ClickOnce and Authenticode - Visual Studio (Windows)

By Mikejo5000

Archived: 2026-04-05 18:45:51 UTC

Authenticode is a Microsoft technology that uses industry-standard cryptography to sign application code with digital certificates that verify the authenticity of the application's publisher. By using Authenticode for application deployment, ClickOnce reduces the risk of a Trojan horse. A Trojan horse is a virus or other harmful program that a malicious third party misrepresents as a legitimate program coming from an established, trustworthy source. Signing ClickOnce deployments with a digital certificate is an optional step to verify that the assemblies and files are not tampered.

The following sections describe the different types of digital certificates used in Authenticode, how certificates are validated using Certificate Authorities (CAs), the role of time-stamping in certificates, and the methods of storage available for certificates.

Authenticode and code signing

A *digital certificate* is a file that contains a cryptographic public/private key pair, along with metadata describing the publisher to whom the certificate was issued and the agency that issued the certificate.

There are various types of Authenticode certificates. Each one is configured for different types of signing. For ClickOnce applications, you must have an Authenticode certificate that is valid for code signing. If you attempt to sign a ClickOnce application with another type of certificate, such as a digital e-mail certificate, it will not work. For more information, see Introduction to code signing.

You can obtain a certificate for code signing in one of three ways:

- Purchase one from a certificate vendor.
- Receive one from a group in your organization responsible for creating digital certificates.
- Generate your own certificate by using the New-SelfSignedCertificate PowerShell cmdlet, or by using *MakeCert.exe*, which is included with the Windows Software Development Kit (SDK).

A certificate generated using New-SelfSignedCertificate or the *MakeCert.exe* utility is commonly called a *self-cert* or a *test cert*. This kind of certificate works much the same way that a *.snk* file works in the .NET Framework. It consists solely of a public/private cryptographic key pair, and contains no verifiable information about the publisher. You can use self-certs to deploy ClickOnce applications with high trust on an intranet. However, when these applications run on a client computer, ClickOnce will identify them as coming from an Unknown Publisher. By default, ClickOnce applications signed with self-certs and deployed over the Internet cannot utilize Trusted Application Deployment.

By contrast, if you receive a certificate from a CA, such as a certificate vendor, or a department within your enterprise, the certificate offers more security for your users. It not only identifies the publisher of the signed software, but it verifies that identity by checking with the CA that signed it. If the CA is not the root authority, Authenticode will also "chain" back to the root authority to verify that the CA is authorized to issue certificates. For greater security, you should use a certificate issued by a CA whenever possible.

For more information about generating self-certs, see New-SelfSignedCertificate or MakeCert.

Timestamps

The certificates used to sign ClickOnce applications expire after a certain length of time, typically twelve months. In order to remove the need to constantly re-sign applications with new certificates, ClickOnce supports timestamp. When an application is signed with a timestamp, its certificate will continue to be accepted even after expiration, provided the timestamp is valid. This allows ClickOnce applications with expired certificates, but valid timestamps, to download and run. It also allows installed applications with expired certificates to continue to download and install updates.

To include a timestamp in an application server, a timestamp server must be available. For information about how to select a timestamp server, see How to: Sign Application and Deployment Manifests.

Update expired certificates

In earlier versions of the .NET Framework, updating an application whose certificate had expired could cause that application to stop functioning. To resolve this problem, use one of the following methods:

- Update the .NET Framework version 3.5 or later.
- Uninstall the application, and reinstall a new version with a valid certificate.

Store certificates

- You can store certificates as a *.pfx* file on your file system, or you can store them inside of a key container. A user on a Windows domain can have a number of key containers. By default, *MakeCert.exe* will store certificates in your personal key container, unless you specify that it should save it to a *.pfx* instead. *Mage.exe* and *MageUI.exe*, the Windows SDK tools for creating ClickOnce deployments, enable you to use certificates stored in either fashion.

Related content

- ClickOnce security and deployment
- Secure ClickOnce applications
- Trusted application deployment overview
- Mage.exe (Manifest Generation and Editing Tool)
