driver of the USB eToken
- Reader Name - Used by the build signing tool to identify the token. Not required if only one token is attached
- Container Name - used by the build signing tool to identify the private key of the code signing certificate

Required Steps

Here’s a compiled list of actions that must be done, to setup a new HSM token:

- Install SafeNet Authentication Client Software
- Fix CCID Issue (if signing within a VM)
- Check Token Presence
- Create New Token Password
- Export Signing Certificate
- Configure Authentication
- Record Identifying Parameters
- Verify Signing

Steps to follow

Install SafeNet Authentication Client Software

This will install the SafeNet Authentication Client Software.

Download the latest version of the SafeNet Authentication Client software from the vendor website.

> NOTE: The current version, as of 20211228, is V10.8 R6. It has been cached, here.

Fix CCID Issue

Once installed, we need to fix the CCID Issue (when inside a VM).

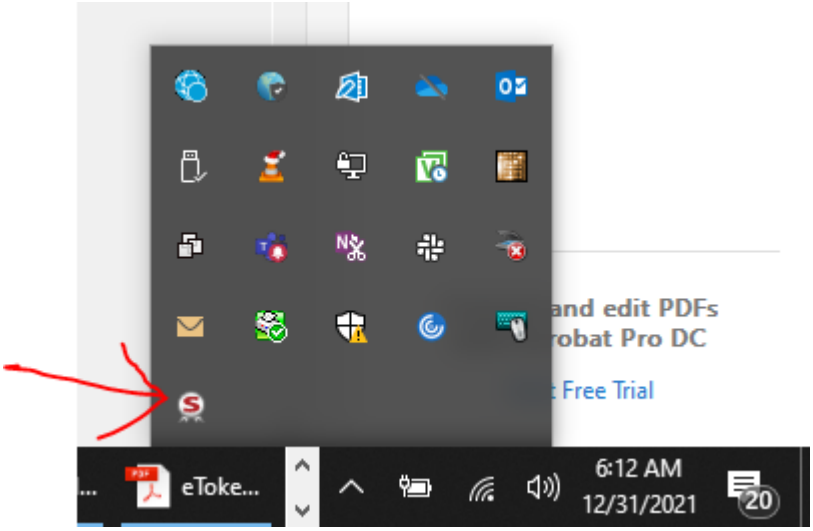
This is a problem associated with VMWare VSphere, where a USB Safenet token attached to the ESX host does not pass-through correctly to a guest VM.

The workaround for this is, Adding a HSM Token to a VM

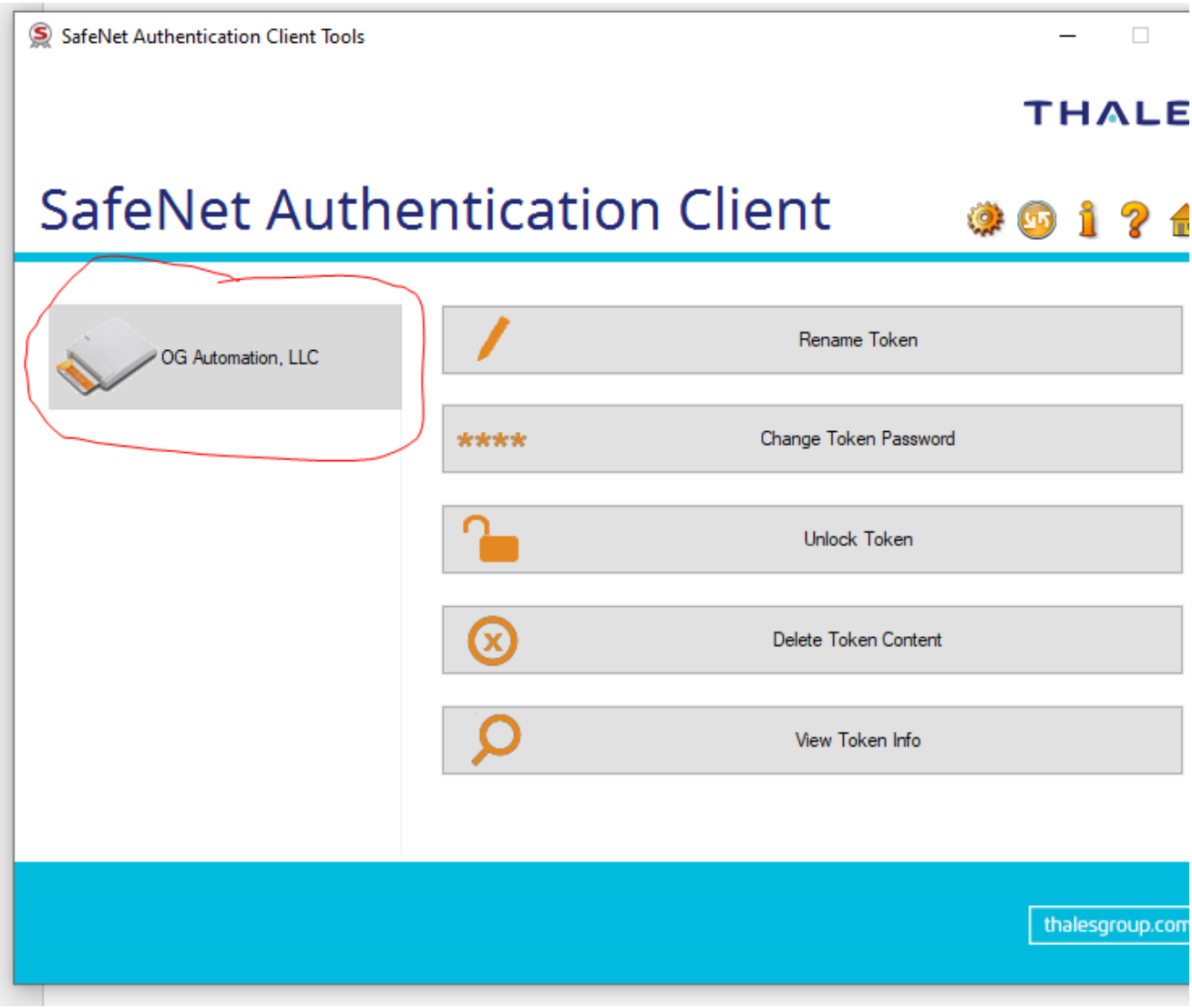
Check Token Presence

To check the presence of the eToken, do the following:

1. Make sure the eToken is attached to the machine (or connected via USB-Passthrough to the VM).
2. Double-click the SafeNet Authentication Client in the task bar.



3. If present, the token should appear in the left-hand pane of the client window.



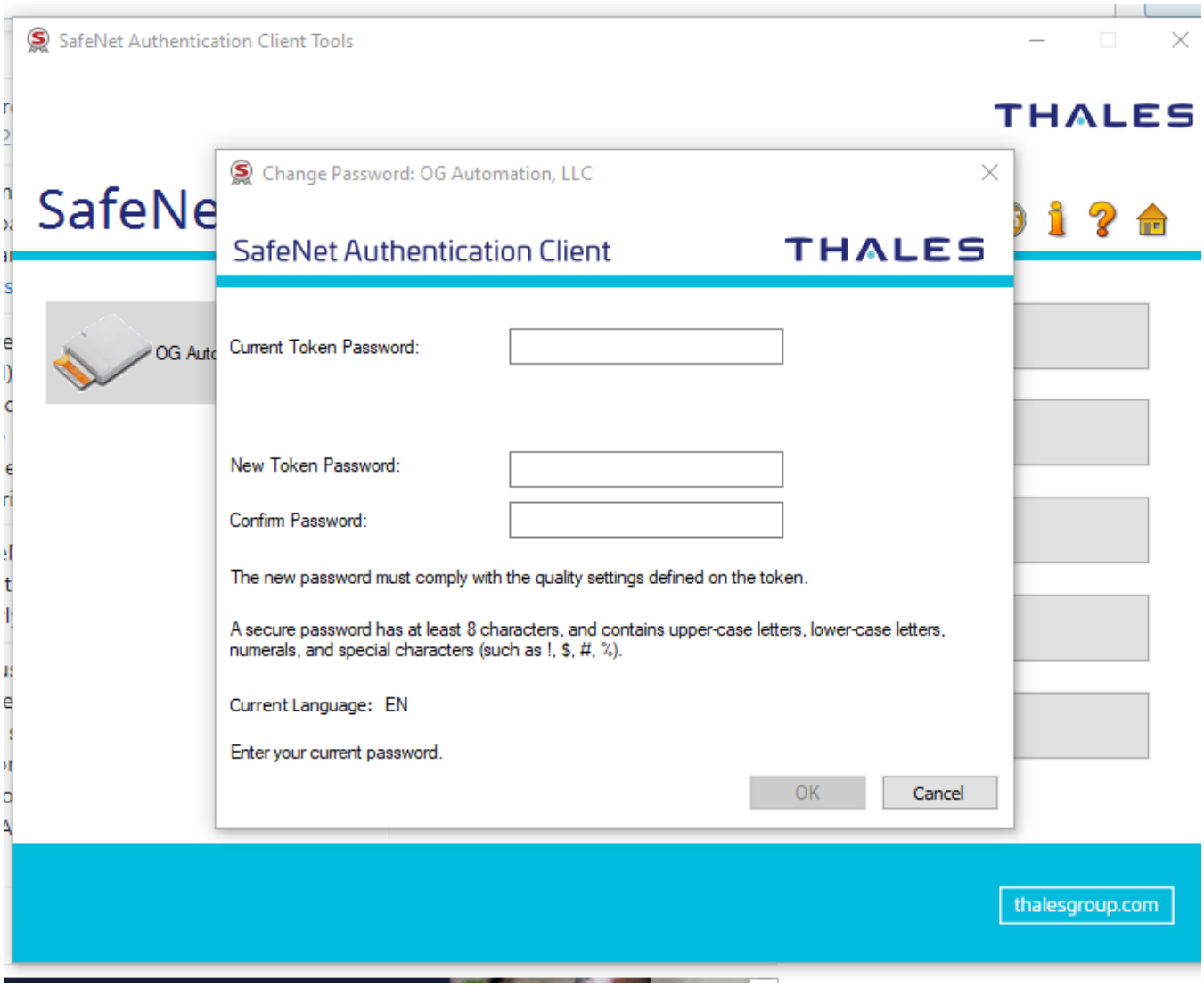
Create New Token Password

When a token is issued, you are given an initial token password.

This password is used to access your token.
This password should be changed to ensure no one has intercepted it along the way.
This password should be a very complex password. And, it is not required to memorize it, because it will be stored in a build automation tool.

To change the token password, do the following:

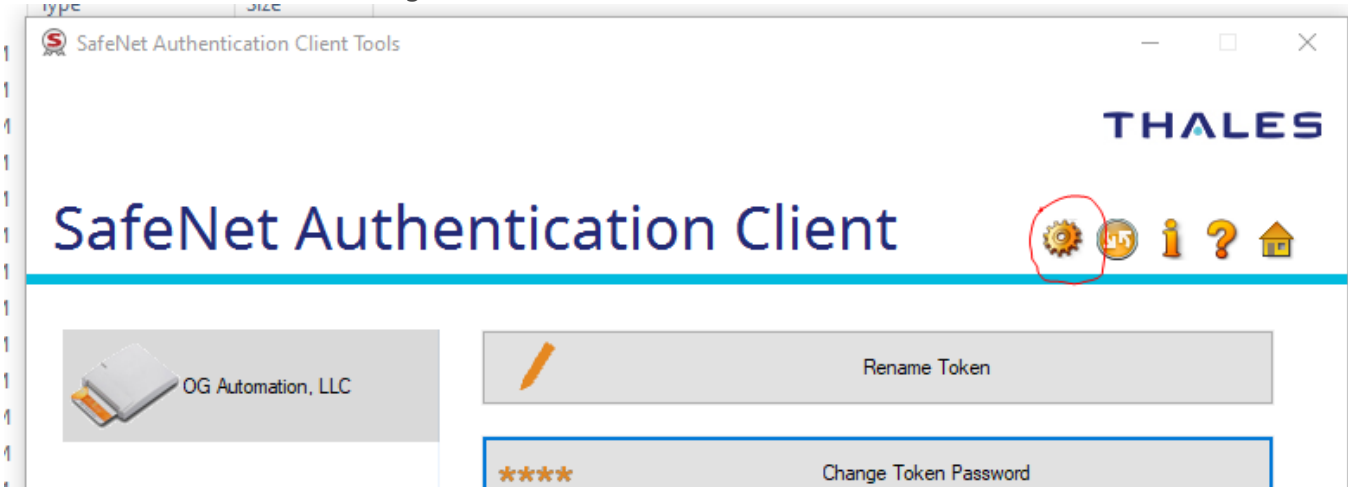
1. Formulate a new password to be used.
   It must exceed the complexity requirements (at least 8-characters, have upper and lower case letters, numbers, and special characters).
   Using a tool, such as LastPass Generator, or an online high-entropy password tool are useful for this.
2. Open the SafeNet client.
3. Select Change Token Password, and enter the new password at the dialog:



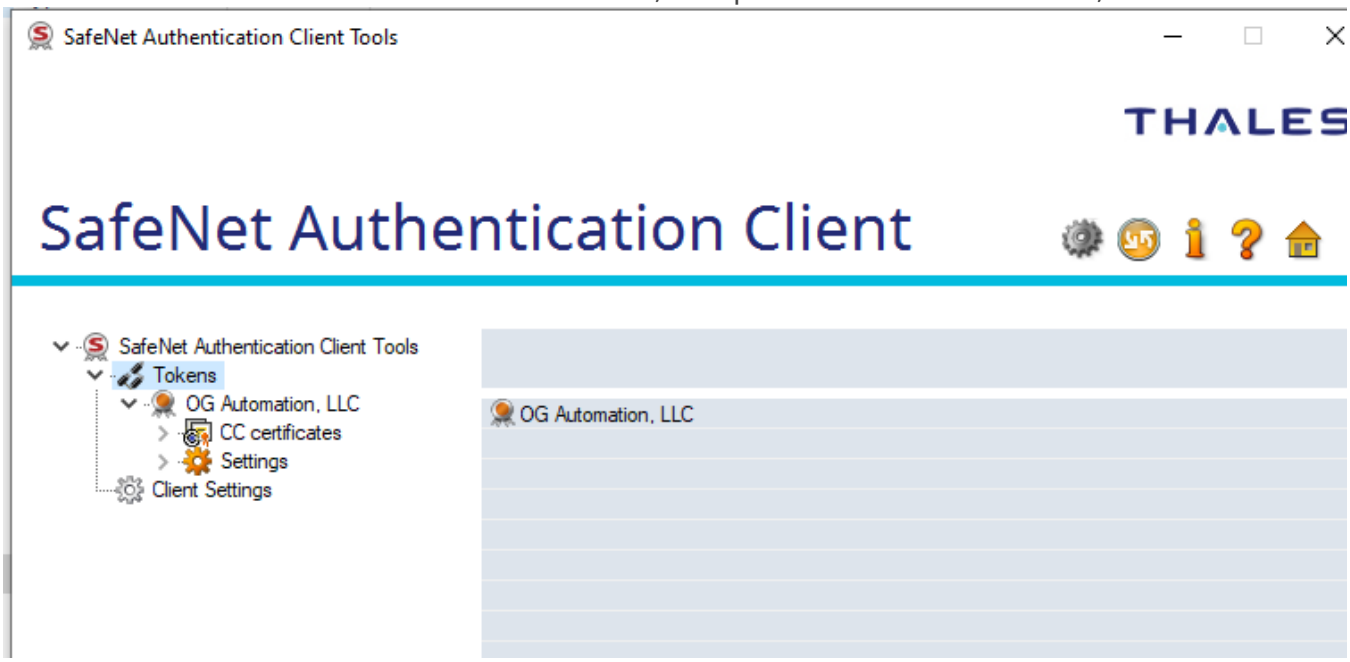
Export Signing Certificate

A build signing tool requires a copy of the code signing key's certificate.
So, we need to export it to a location that we can pass to the build signing tool.
Do the following to export the certificate:

1. Open the SafeNet Client.
2. Select the Gear Icon to change to Advanced Mode.

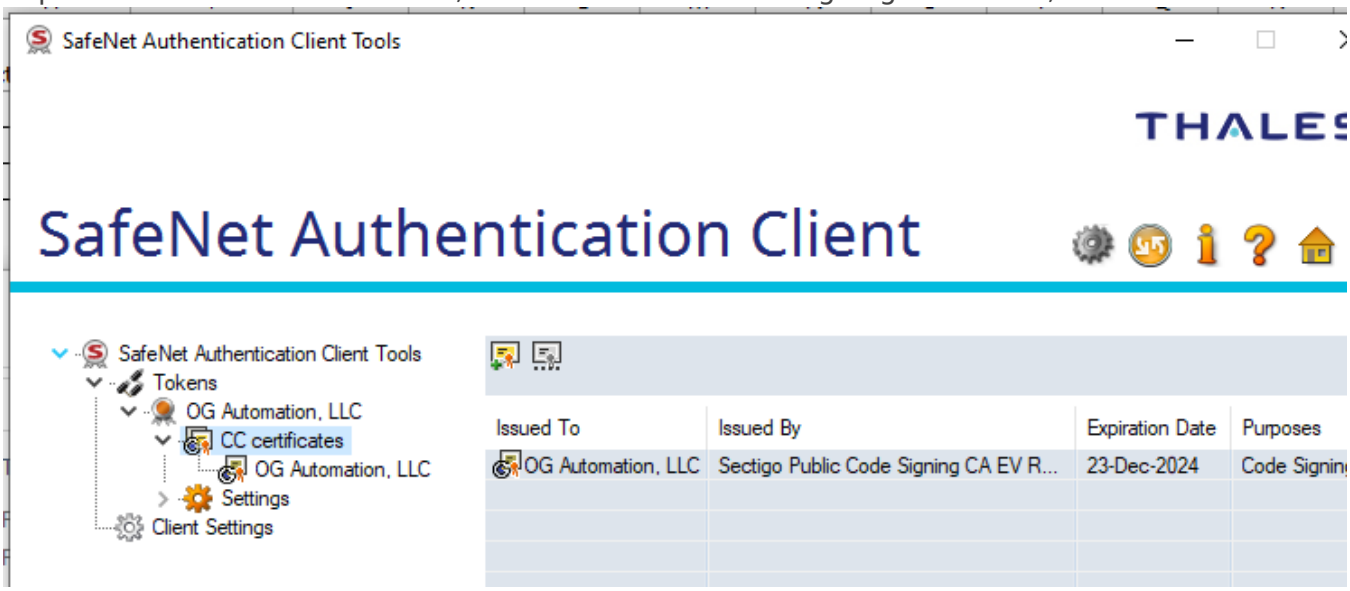


3. This will switch the client to its advanced view, and present a tree-view menu, like this.



The Tokens node of this menu, lists all tokens connected to the machine, with the certificate under the CC certificates node.

4. Expand the CC certificates node, to see the actual code-signing certificate, like this.



5. Right-click the certificate, and select the Export Certificate, to save the public certificate to a file.

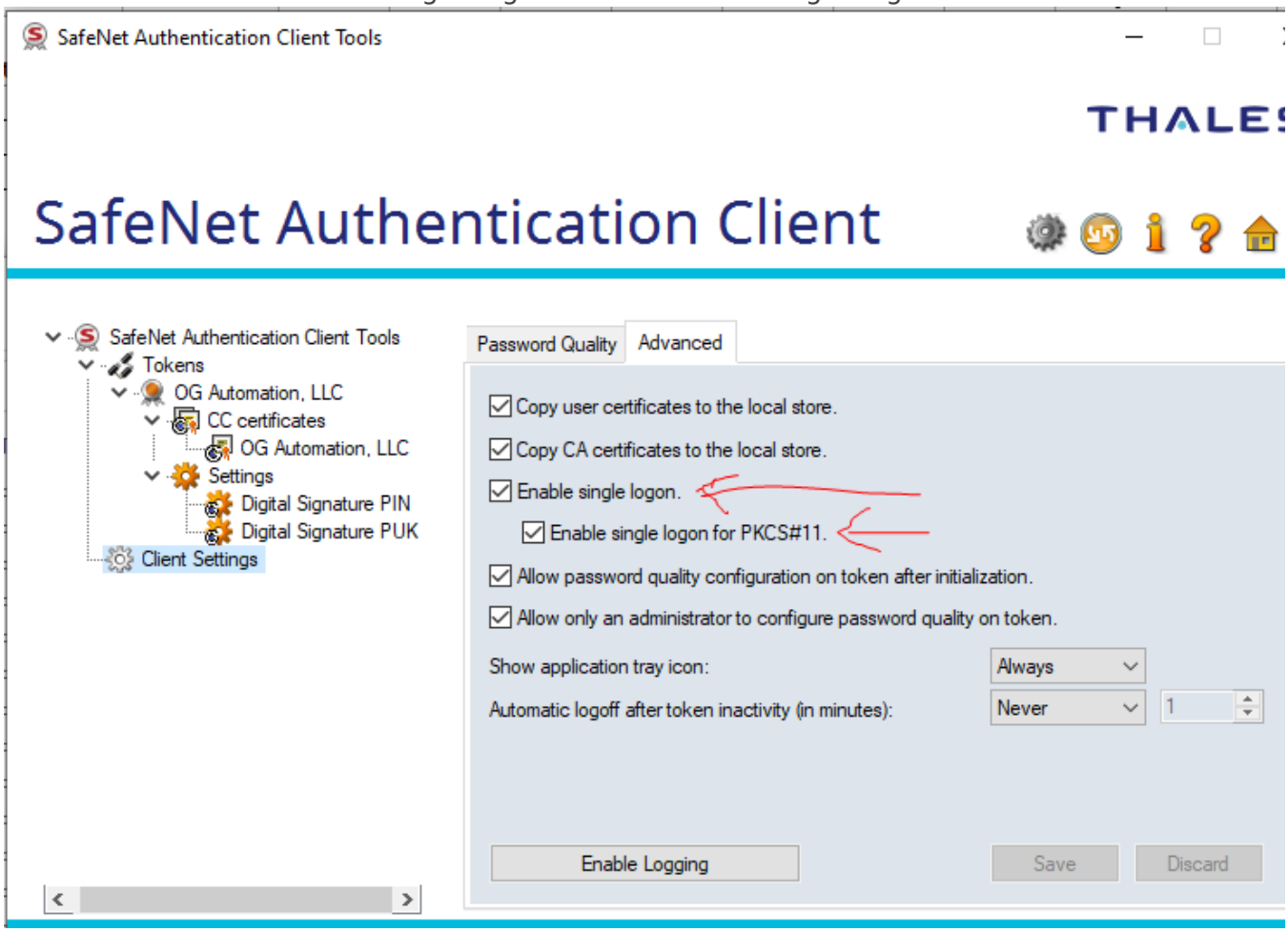
> NOTE: This file is required by an automated build signing tool. So, save it in an accessible path.

Configure Authentication

In order for an automated build tool to use the code signing certificate many times over long duration, we must adjust the authentication configuration for the token and software to allow single-sign on and to not Auto-logoff with inactivity.

> NOTE: This step may not be necessary since we provide the token password in commandline arguments to the build tool.
> Further testing is needed to see if this step can be omitted.

1. Open the Advanced View in the SafeNet Authentication Client.
2. Click on Client Settings and the Advanced Tab.
3. Check the boxes for Enable Single Logon and for Enable Single Logon for PKCS#11.



4. Click Save to save settings.
5. The documentation says this change requires a session update to take effect.
   But, this doesn't seem to always be the case.
   And, we need to also disable auto-logoff in the next step. So, we will continue, and reboot after that.
6. Make sure the Automatic Logoff duration is set to Never.
   This ensure we will remain logged into the token for a long-duration.
7. Reboot the machine to ensure the changes take effect.

Record Identifying Parameters

An automated build tool requires a few parameters to access and use the private key of the code signing certificate.

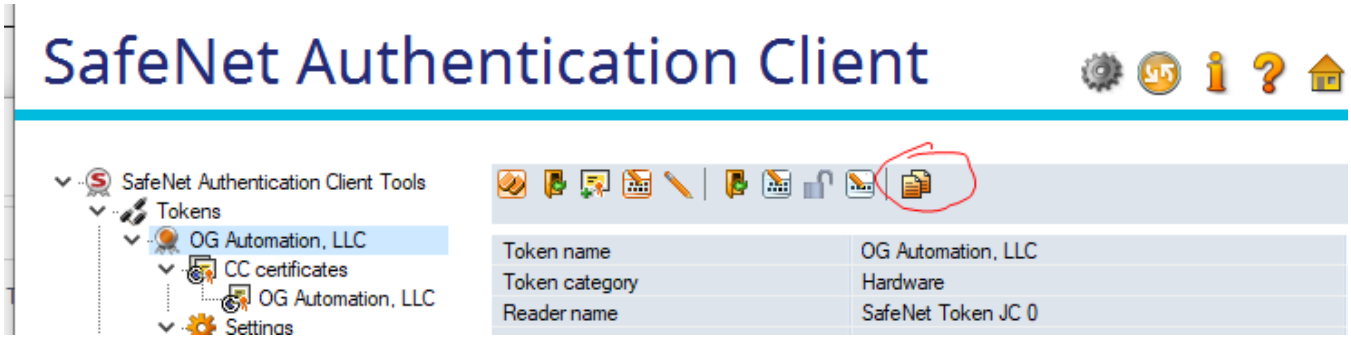
These are the "CSP", "Reader Name", and the "Container Name", and are all accessible from the SafeNet Authentication Client.

Here’s a list of steps to retrieve each of these:

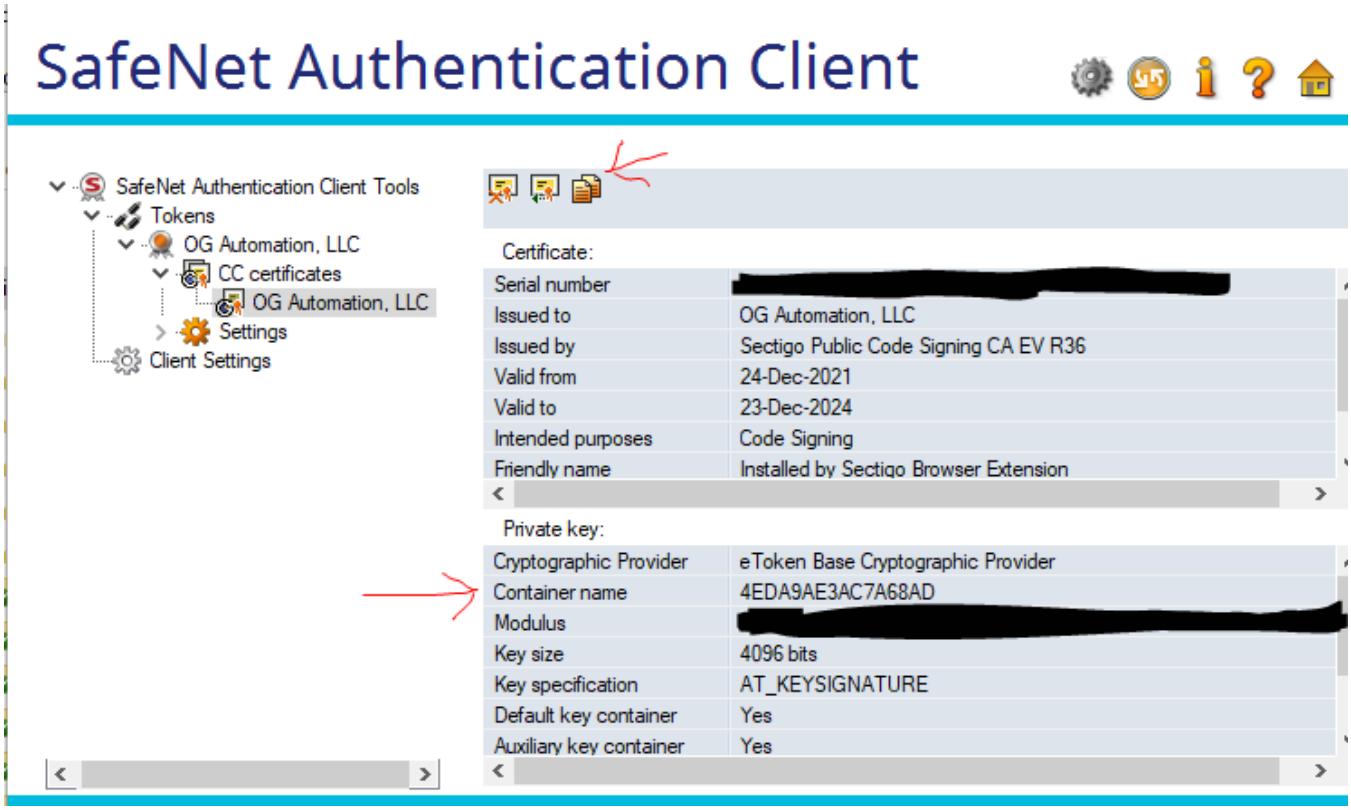
1. To get the Reader Name, switch to the Advanced view, and click on the token node (under the Tokens list).

   > NOTE: The Reader Name is only required if you have more than one reader connected. If you have only one token, you can skip this step.

   The Reader Name is listed as a parameter in the right-hand pane. It must be copied out, exactly as written.
   The best way to retrieve the Reader Name parameter is to use the Copy-To-Clipboard function on the toolbar:

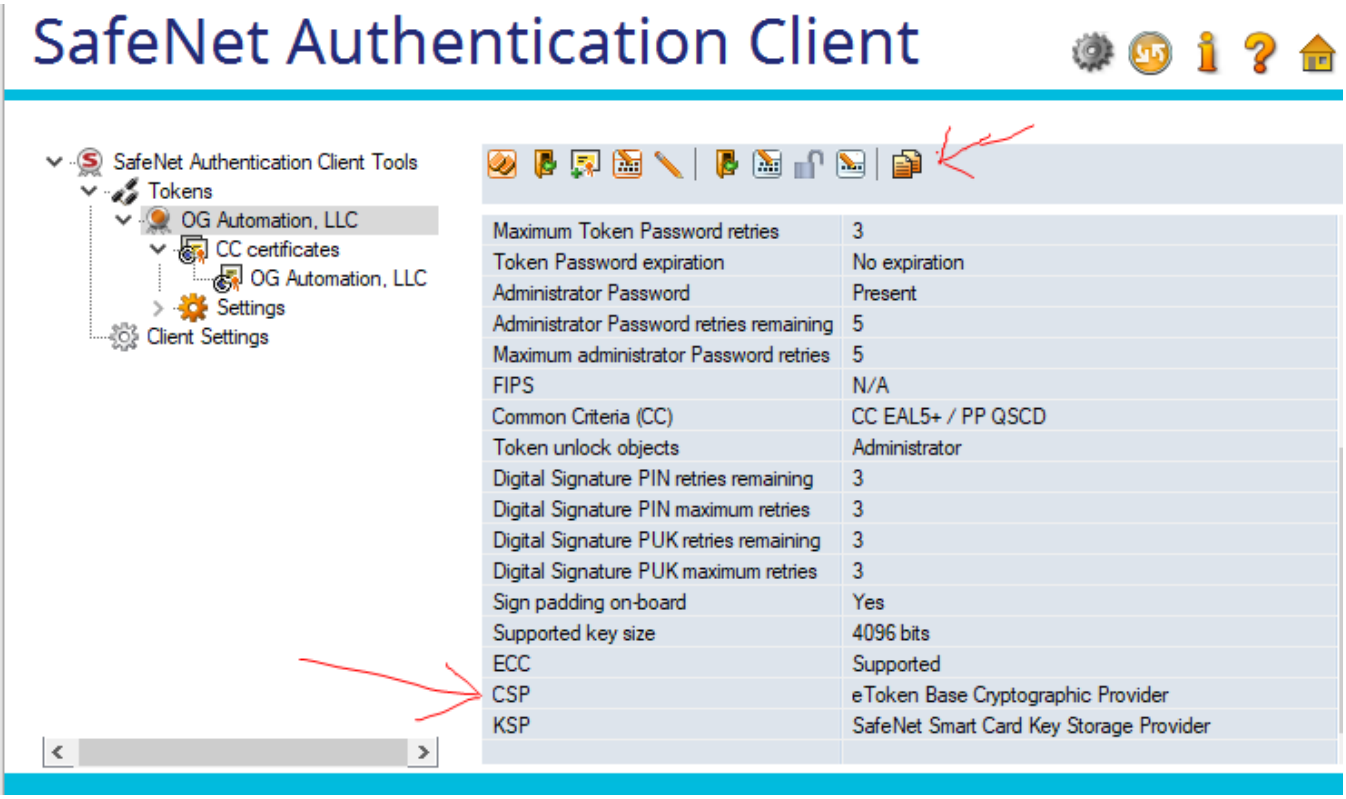


2. Then, paste the clipboard contents into notepad and recover the Reader Name, there.
3. To get the Container Name, stay in the Advanced view, and click on the individual certificate (under the Certificates node).
   The Container Name is a parameter of the private key in the right-hand pane. It must be copied out, exactly as written.
   It can be retrieved, same as the Reader Name, by using the Copy-To-Clipboard function on the toolbar:



4. Then, paste the clipboard contents into notepad and recover the Container Name, there.

5. To get the CSP (Cryptographic Service Provider) name, stay in the Advanced view, and click on the token node in the left-hand pane.
6. In the right-hand pane, scroll to the bottom of the token's parameter list, and locate the CSP parameter. It must be copied out exactly as written.
   It can be retrieved, same as the other parameters, by using the Copy-To-Clipboard function on the toolbar:



7. Then, paste the clipboard contents into notepad and recover the CSP name, there.
8. Once the CSP, Reader Name, and Container Name are retrieved, they need to be added to the build signing tool configuration, along with the token password, and the path to the signing certificate file.

Verify Signing

> Warning: A USB eToken will enter a lockdown state if too many failed password attempts have been made (requiring a certificate-vendor support to unlock).

And, the SafeNet Authentication Client software has no reliable means to quickly verify that a token password is correct.
The token must be used, to check the password.

So, any developed logic that orchestrates the build signing tool must include functionality to perform a single-step check of the token password. This will allow you to verify the token password works with a new token, along with verifying that other parameters (Reader and Container) are good to use.

1. With all the previous steps executed and data collected, we must perform a single-step signing to verify the token password, Reader Name, and Container Name are all correct.

Revision #1
Created 1 September 2025 03:34:50 by glwhite
Updated 1 September 2025 05:19:35 by glwhite