

dotNet Protector 5®

Générer un projet de déploiement Visual Studio avec des assemblages protégés

1. Exe Protégés avec la runtime embarquée

Si la runtime dotNet Protector est embarquée dans le projet, VS2005 ne peut pas générer un projet de deployment (il ne trouve pas embedded.netmodule).

Vous pouvez ignorer l'avertissement concernant la dépendance manquante PvLogiciels.dotNetProtector.Runtime (cet assembly est aussi embarqué dans votre exe).

Solution :

Générer le vdproj avec un netmodule bidon et enlever le netmodule du msi apres generation.

Contenu de EmbeddedSetupPatch.zip

embedded.netmodule : le netmodule bidon utilisé par PV Logiciels pour compiler la runtime embarquée de dotNet Protector®.

MsiRemoveEmbeddedModule.exe : un utilitaire en ligne de commandes pour supprimer embedded.netmodule du msi généré.

Executables générés par dotNet Protector lorsque la fonctionnalité 'embarquer la runtime' est activée

Starting with the v2 of .NET, assemblies can run in 64 bit mode on 64 bit platforms.

A partir de la v2 de .Net, les assemblages peuvent s'exécuter en mode 64 bit sur les plateformes 64 bit.

Si votre assembly est compilé avec l'option 'cpu x86', il fonctionnera dans un mode x86 (émulé sous windows 64 bits) sur chaque plateforme. Dans ce cas, dotNet Protecteur génère un seul exe.

Si votre assembly est compile 'Any cpu', la valeur par défaut pour les projets pure-IL (C#, VB.NET), il sera converti en executable 64 bit lors de l'exécution sous windows 64 bits. Dans ce cas, dotNet Protector génère 3 exécutables, un pour chaque plateforme : x86, AMD64 et Itanium.

64 bits executable at run time on 64 bits windows. In this case, dotNet Protector will build 3 executable, one for each platform : x86, ADM64 and Itanium.

If you don't have the 64 bits platforms available to test the 64 bits exe, forget them and distribute only the x86 one.

If you can test the 64 bits, you can build up to 3 setup projects, one for x86, one for AMD64 and one for Itanium. (change the deployment project's 'TargetPlatform' property accordingly)

Si vous n'avez pas de plateforme 64 bits disponible pour tester les exe 64 bits, oubliez-les et distribuez uniquement la version x86. Si vous pouvez tester et supporter les 64 bits, vous pouvez générer jusqu'à 3 projets de déploiement, un pour x86, un pour AMD64 et un pour Itanium (changez la propriété TargetPlatform du projet).

Compiler le projet vdproj

Cette étape est la meme pour chaque plateforme.

Pour générer le projet de déploiement, vous devez fournir un embedded.netmodule compatible. Copiez le fichier de EmbeddedSetupPatch dans le même répertoire que votre exe.

Générez le projet. Dans la suite, on supposera que le vdproj génère *mysetup.msi*

ATTENTION! Si vous exécutez votre exe avec embedded.netmodule sur disque, le framework charger celui-ci al lieu de la version embarquée, et votre programme échouera. Ce module bidon doit être absent du repertoire de destination.

La seconde étape supprime embedded.netmodule du repertoire de destination en modifiant la base msi.

Exécutez la commande

MsiRemoveEmbeddedModule mysetup.msi

Après cette étape, mysetup.msi n'installera plus embedded.netmodule.

Génération automatique du projet vdproj

Vous pouvez automatiser le processus de generation en utilisant un fichier de commandes

Supposons que

Le chemin de la solution est 'C:\My Projects\mysetup.sln'

Le chemin de sortie de dotNet Protector est 'C:\My Projects\MyExe\Protected

EmbeddedSetupPatch a été décompressé dans C:\ EmbeddedSetupPatch

X86 Seulement

Le nom du projet est mysetup, sa sortie est dans 'C:\My Projects\mysetup\Release'

Le fichier de commandes pourrait être

```
@ECHO OFF
COPY /Y "C:\dotNetProtector\EmbeddedSetupPatch\embedded.netmodule" "C:\My Projects\MyExe\Protected"
SETLOCAL
CALL "%VS80COMNTOOLS%\VSVARS32"
"%VSINSTALLDIR%\Common7\IDE\DEVENV" "C:\My Projects\mysetup.sln" /build "Release" /project mysetup
"C:\EmbeddedSetupPatch\MsiRemoveEmbeddedModule" "C:\My Projects\mysetup\Release\mysetup.msi"
ENDLOCAL
DEL C:\My Projects\MyExe\Protected\embedded.netmodule
```

Plateformes multiples

Les chemins de sortie dotNet Protector sont 'C:\My Projects\MyExe\x86', 'C:\My Projects\MyExe\AMD64' et 'C:\My Projects\MyExe\Itanium' respectivement

Le projet x86 est mysetupx86, sa sortie est 'C:\My Projects\mysetupx86\Release',

Le projet AMD64 est mysetupx64, sa sortie est 'C:\My Projects\mysetupx64\Release', et

Le projet Itanium est mysetupia64, sa sortie est 'C:\My Projects\mysetupia64\Release'

Le fichier de commandes pourrait être

```
@ECHO OFF
COPY /Y "C:\dotNetProtector\EmbeddedSetupPatch\embedded.netmodule" "C:\My Projects\MyExe\x86"
COPY /Y "C:\dotNetProtector\EmbeddedSetupPatch\embedded.netmodule" "C:\My Projects\MyExe\AMD64"
COPY /Y "C:\dotNetProtector\EmbeddedSetupPatch\embedded.netmodule" "C:\My Projects\MyExe\Itanium"
SETLOCAL
CALL "%VS80COMNTOOLS%\VSVARS32"
"%VSINSTALLDIR%\Common7\IDE\DEVENV" "C:\My Projects\mysetup.sln" /build "Release" /project mysetupx86
"C:\EmbeddedSetupPatch\MsiRemoveEmbeddedModule" "C:\My Projects\mysetupx86\Release\mysetupx86.msi"
"%VSINSTALLDIR%\Common7\IDE\DEVENV" "C:\My Projects\mysetup.sln" /build "Release" /project mysetupx64
"C:\EmbeddedSetupPatch\MsiRemoveEmbeddedModule" "C:\My Projects\mysetupx64\Release\mysetupx64.msi"
"%VSINSTALLDIR%\Common7\IDE\DEVENV" "C:\My Projects\mysetup.sln" /build "Release" /project mysetupia64
"C:\EmbeddedSetupPatch\MsiRemoveEmbeddedModule" "C:\My Projects\mysetupia64\Release\mysetupia64.msi"
ENDLOCAL
DEL C:\My Projects\MyExe\x86\embedded.netmodule
DEL C:\My Projects\MyExe\AMD64\embedded.netmodule
DEL C:\My Projects\MyExe\Itanium\embedded.netmodule
```

2. Projects qui n'embarquent pas la runtime dotNet Protector

In projects not embedding the runtime, the vdproj should build correctly, but in these projects, dotNet Protector's runtime is needed to run.

Dans les projets n'embarquant pas la runtime, le vdproj doit se générer sans problème, mais la runtime est nécessaire à l'exécution

La runtime dotNet Protector est composée de 5 fichiers :

- . PvLogiciels.dotNetProtector.Runtime.dll : c'est la runtime commune, nécessaire dans tous les cas
- . PvLogiciels.dotNetProtector.RuntimeV1.dll : c'est la runtime compatible framework v1.1. Vous n'en n'avez pas besoin si votre assembly est un v2 ou plus.
- . PvLogiciels.dotNetProtector.RuntimeX86.dll : c'est la runtime v2.0 x86. Vous en avez besoin si votre assembly est chargé par la v2(ou ultérieure) du framework .NET, excepté si votre assembly ne s'exécute que sur une plateforme 64 bits.
- . PvLogiciels.dotNetProtector.RuntimeAMD64.dll : c'est la runtime v2.0 ADM64 runtime. Vous en avez besoin si votre assembly est chargé par la v2(ou ultérieure) du framework .NET sur windows x64, excepté si votre assembly ne s'exécute que sur x86 (x86 au lieu de AnyCPU).
- . PvLogiciels.dotNetProtector.RuntimeItanium.dll : c'est la runtime v2.0 Itanium. Vous en avez besoin si votre assembly est chargé par la v2(ou ultérieure) du framework .NET sur windows IA64, excepté si votre assembly ne s'exécute que sur x86 (x86 au lieu de AnyCPU)

If your assembly is a v2+ assembly and not built to run only on a 32 box (x86), you NEED 3 setup projects (for each platform)

Si votre assembly est v2+ et non généré pour fonctionner en mode x86 seulement, vous AVEZ besoin de 3 projets de deployment.

Ajout de la runtime au projet. Si la runtime commune est présente dans le répertoire de votre assembly, Visual Studio l'ajoutera de lui-même. Dans le cas contraire, ajoutez-la manuellement dans le répertoire de votre application.

Suivant la version/plateforme de votre projet de deployment, vous devez ajouter des fichiers supplémentaires:
assembly V1.1 /projet x86: ajoutez RuntimeV1 et RuntimeX86
assembly V1.1 / projet x64: ajoutez RuntimeV1, RuntimeX86 et RuntimeAMD64
assembly V1.1 / projet IA64: ajoutez RuntimeV1, RuntimeX86 et RuntimeItanium

assembly V2+/ projet x86: ajoutez RuntimeX86
assembly V2+/ projet x64: ajoutez RuntimeX86 et RuntimeAMD64
assembly V2+/ projet IA64: ajoutez RuntimeX86 et RuntimeItanium