

dotNet Protector 5 Quick Start Guide

File Menu

New

Create a new project file

Open

Open an existing project file

Save Project

Save the current project

Save Project As

Saves a copy of the current project

Tools Menu

Build project

Launch protection with the current project parameters.

Activate dotNet Protector

Launch dotNet Protector activation wizard. After installation, dotNet Protector works in demonstration mode (all generated assemblies will have a 5 days execution limit).

If you acquired a licence for dotNet Protector, you will have to activate it (see the Activation section).

The product key you will enter during activation will determine available features (dotNet Protector Edition : Classic, Professional or Advanced).

Re-check License

If your copy of dotNet Protector is activated, but the title bar indicates (locked), this means that dotNet Protector does not recognize any more the hardware associated with your license. This option makes it possible to start again the verification.

Import /Export License File

Allows to save / restore your licence file. If you must reinstall your computer, save your licence file before un-installation; you will then be able to re-import and avoid activating once again. This option will work only if the material on which the license is imported is the same one as that on which it was exported.

This option is also useful if you chose the USB Key activation. You can then export to import your license file after activation, and import it on all the computers where you wish to run dotNet Protector (dotNet Protector will run only if the activated key USB is present).

This option is valid also for the 'instant activation' mode. In this mode, a license token is requested from each run of dotNet Protector. This token is associated with the computer that requested it. Computer-token association is deleted every 24h, which enables you to change computer each day. You must have access to Internet to run dotNet Protector in this mode.

De-Activate dotNet Protector

Remove the license file and erases the hardware hash on our activation servers. This option will enable you to activate another computer (hardware change). A de-activation key will be required; this key is not your product key. If you did not receive the de-activation key during activation or if you lost it, your license file will be removed, but hardware association will remain on our activation servers; you will not be able to activate another computer. If this case were to arrive, send an email to dotnetprotector@pvlog.com by specifying your license number; we will then do the de-activation manually.

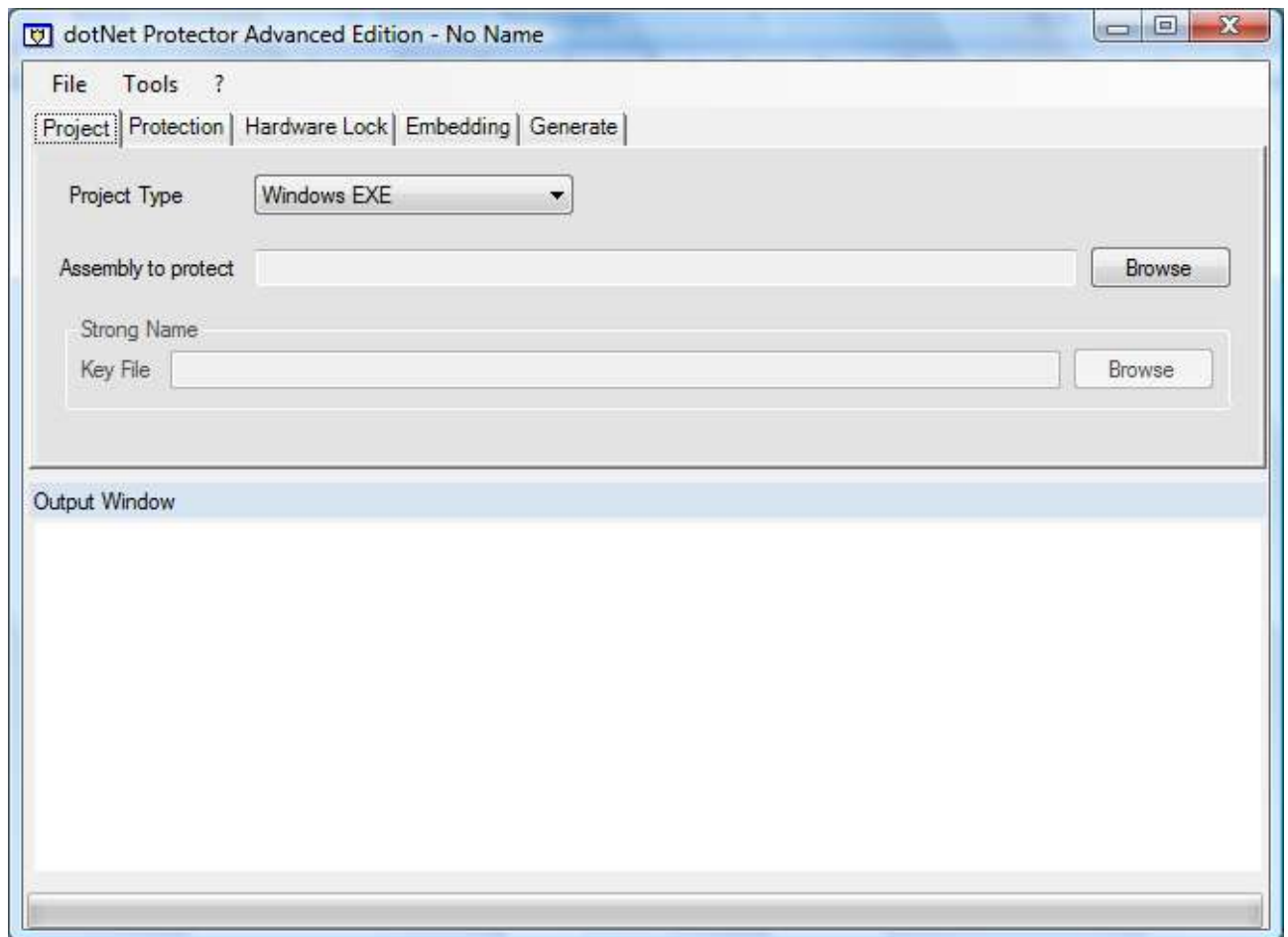
Import Keyset

dotNet Protector activation process uses public/private key pairs and a symmetrical key. The key set contains all these keys. If you do not use the activation features (hardware lock) the key set is not fundamental. If on the contrary you activate this functionality, you must make sure that your program is protected with the same key set as your tool that delivers the licenses. The key set determines the encoding of product keys, configuration strings and license keys. It guarantees to you that one will not be able to generate product keys licenses, unless one has your private keys. Of course, the protected program only embeds the public keys. A key set is generated during the first dotNet Protector run. The generation of this key set relies on a cryptographic random generator to have the best uniqueness guarantee.

Export Keyset

Allows you to save your key set.

Project Tab



Project Type

Allows choosing the type of assembly to protect (Windows EXE, Windows DLL - or ASP.Net, SQL 2005)

Assembly to protected

Chose your assembly using the Browse button

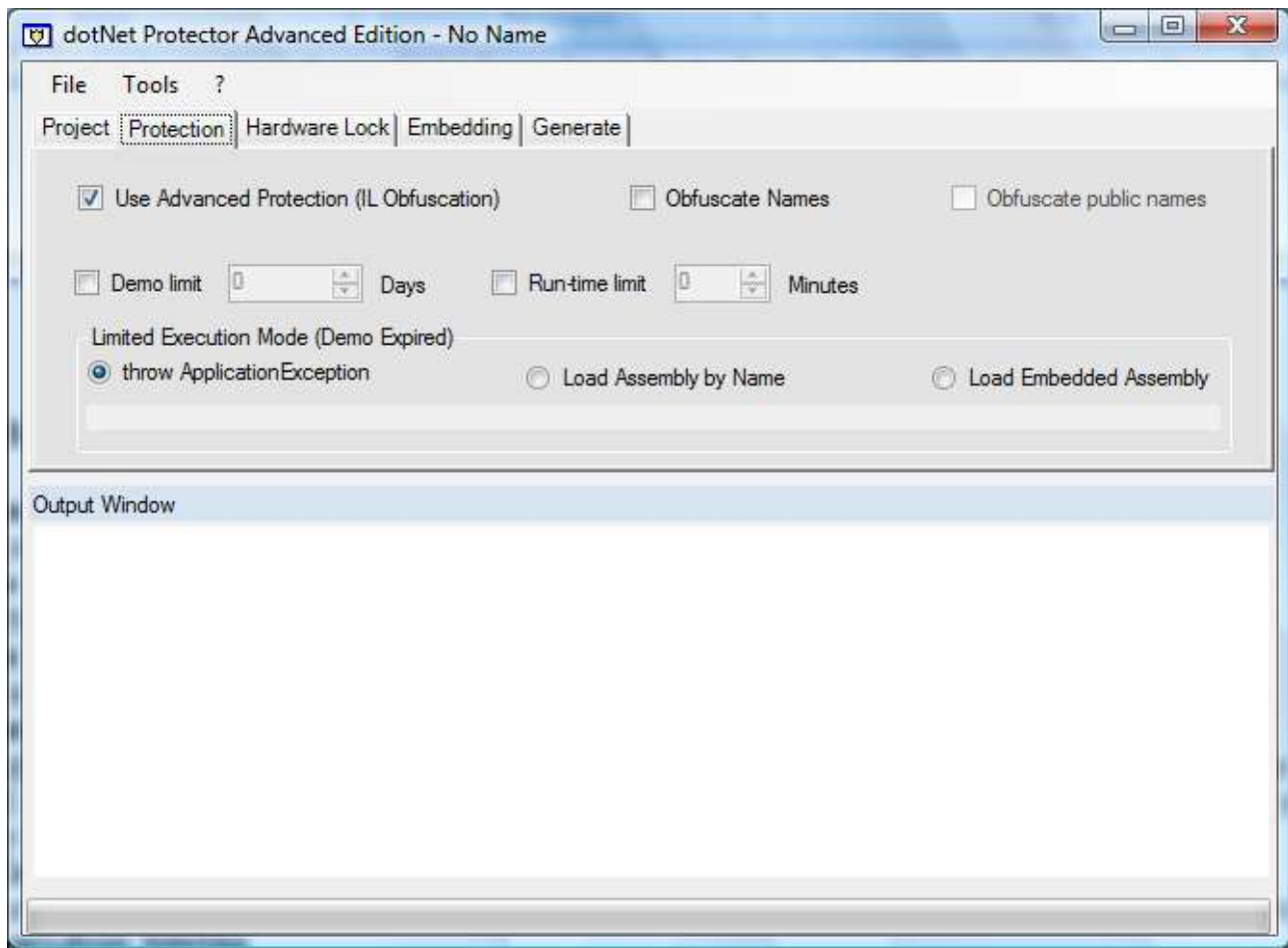
Strong name

If your assembly is signed, you must select the snk file to resign it after protection

Output window

Protection progress will show up in this window

Protection Tab



Use advanced protection (IL obfuscation)

Activate protection by methods encryption

Obfuscate names

Activate the 'traditional' obfuscation

Obfuscate public names

For Exe only, allows to obfuscate all the names.

Demo Limit

Sets the execution limit of your program (demonstration). The limit starts the date you generate the protected assembly and not it's installation date. Note: when dotNet Protector runs in demo mode, your assembly will be limited to 5 days anyway.

Run-time Limit

Sets the run-time limit of your program (Exe only). The process will be killed when this period expires.

Limited execution Mode

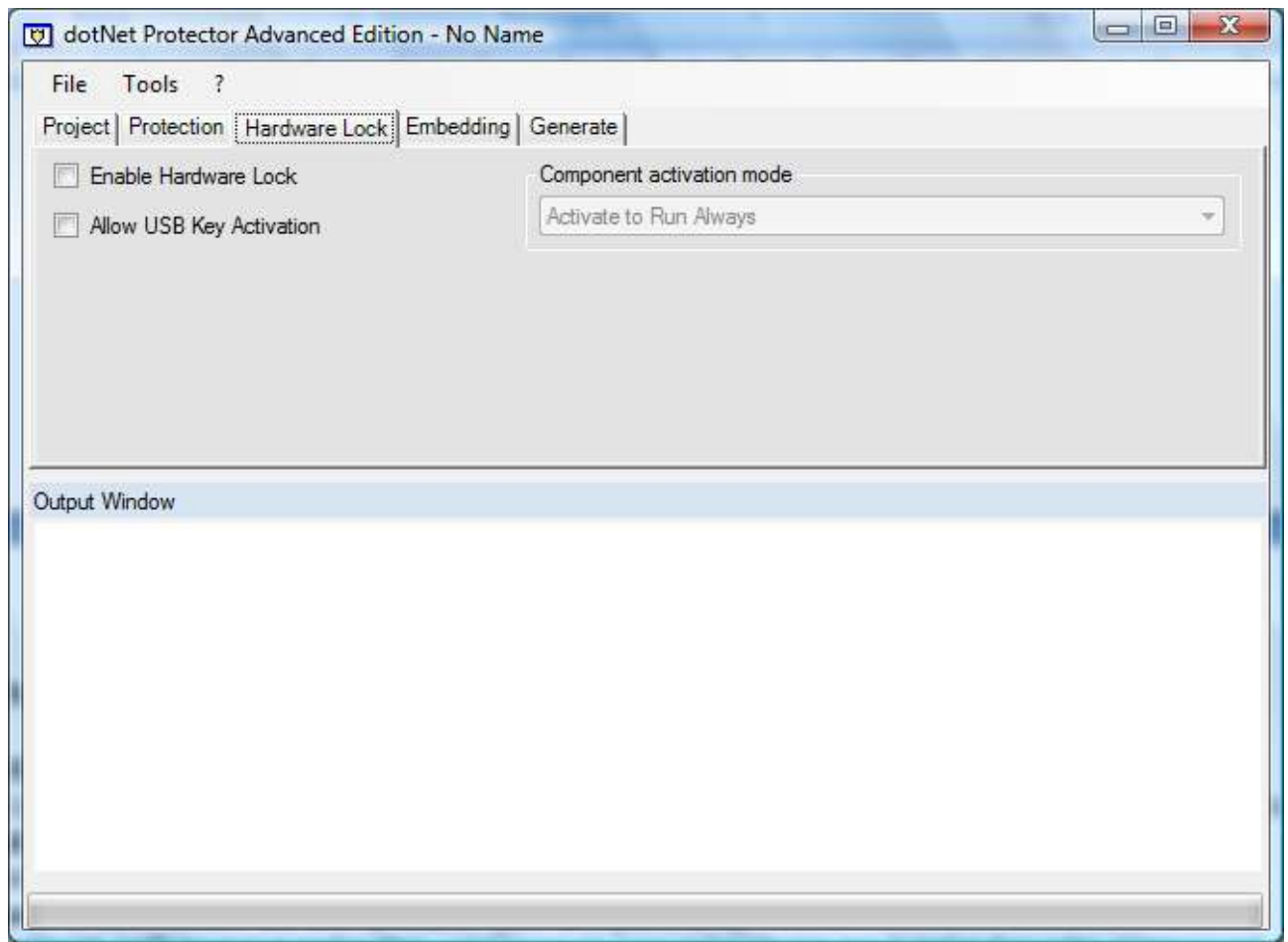
Sets what will happen when the demo period expires (if no license can be granted)

Throw ApplicationException (default) : an ApplicationException will be thrown, killing the process.

Load Assembly by name : this allows you to tell an alternate assembly to load. You'll be invited to browse an assembly to run, then its full name will be recorded in the protected assembly. It is highly recommended to select a strong name assembly.

Load embedded Assembly : you'll have to browse for the alternate assembly to run; this assembly will be embedded in the final executable.

Hardware Lock Tab



Enable Hardware Lock

If this box is checked, you'll have to generate a license to run your program (Activation)

Allow USB Activation

Allows activation by using a USB storage key instead of a computer hardware.

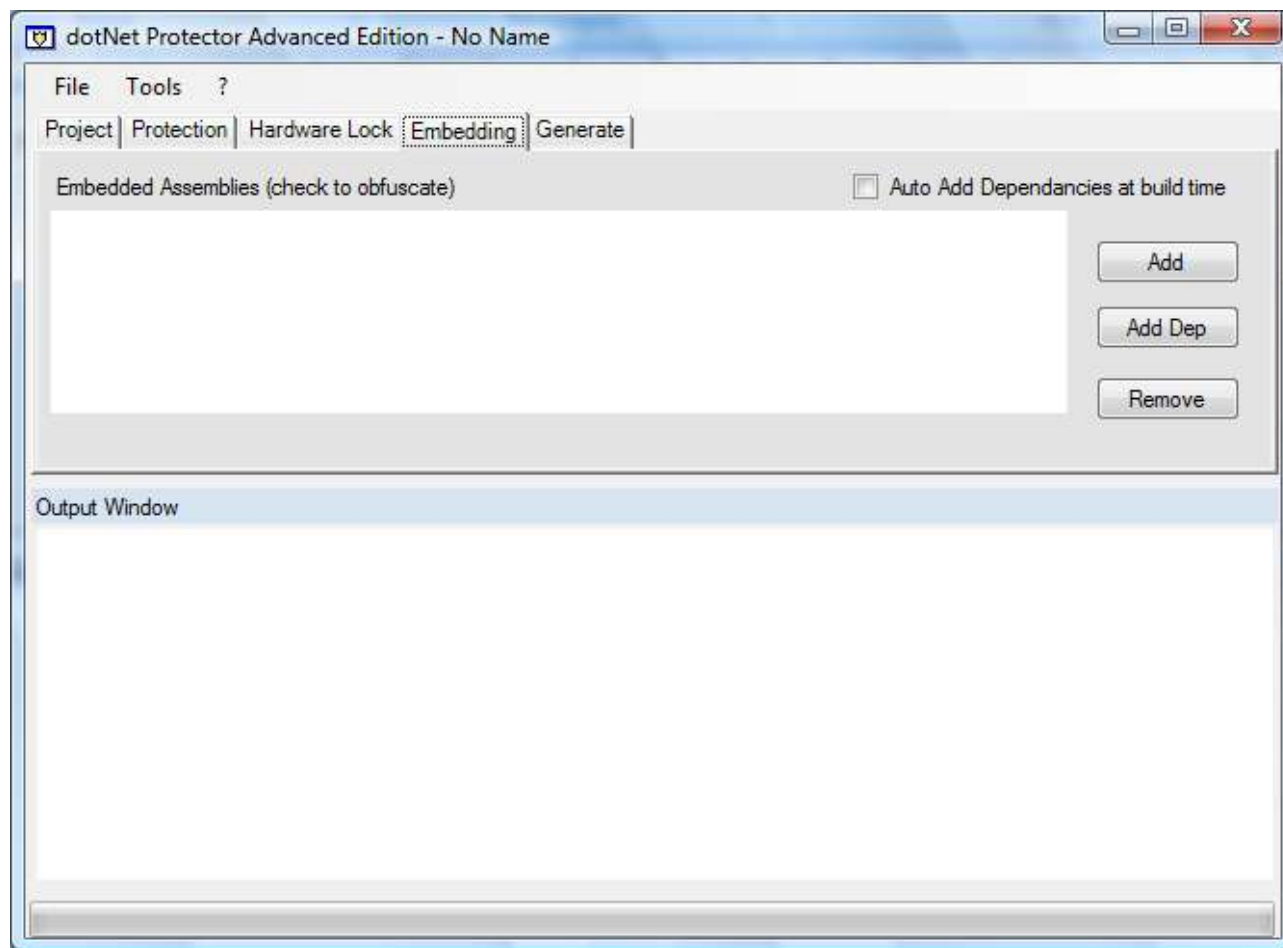
Component activation mode (Dll only)

Activate to run always : a license is required to run your component.

Activate to reference : a license is required to reference your component in another program. No license is required at runtime.

Activate to reference and run ASP.Net : a license is required to reference your component in a Windows application. A runtime license is required to run in ASP.Net

Embedding Tab (EXE only)



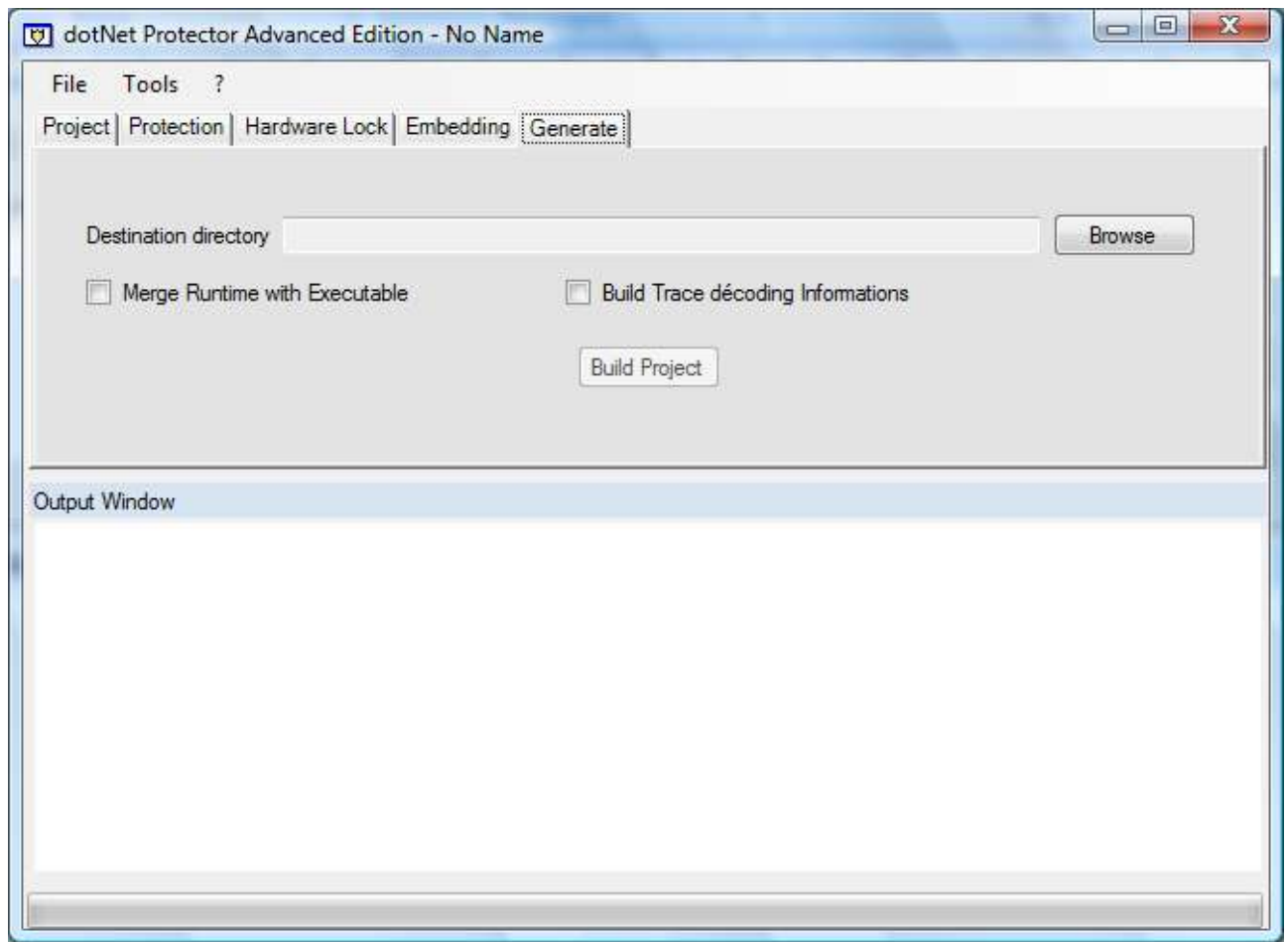
Add : allows to add an assembly (a referenced Dll) into the final executable.

Add Dep : adds all referenced assemblies that are present on the same location as the main assembly.

Remove : removes an embedded Dll from the list.

Auto add dependencies at build time : when dotNet Protector builds the project, it looks for referenced assemblies and adds all of them that are present in the same location as the main assembly. This option gives you no control on which assemblies are embedded or not. It is better to add referenced assembly in the project than to check this box.

Generate Tab



Destination directory

Let you select the output location. If you decide to embed dotNet Protector's runtime (EXE project) additional directory will be created to separate different processor architectures (x86, amd64, Itanium).

Merge runtime with executable

Lets you build an EXE without dotNet Protector's runtime dependency. The generated exe is then a mixed (native + managed) exe embedding the runtime. If your program is a v1.1 assembly, it will be converted to v2 to embed the runtime, which can cause som compatibility issues if your assembly can't run on the v2. When you select this option, your assembly is converted to an embedded netmodule. The final exe is then a multi-modules mixed assembly. Don't forget that `Assembly.GetExecutingAssembly.ManifestModule` will give you dotNet Protector's runtime module. Your assembly's properties (name, version, assembly custom attributes) will be transferred to the generated assembly, as well as manifest resources. Then, `GetManifestResourceStream` should have the desired result.

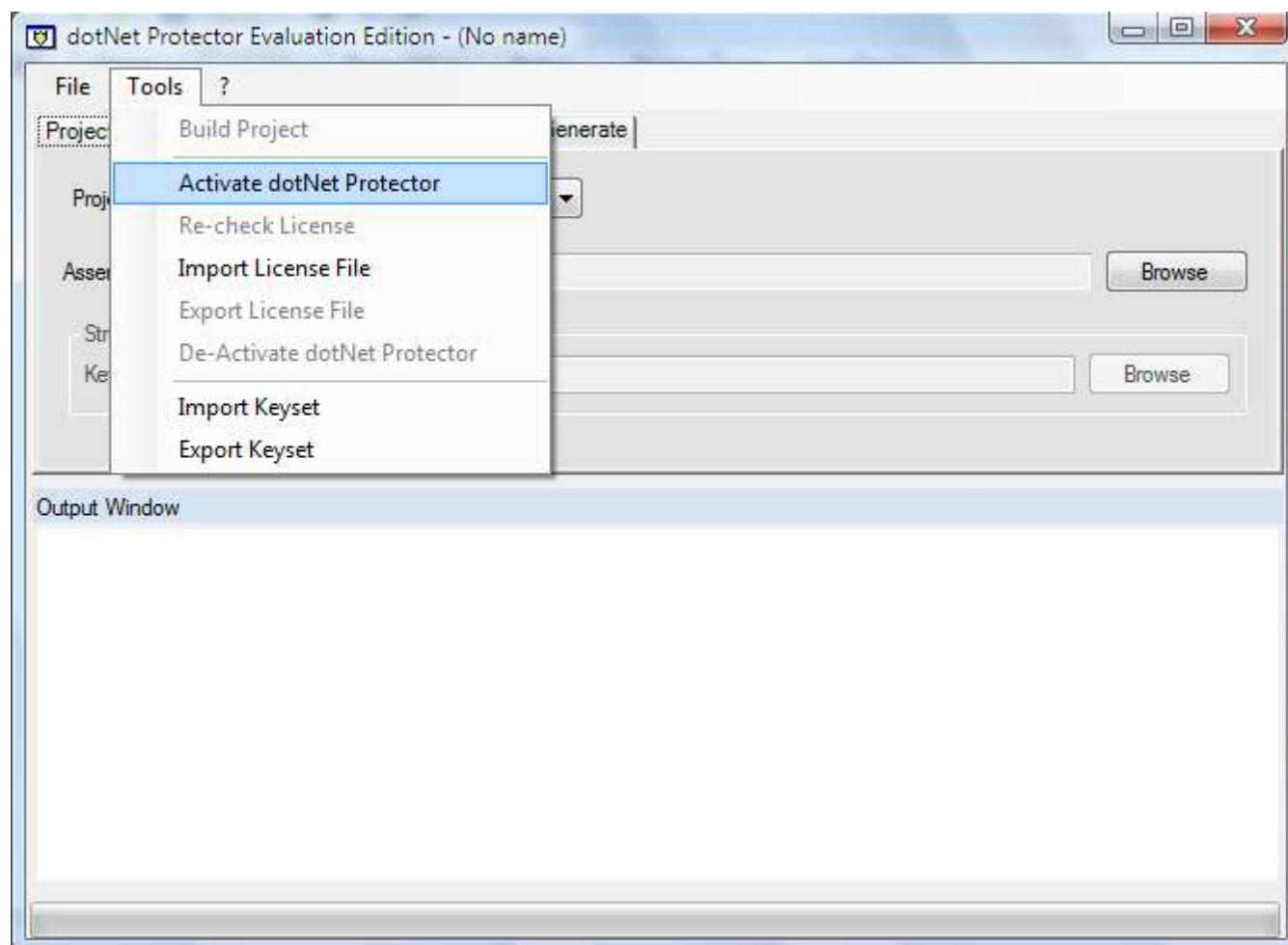
Build Trace decoding informations

dotNet Protector strips all debug informations before protecting assembly. Original pdb files would be incompatible with the protected assembly anyway. If pdb are available in the same location of an assembly to protect, dotNet Protector can build a file allowing obfuscated StackTrace decoding.

Build Project button

Runs protection with current project parameters.

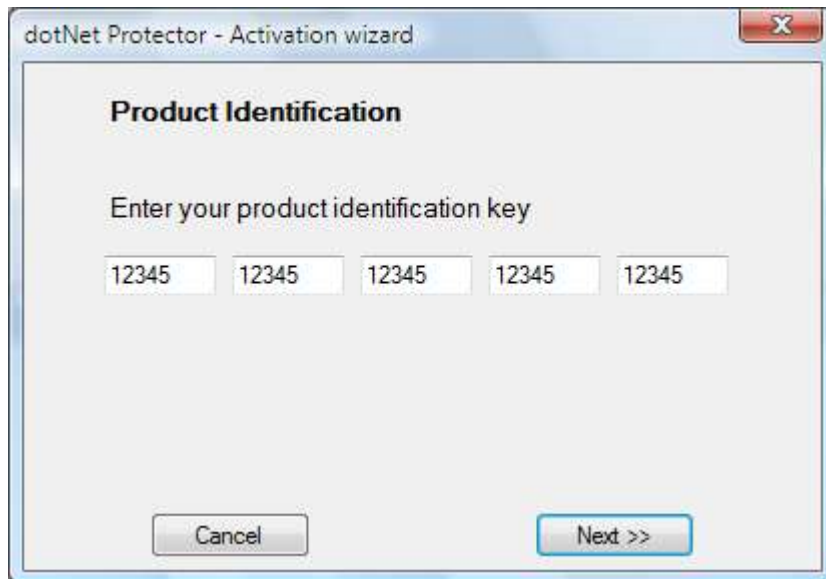
Activating dotNet Protector



Enter one of the provided product keys. You need one for each computer/os pair.

If you install both dotNet Protector x86 and dotNet Protector amd64 on Vista x64, you can use the same key for the 2 products. In this case, just activate the first one, the export your license file and import it for the

second install; you'll save time in not entering the key and activating again.



dotNet Protector - Activation wizard

Product Identification

Enter your product identification key

12345 12345 12345 12345 12345

Cancel Next >>

This screenshot shows the 'Product Identification' step of the 'dotNet Protector - Activation wizard'. The window has a title bar with the text 'dotNet Protector - Activation wizard' and a close button. The main content area is titled 'Product Identification' and contains the instruction 'Enter your product identification key'. Below this instruction, there are five text input fields, each containing the number '12345'. At the bottom of the window, there are two buttons: 'Cancel' on the left and 'Next >>' on the right.



dotNet Protector - Activation wizard

Product Identification

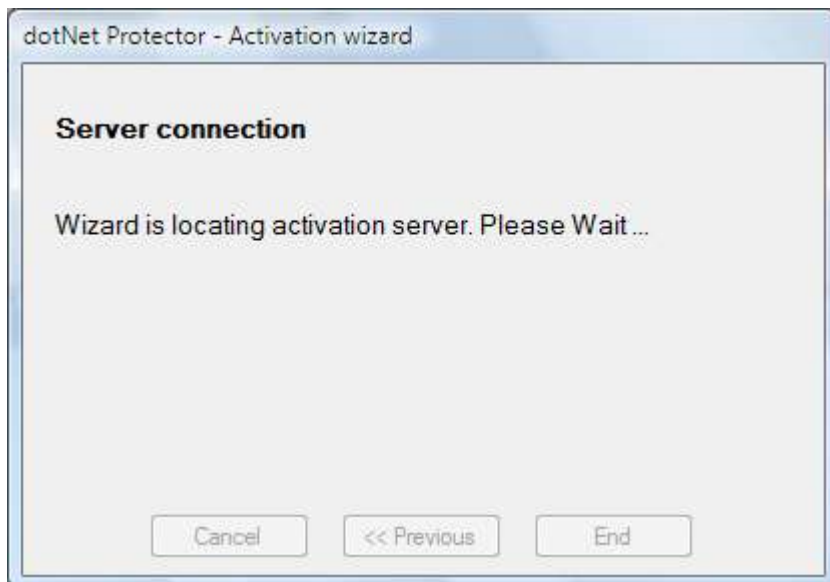
Enter your product identification key

XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX

dotNet Protector Advanced
License Number : XXXXXXXXXX

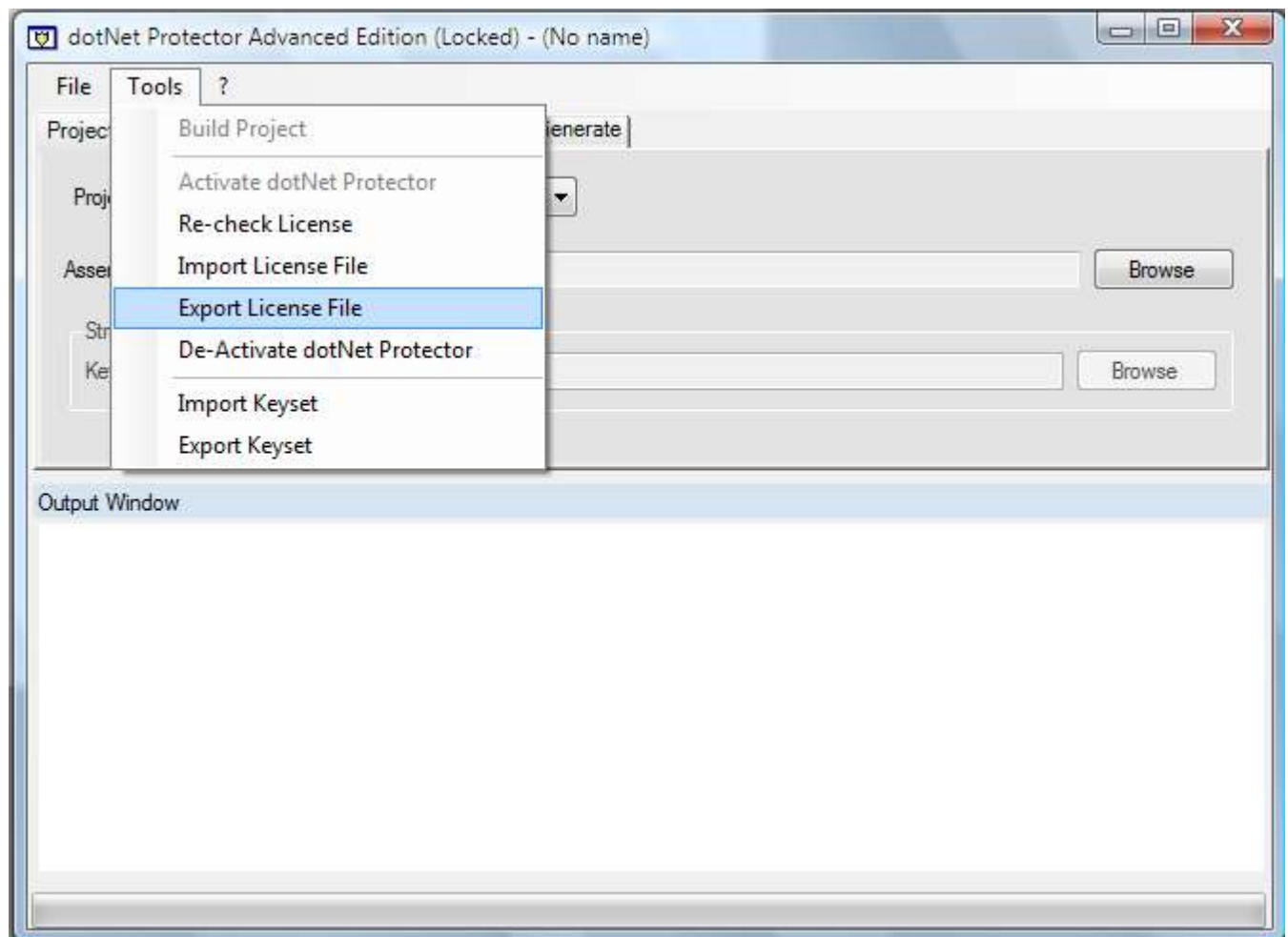
Cancel Next >>

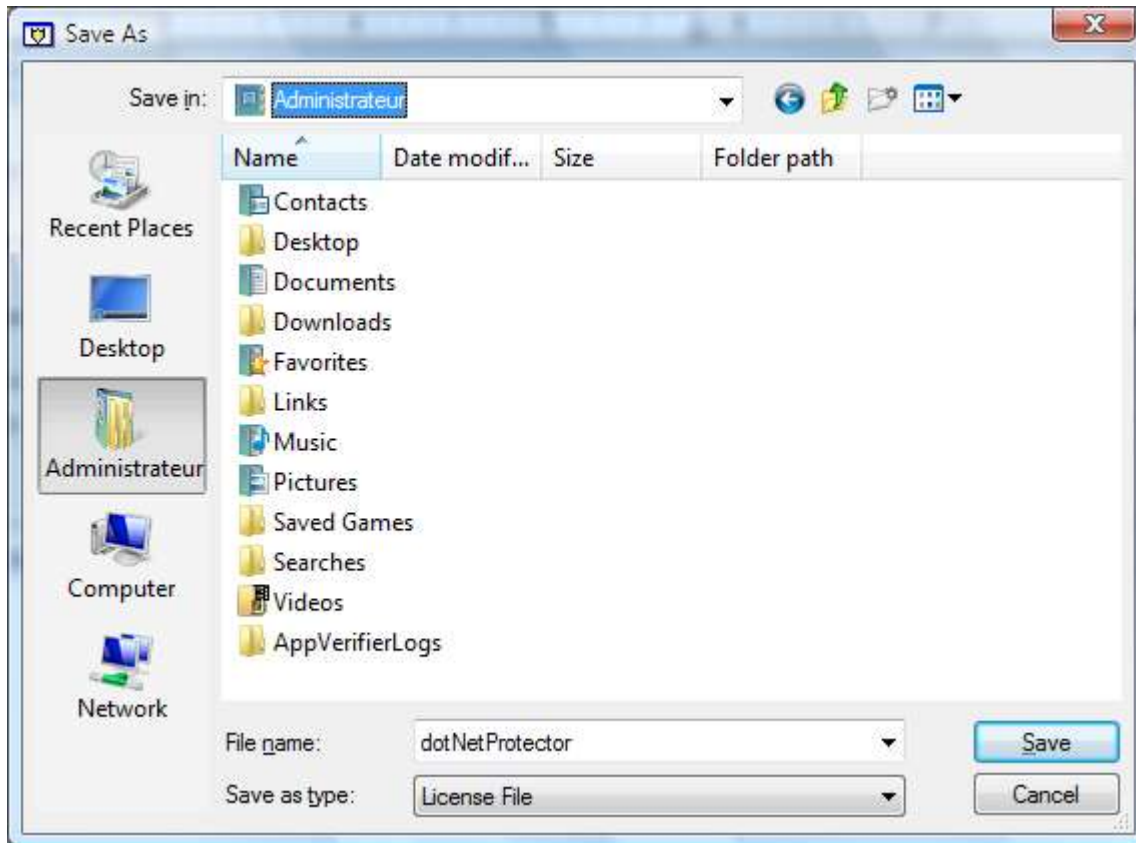
This screenshot shows the 'Product Identification' step of the 'dotNet Protector - Activation wizard'. The window has a title bar with the text 'dotNet Protector - Activation wizard' and a close button. The main content area is titled 'Product Identification' and contains the instruction 'Enter your product identification key'. Below this instruction, there are five text input fields, each containing a series of 'X' characters, indicating that the product identification key has been masked. At the bottom of the window, there are two buttons: 'Cancel' on the left and 'Next >>' on the right.



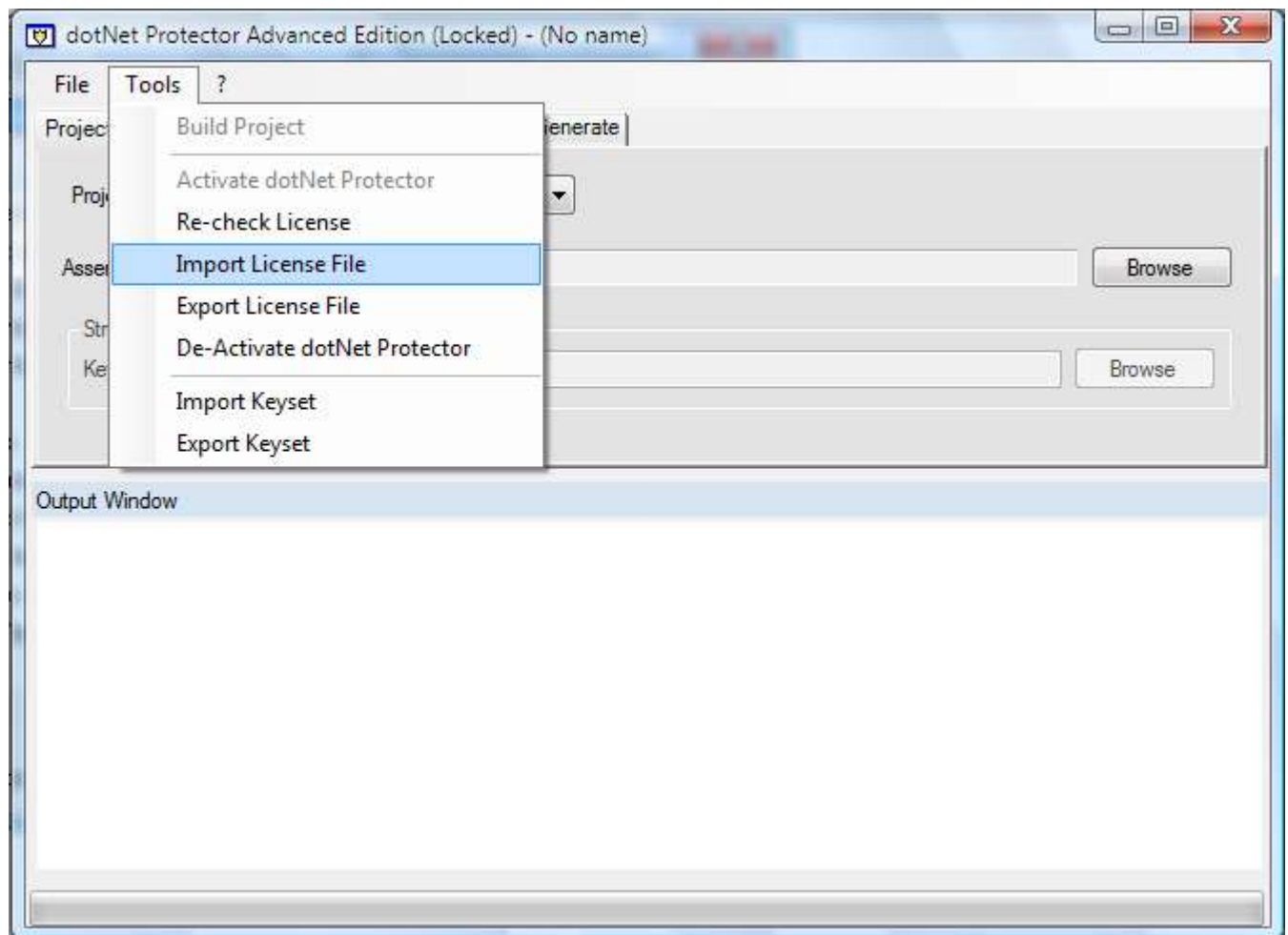
Once you have a successful activation from a server, click End and export your license file to save time the next time you install dotNet Protector

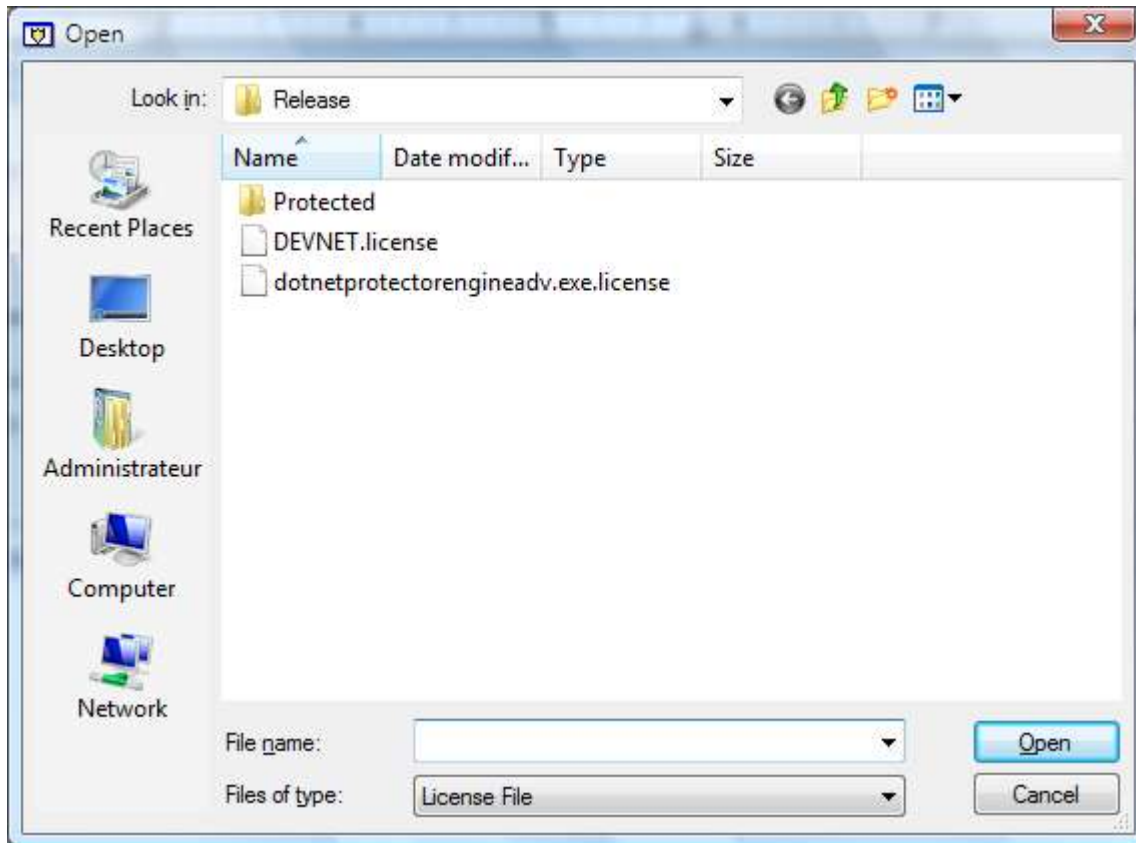
Exporting a License File





Importing a License File





Importing a Key Set

