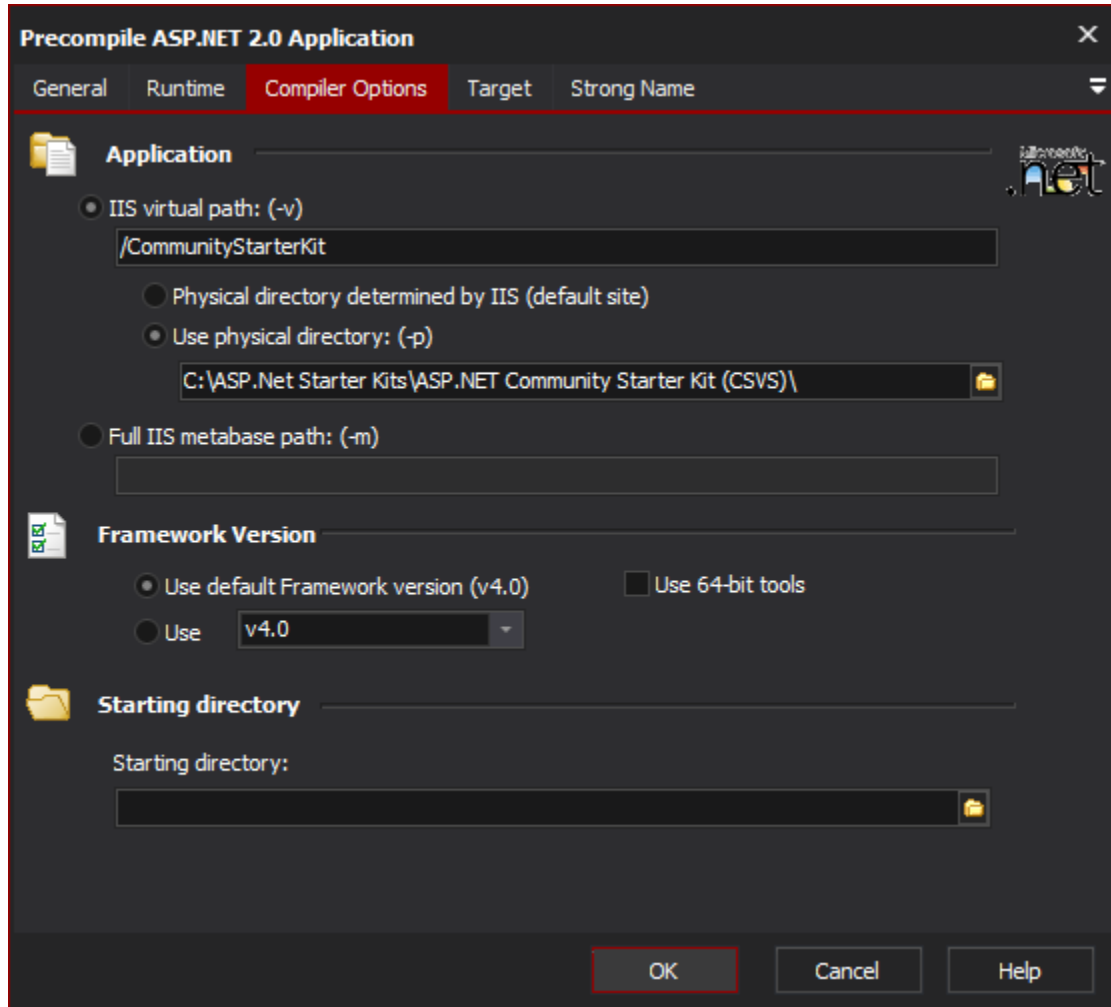


# Precompile ASP.NET 2.0 Application Action

The ASP.NET Precompiler action allows you to compile ASP.NET applications before deployment.

Precompiling has two main advantages:

- The ASP application does not need to be compiled when first accessed.
- By deploying only compiled ASP applications, the ASPX source files do not need to be in an IIS virtual directory.



The screenshot shows the 'Precompile ASP.NET 2.0 Application' dialog box with the 'Compiler Options' tab selected. The dialog has a title bar with a close button (X) and a tab bar with 'General', 'Runtime', 'Compiler Options', 'Target', and 'Strong Name'. The 'Application' section has three radio buttons: 'IIS virtual path: (-v)' (selected), 'Physical directory determined by IIS (default site)', and 'Use physical directory: (-p)'. The 'IIS virtual path' is set to '/CommunityStarterKit'. The 'Use physical directory' option is set to 'C:\ASP.Net Starter Kits\ASP.NET Community Starter Kit (CSV5)\'. The 'Full IIS metabase path: (-m)' is empty. The 'Framework Version' section has two radio buttons: 'Use default Framework version (v4.0)' (selected) and 'Use' (with a dropdown set to 'v4.0'). There is also a checkbox for 'Use 64-bit tools'. The 'Starting directory' section has a text box for 'Starting directory:' which is empty. At the bottom are 'OK', 'Cancel', and 'Help' buttons.

On this page:

## Compiler Options

### Application

There are three ways to specify the path to your ASP.NET Application:

#### IIS virtual path with physical directory determined by IIS

The application will be located by using the IIS metabase and the default website. The physical (local) folder will be determined via the IIS metabase.

#### IIS virtual path with using physical directory

The application will be compiled from the specified physical (local) directory, but as if it was installed at the IIS virtual directory given by the path. This is useful if you want to compile your ASP.NET application from one (offline) directory, but deploy the assemblies in another directory.

## **Full IIS metabase path**

The application will be located by using a full IIS metabase path. The physical (local) folder will be determined via the IIS metabase.

## **Framework version**

### **Use default Framework version (vX.X) / Use vX.X**

Allows the selection of the .NET version to use for the tool. The minimum is .NET v2.0.

### **Use 64-bit tools**

This options specifies as whether to force the usage of 64-bit tools on a 64-bit system or not. As Automise is currently a 32-bit application it will default to running the 32-bit version on a 64-bit system. Check this option if you require the 64-bit version to be run.

## **Starting Directory**

### **Starting Directory**

The directory that pre-compile will be based in. If not set then its the currently working directory of the build.

## **Target**

### **Target Output Directory**

#### **Compile the application in-place**

The application will compile to the same directory as the source.

#### **Fully rebuild target application**

All sources will be rebuilt, not just those which have changed.

#### **Compile the application to target directory**

The application will compile to the specified target directory. Compiling to a target directory automatically implies "fully rebuild target application."

#### **Create updatable application**

The compiled application will be updatable.

#### **Overwrite the target directory**

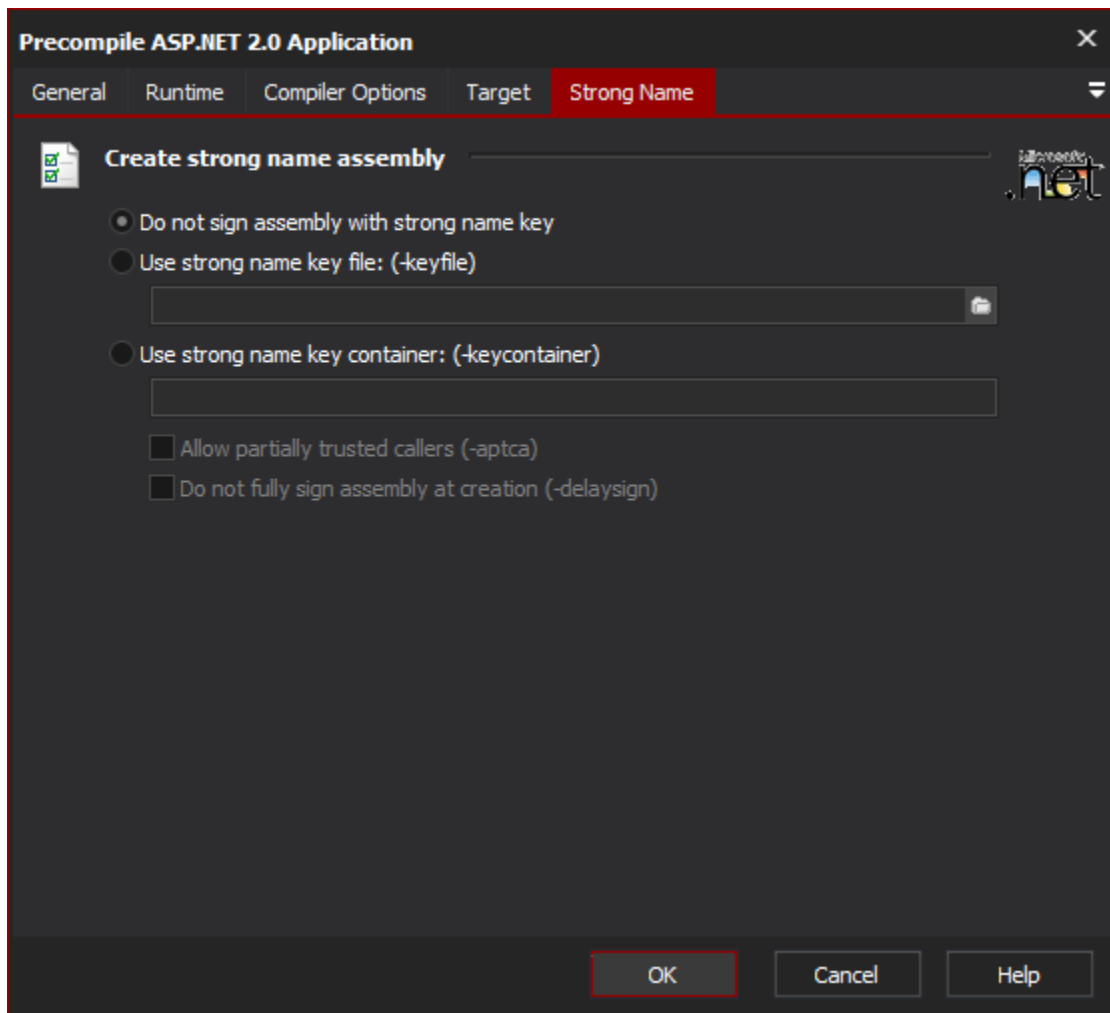
If the target directory already exists, its contents will be overwritten.

#### **Emit debug information**

The compiler will emit debug information to the log during the compile process.

#### **Use fixed names for compiled assemblies**

The compiled assemblies will be given fixed names. If this option is not set, the names will be automatically generated.



### Show error stack if fails to compile

Specify if a stack trace is required in the action output when a failure occurs.

## Strong Name

### Create Strong Name Assembly

#### Do not sign assembly with strong name

Choose not to sign the assemblies with a strong name. This means that weak naming is required to reference any assemblies created.

#### Use strong name key file

The filename which has the public/private key pair to be used in generating the strong name.

#### Use strong name key container

The name of the container which has the public/private key pair to be used in generating the strong name.

#### Allow partially trusted callers

This allows the produced code to be accessed by partially trusted callers.

#### Do not fully sign assembly at creation

Specifies that the assembly should only be signed with the public key token rather than the public/private key pair. The produced code can be run before signing is completed. Take great care that no malicious users can access before signing has been completed.