# dotNet Protector
# Guide de démarrage rapide

## Menu Fichier

### Nouveau
Crée un nouveau fichier de projet

### Ouvrir
Ouvre un fichier de projet existant

### Enregistrer le Projet
Sauvegarde le projet en cours

### Enregistrer le Projet Sous
Sauvegarde une copie du projet en cours

## Menu Outils

### Générer le projet
Lance la protection avec les paramètres de projet en cours.

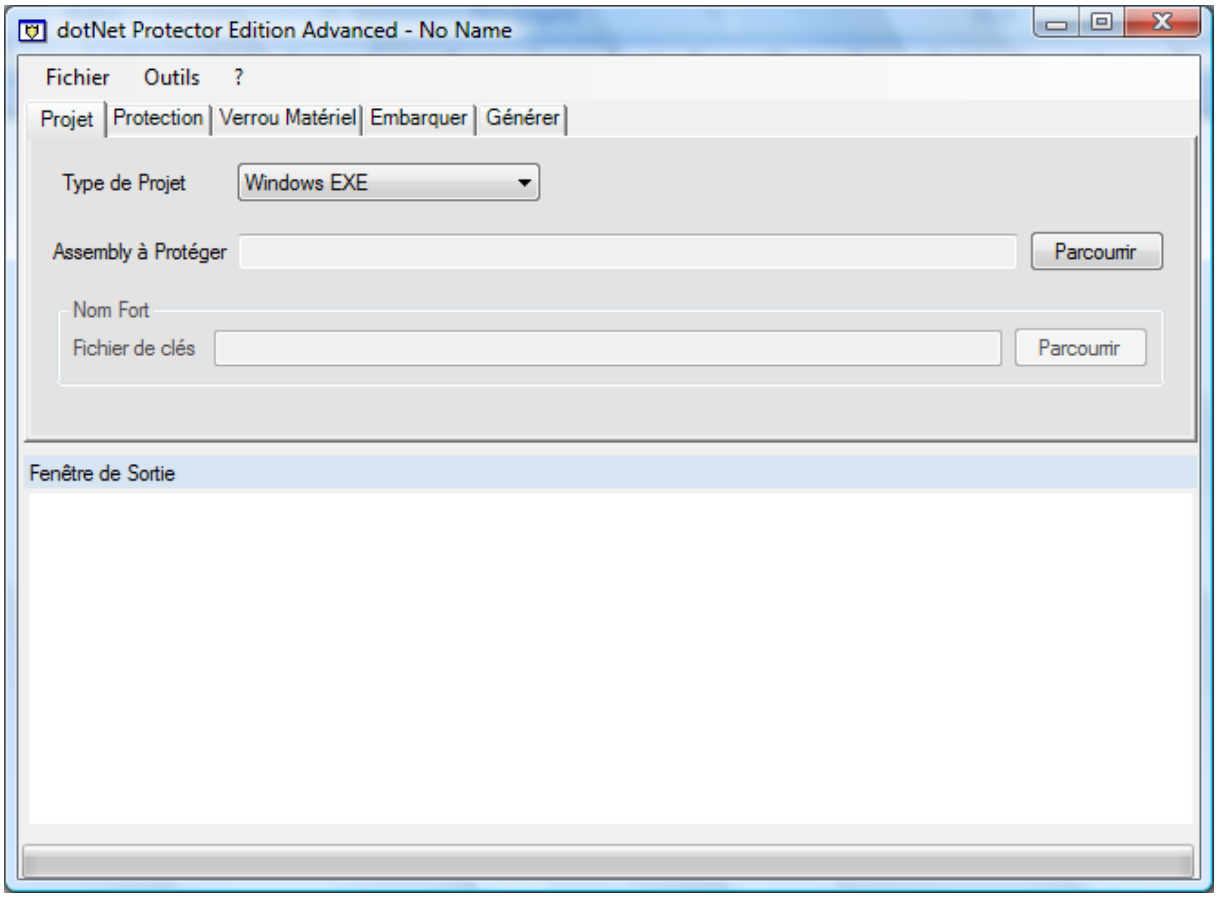
### Activer dotNet Protector
Lance l'assistant d'activation. Après l'installation, dotNet Protector s'exécute en mode démo (tous les assemblies générés auront une limite d'exécution de 5 jours).
Si vous avez acquis une licence de dotNet Protector, vous devrez l'activer (voir la section Activation).
La clé produit que vous allez entrer lors de l'activation déterminera les fonctionnalités disponibles (Edition de dotNet Protector : Classic, Professional ou Advanced).

### Importer in fichier de licence
Permet de copier un fichier de licence sauvegardé. Si vous devez réinstaller votre ordinateur, sauvegardez votre fichier de licence avant la désinstallation ; vous pourrez alors le ré-importer et éviter d'activer une nouvelle fois. Cet option ne fonctionnera que si le matériel sur lequel la licence est importé est le même que celui sur lequel elle a été exportée.
Cette option est aussi utile si vous avez opté pour l'activation d'une clé USB. Vous pouvez alors exporter votre fichier de licence après l'activation, et l'importer sur tous les ordinateurs où vous désirez exécuter dotNet Protector (dotNet Protector ne s'exécutera que si la clé USB activée est présente).
Cette option est valable également pour le mode d'activation 'instantanée'. Dans ce mode, un jeton de licence est demandé à chaque exécution de dotNet Protector. Ce jeton est associé à l'ordinateur qui l'a demandé. L'association ordinateur-jeton est effacée toutes les 24h, ce qui vous permet de changer d'ordinateur chaque jour. Vous devrez avoir accès à internet pour exécuter dotNet Protector.

### Importer un jeu de clés
Le processus d'activation de dotNet Protecteur utilise des paires de clés publique/privée et une clé symétrique. Le jeu de clés contient toutes ces clés. Si vous n'utilisez pas les fonctionnalités d'activation (verrou matériel) le jeu de clés n'est pas fondamental. Si au contraire vous activez cette fonctionnalité, vous devez vous assurer que votre programme est protégé avec le même jeu de clé que votre outil permettant de délivrer les licences. Le jeu de clé détermine l'encodage des clés produit, des chaînes de configuration et des clés de licence. Il vous garantit qu'on ne pourra pas générer des clés produit ou des licences, a moins de disposer de vos clés privés. Bien entendu, le programme protégé embarque les clés publiques uniquement. Un jeu de clé est généré lors de la première exécution de dotNet Protector. La génération de ce jeu repose sur un générateur aléatoire cryptographique pour avoir les meilleurs garanties d'unicité.

### Exporter un jeu de clés
Vous permet de sauvegarder vos clés.

## Onglet Projet



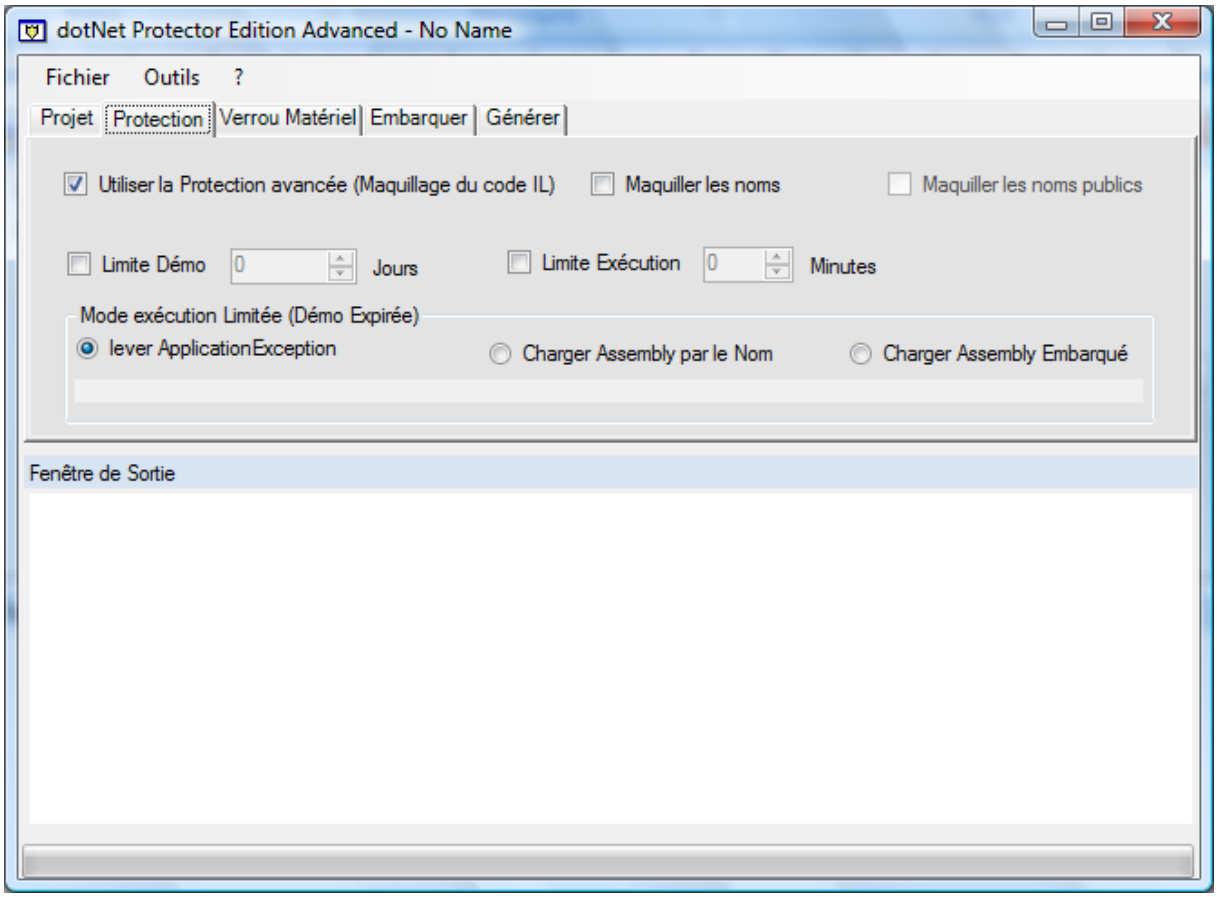

### Type de Projet
Permet de choisir le type d'assembly à protégér (Windows EXE, Windows DLL - ou ASP.Net, SQL 2005)

### Assembly à protéger
Chosissez votre assembly à l'aide du bouton Parcourrir

### Nom Fort
Si votre assembly est signé, vous devez selectionner le fichier snk pour le re-signer après protection

### Fenêtre de sortie
Affichera l'avancement de la protection

## Onglet Protection




### Utiliser la protection avancée (Maquillage du code IL)
Active la protection par cryptage des méthodes.

### Maquiller les noms
Active l'obfuscation 'traditionnelle'

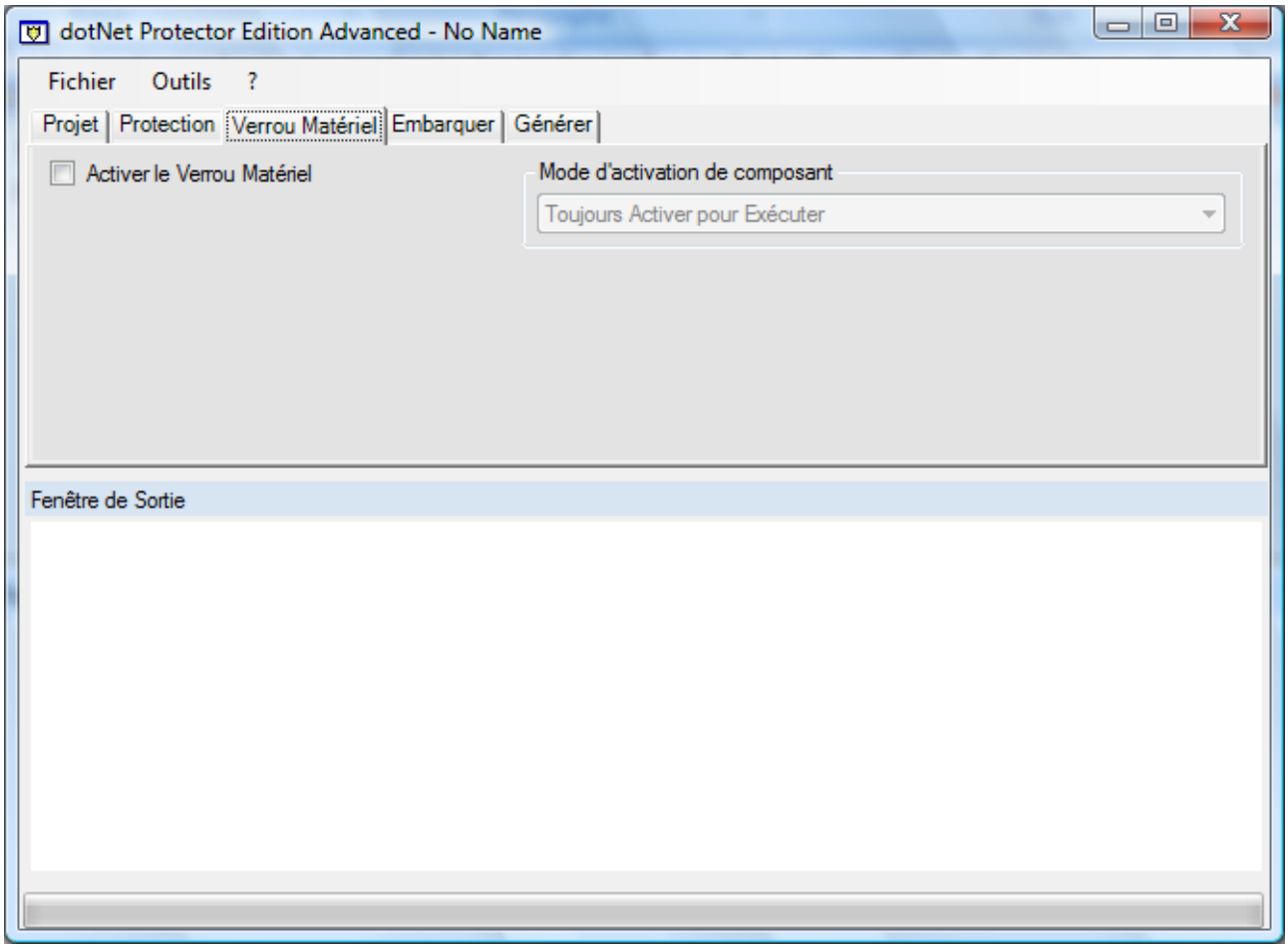
### Maquiller les noms publics
Pour un Exe uniquement, permet de maquiller tous les noms.

### Limite Démo
Permet de fixer la limite d'exécution de votre programme (démo). La limite est comptée à partir de la date de génération de l'exécutable protégé et pas la date d'installation. Note : avec la version démo de dotNet Protector, votre exécutable sera limité à 5 jours maximum.

### Limite Exécution
Fixe la durée d'exécution du processus (Exe uniquement). Le processus sera tué au terme de cette période.

### Mode exécution limitée
Fixe le comportement du programme après expiration de la période de démo (si aucune licence valide ne peut être trouvée)
**Lever ApplicationException (defaut)** : une exception ApplicationException sera levée, mettant fin au processus.
**Charger Assembly par le nom** : permet d'indiquer un nom d'assembly à charger. dotNet Protector vous invitera à rechercher l'assembly à exécuter, puis enregistrera son nom complet. Il est recommander d'utiliser un assembly de nom fort.
**Charger Assembly embarqué** : dotNet Protector vous invitera à rechercher l'assembly à exécuter ; cet assembly sera alors incorporé à l'assembly final.

## Onglet Verrou Matériel



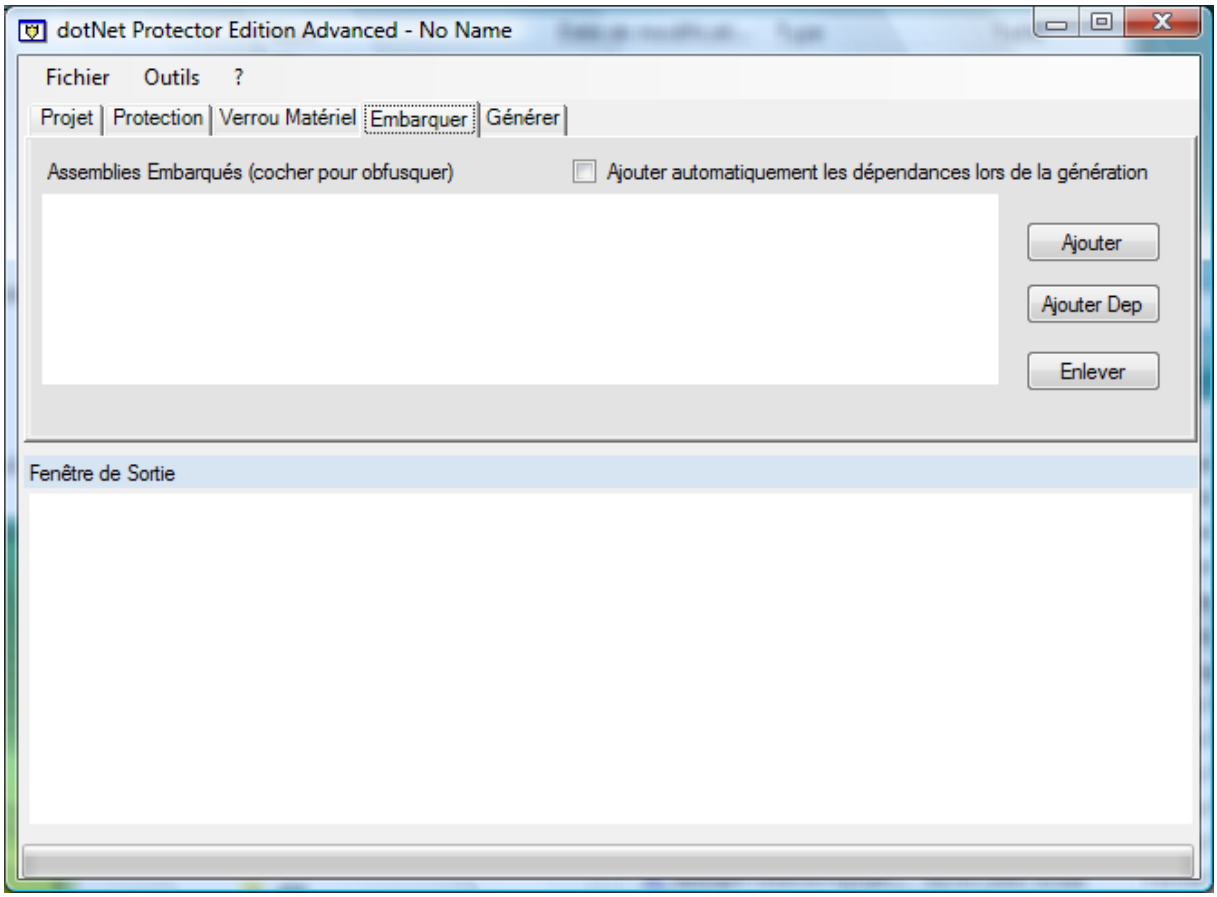

### Activer le verrou matériel

Si cette case est cochée, vous devrez générer une licence pour exécuter votre programme (Activation)

### Mode d'activation de composant (Dll uniquement)

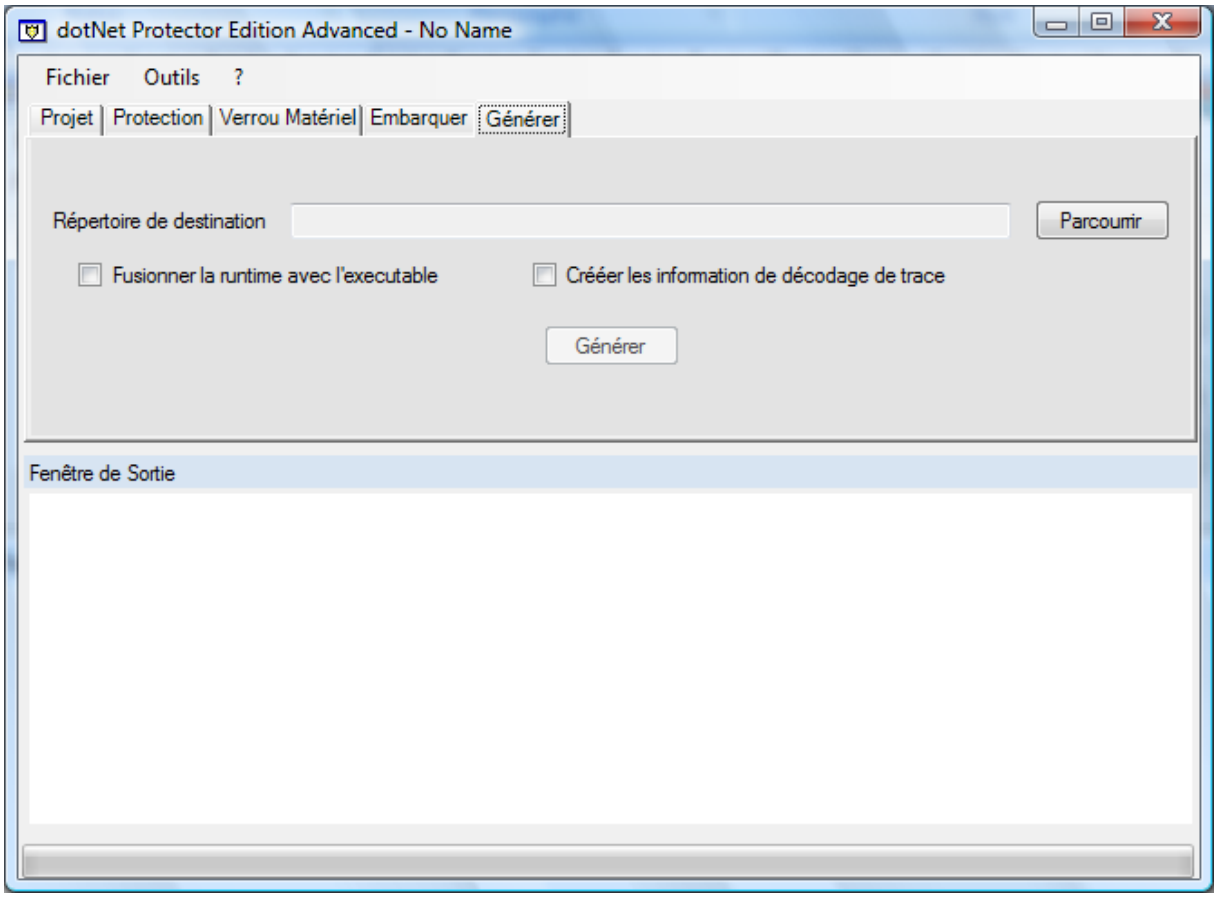
**Toujours activer pour exécuter** : une licence sera nécessaire pour exécuter votre composant
**Activer pour référencer** : une licence sera nécessaire pour consommer votre composant dans un autre programme, aucune licence ne sera nécessaire à l'exécution.
**Activer pour référencer et exécuter ASP.Net** : une licence sera nécessaire pour consommer votre composant dans une application Windows. Une license sera également nécessaire pour exécuter votre composant dans ASP.Net

## Onglet Embarque (EXE seulement)




**Ajouter :** permet d'ajouter un assembly (une dll référencée) à l'exécutable final.
**Ajouter Dep :** permet d'ajouter les Dlls référencées
**Enlever :** permet désactiver l'incorporation d'une Dll.
**Ajouter automatiquement les dépendances lors de la génération :** au moment de la génération de l'assembly protégé, dotNet Protector cherche les dépendances et les inclue à l'exécutable. Cette fonction ne vous donne aucun contrôle sur les assemblies qui pourront être embarqués ; il est recommandé d'ajouter les références au projet plutôt que de cocher cette case.

## Onglet Générer




### Répertoire de destination

Permet de choisir l'emplacement de la sortie. Si vous choisissez d'embarquer la runtime dotNet Protector (projet EXE), des répertoires seront créés pour séparer les architectures pour lesquelles un assembly est générer (x86, amd64, Itanium).

### Fusionner la runtime avec l'exécutable

Vous permet de générer un Exe sans qui ne dépend pas de la runtime dotNet Protector. L'exe généré est alors un exe Mixte (code natif + code managé) contenant la runtime. Si l'exe à protéger a été compilé en v1.1, il sera converti en v2 pour embarquer la runtime, ce qui peut poser des problèmes de compatibilité avec certains assemblies v1. Si vous utilisez cette option, votre assembly sera converti en netmodule embarqué. L'exécutable généré sera donc un assembly mixte multi-modules rendant votre code complètement caché pour les outils conventionnels de désassemblage. N'oubliez pas que dans ce cas Assembly.GetExecutingAssembly.ManifestModule vous donnera le module de la runtime dotNet Protector.
Les propriétés de votre assembly (nom, version, attibuts au niveau de l'assembly) seront transférés à l'assembly généré, ainsi que les ressources de manifeste. Ainsi GetManifestResourceStream aura l'effet désiré.

### Créer les informations de décodage de trace

dotNet Protector supprime les informations de débogage présent dans les assemblies avant de les protéger. Les fichiers ne serait de toute façon pas compatibles avec l'assembly protégé. Si les fichiers pdb sont présents lors de la protection, dotNet Protector peut générer un fichier permettant de décoder les StackTrace obfusquées.

### Bouton Générer

Lance la protection avec les paramètres de projet en cours