VBConversions VB.Net to C# Converter

Welcome

Welcome to the VBConversions VB.Net to C# Converter!

VBConversions has the most accurate VB.Net to C# Converter on the market and has won the coveted Visual Studio Magazine Readers Choice Award (Developer Tools Category) as well as several other industry awards.

The VBConversions VB.Net to C# Converter supports the latest features of VB.Net as well as every project type and Visual Studio version (2003, 2005, 2008, and 2010). It is used by many of the world's leading companies to convert their VB.Net code to C#. See what it can do for you!

Some of our customers include:

Summary of Features

VBConversions supports converting all versions of Visual Basic.Net to C#

- Visual Basic 2003
- Visual Basic 2005
- Visual Basic 2008
- Visual Basic 2010

All VB.Net project types are supported:

- Windows forms projects
- Web projects (requires Web Application Project plug-in for VS 2005)
- Compact Framework projects
- WPF projects
- Console Application projects
- Class Library projects

Support for older VB.Net code:

Your VB.net code doesn't have to be perfect to convert to C#

- Best in class On Error Goto to try/catch conversions
- Unnecessary ByRef arguments are automatically converted to be passed by value
- Conversion of built in VB functions (Len, MsgBox, UCase, etc) to their .Net equivalents
- Support for ReDim, Err object, VB constants (VBCrlf, VBTab, etc), and other VB centric features
- Automatically type determination for untyped constants

Support for all the latest VB.Net features:

VBConversions is the first to convert new VB.Net features after every Visual Studio release!

- LINQ
- In-line XML, including "code hole" support
- Object initializers
- Extension methods
- Lambda expressions

- Nullable data types
- Implicit types
- Anonymous types

Support for the tough conversion issues:

- Complex Case statements (not allowed in C#) , including ranges, lists, and non-integer case variables.
- Parameterized properties
- Correct casing of identifiers (VB is case insensitive, C# is case sensitive)
- Custom event handling
- Correct conversion of With statements
- Automatic commenting out of dead code (dead code isn't allowed in C#)
- Auto initializing variables if necessary
- Automatic correction of most C# compiler errors in the generated code
- and much, much more...

Multiple project conversion support:

VBConversions uses this feature to volume test releases on over 5,000 projects

- Large groups of projects can be converted at once
- Comprehensive reporting on groups of projects
- Perform VB.Net or C# compilation of groups of projects
- Click here for more details on converting multiple projects

Integrated Side by Side Code Browser:

- Navigate your VB.Net and Converted C# code side by side in a Visual Studio like browser
- Selecting a line of code in VB.Net or C# will highlight the equivalent lines in the other language
- Easily review any potential problems, conversion notes, or C# compiler errors

Customization:

Your VB.Net projects are unique and may require custom handling during conversion. There are literally dozens of conversion options you can configure to meet your specific needs:

- Detailed control of C# code formatting
- Control the program navigation extensively, determining which review screens to stop at and which warnings to see or ignore
- Specify which VB namespace functions (Mid, Right, Left, IIF, etc) you want converted to their .Net equivalents.
- Choose to have unreachable code commented out (to prevent C# compiler errors)
- Decide how ref and out arguments are converted
- Specify how With statements are to be converted
- Choose if X=X+1 and X+=1 are to be converted to X++
- Plus many more options. Click here for more details on setting conversion options.

Comprehensive reporting:

You're not left wondering what you should do to prepare for conversion or what happened during the conversion process. You will get comprehensive reports on:

- Potential problem areas of your code
- Conversion notes - interesting things which happened during conversion
- Any C# compiler errors which could not automatically be fixed by the converter
- and much more...

Frequently Asked Questions

What versions of Visual Studio are supported?

All versions of Visual Studio are supported: 2003, 2005, 2008, 2010.

Is my VB.Net code guaranteed to translate to C# perfectly?

No. Due to Visual Basic's inherent ambiguities (which VBConversions goes the extra mile to resolve), no automated converter can always translate VB.Net to C# perfectly. However, many programs do convert without errors. The 101 sample projects from Microsoft convert and compile without errors, as do hundreds of other publicly available sample VB.Net programs.

What types of VB.Net projects can be converted?

All project types can be converted: Windows Forms, WPF, Console, Windows Service, Web Forms, Web Control, Windows Control, Class Library, and Compact Framework projects.

How Do I Convert Web Projects?

You can convert web sites by making them Web Application Projects. Web Application Projects is an add on to Visual Studio 2005 (but built in to Visual Studio 2008 and above). You can download the Visual Studio 2005 Web Application Project add-in here.

Can I Convert Visual Basic 6 Projects?

Visual Basic 6 projects can be upgraded with the VB6 Upgrade Wizard in Visual Studio and then converted as a VB.Net project. Make sure the upgraded project compiles successfully in VB.Net before attempting to convert it to C#.

Why Should I Choose VBConversions?

Accuracy. VBConversions is simply much more accurate than the competition, but don't take our word for it. You are encouraged to download all the available trial editions and evaluate them thoroughly. We are confident you'll agree that VBConversions has the most accurate VB.Net to C# converter available.

In addition to superior accuracy, VBConvesions offers many unique usability features, such as integrated side by side code browsing, comprehensive reporting, potential conversion problem detection, detailed conversion notes, and dozens of code conversion and C# code formatting options.

Can I Convert from C# to VB.Net?

No. Conversions from VB.Net to C# are supported only.

What's the Difference Between the Trial Version and the Full Version?

The trial version expires in 15 days and is limited to 1100 lines of code. The full version allows project conversions of unlimited size and doesn't expire.

Test Results

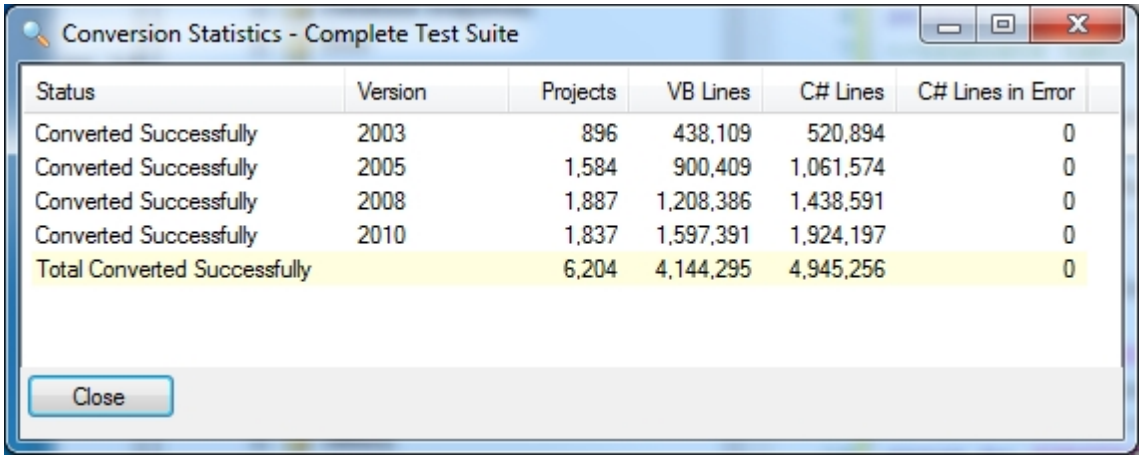
VBConversions Test Results

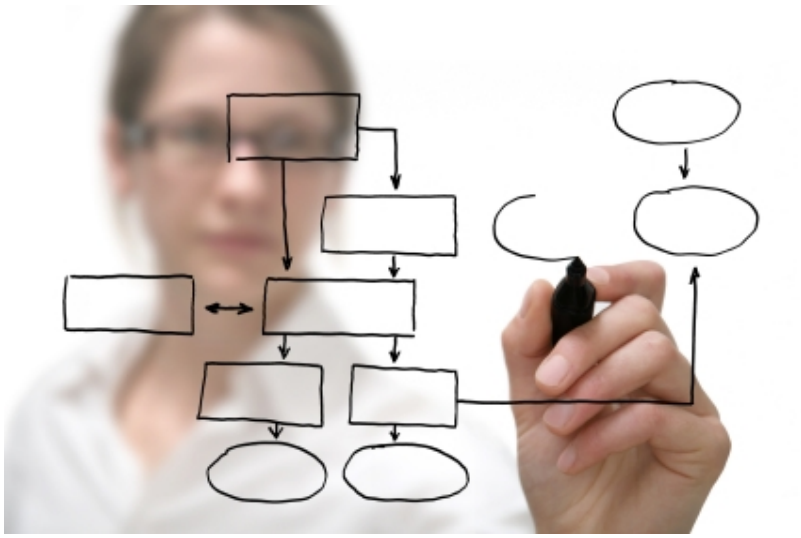
The Version 2.32 release was tested extensively on thousands of projects and millions of lines of code prior to releasing.

The native capabilities of the VB.Net to C# converter to convert multiple projects at once was used. Below is the Group Stats screen for the VBConversions test suite.

The test projects come from four sources:

1) Public domain VB.Net projects, such as the Microsoft 101 VB.Net samples, and sample projects on CodePlex, SourceForge, CodeProject, etc.
2) Sample projects from third-party controls, such as Telerik, Infragistics, Xceed. etc.
3) User contributed projects.
4) Projects written by VBConversions to test specific conversion scenarios.





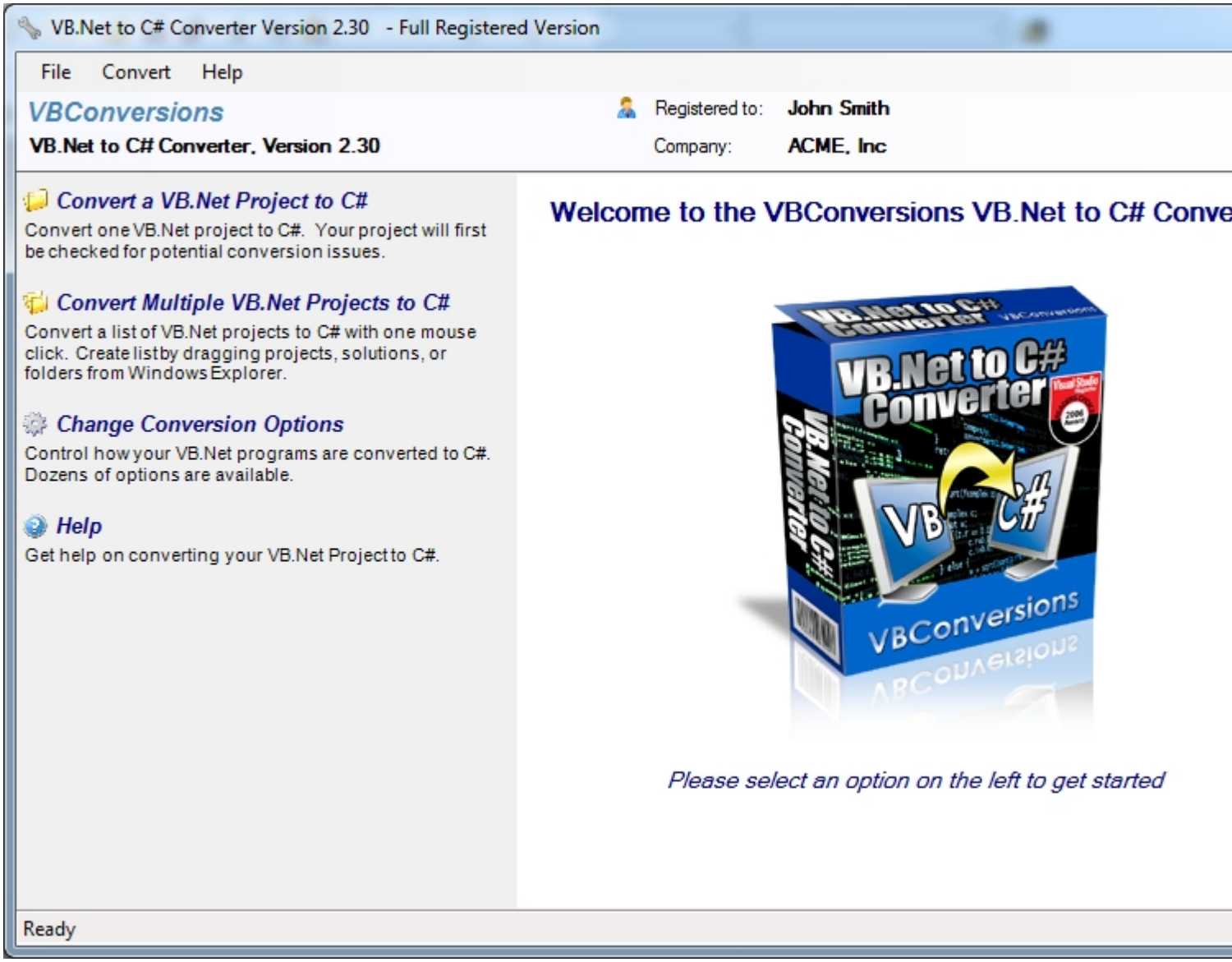
How to Convert a VB.Net Project to C#

There are four steps to converting a Visual Basic.Net project to C#:

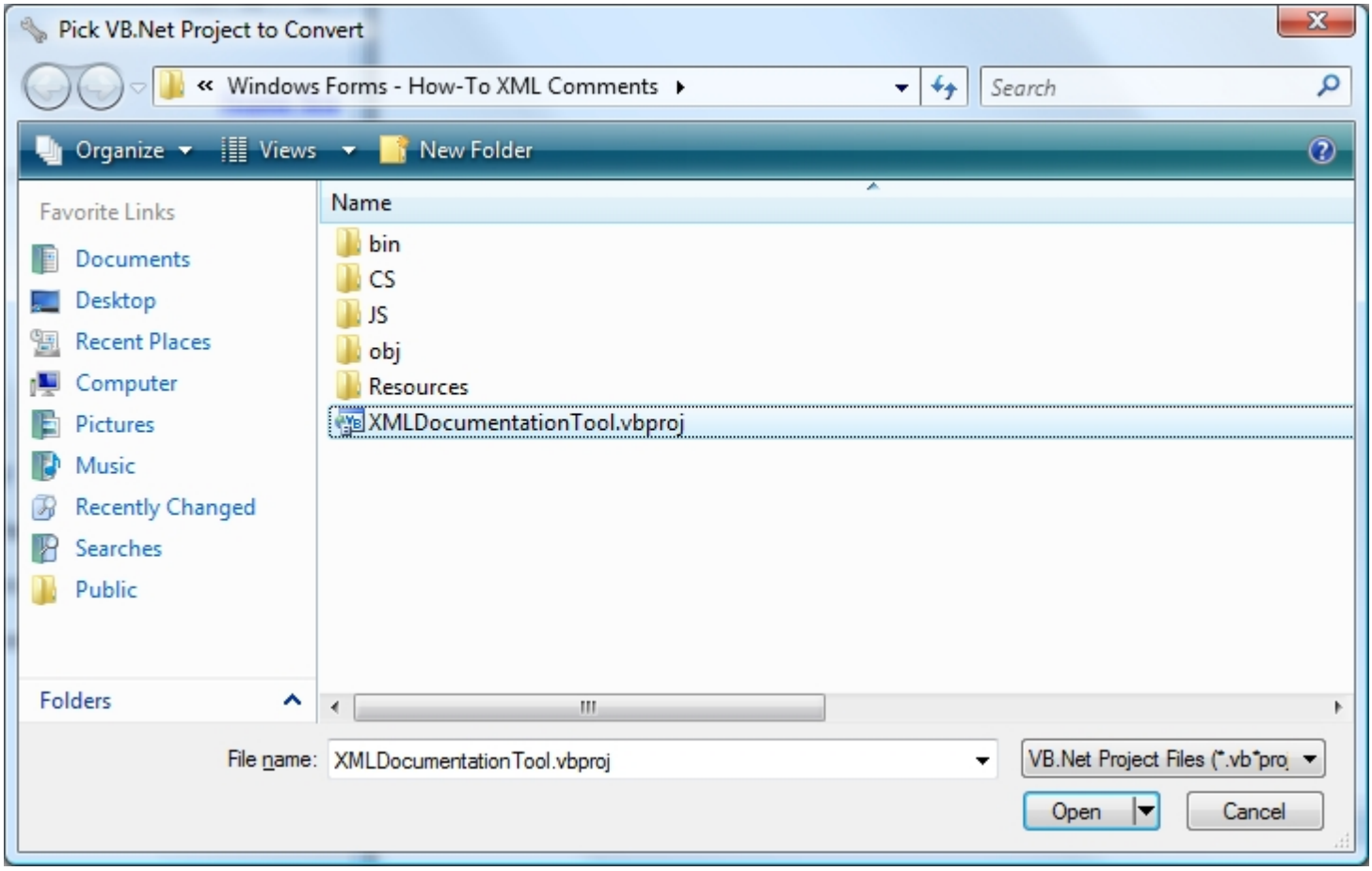
1) Check for potential conversion problems and correct
2) Convert your VB.Net project to C# and review conversion notes
3) Fix C# compiler errors (by changing VB.Net code if possible) and reconvert
4) Fix any runtime issues

Check for Potential Conversion Problems

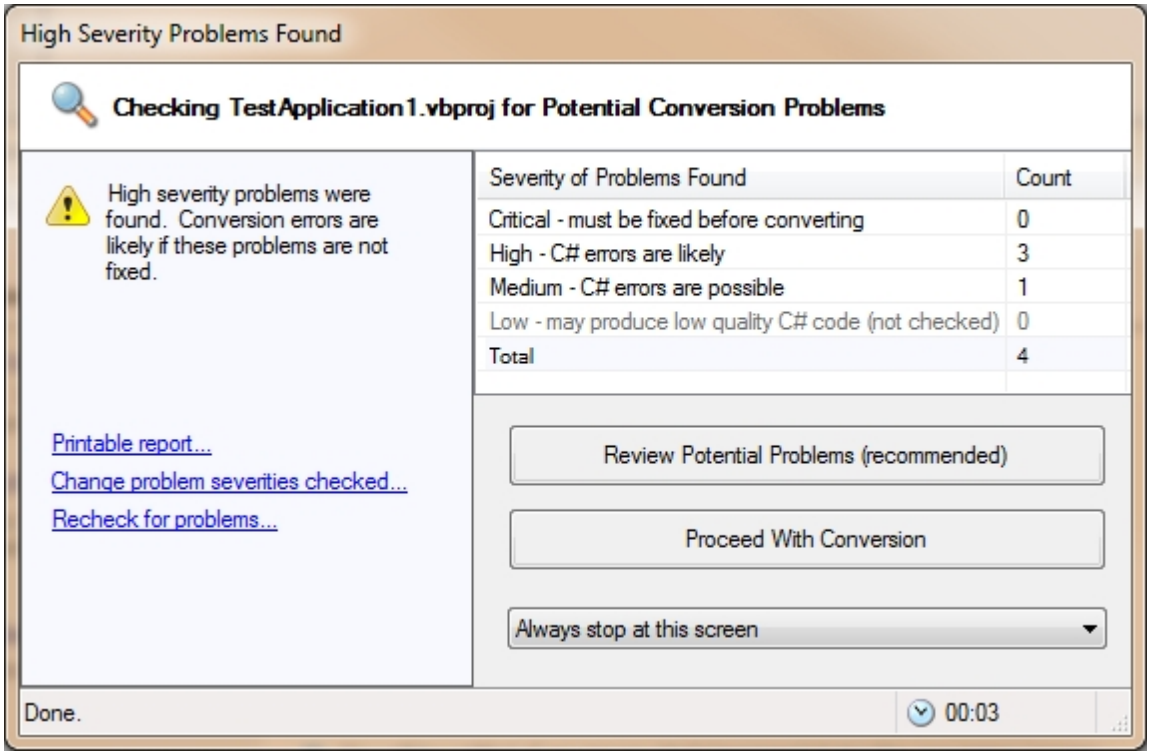
The first step in a conversion is to check for potential conversion problems. From the main screen, pick "Convert a VB.Net Project to C#". Before the conversion actually takes place, your project will be automatically checked for possible conversion issues.



Next, you will be prompted to pick a VB.Net project to convert. Choose your project and press "Open".

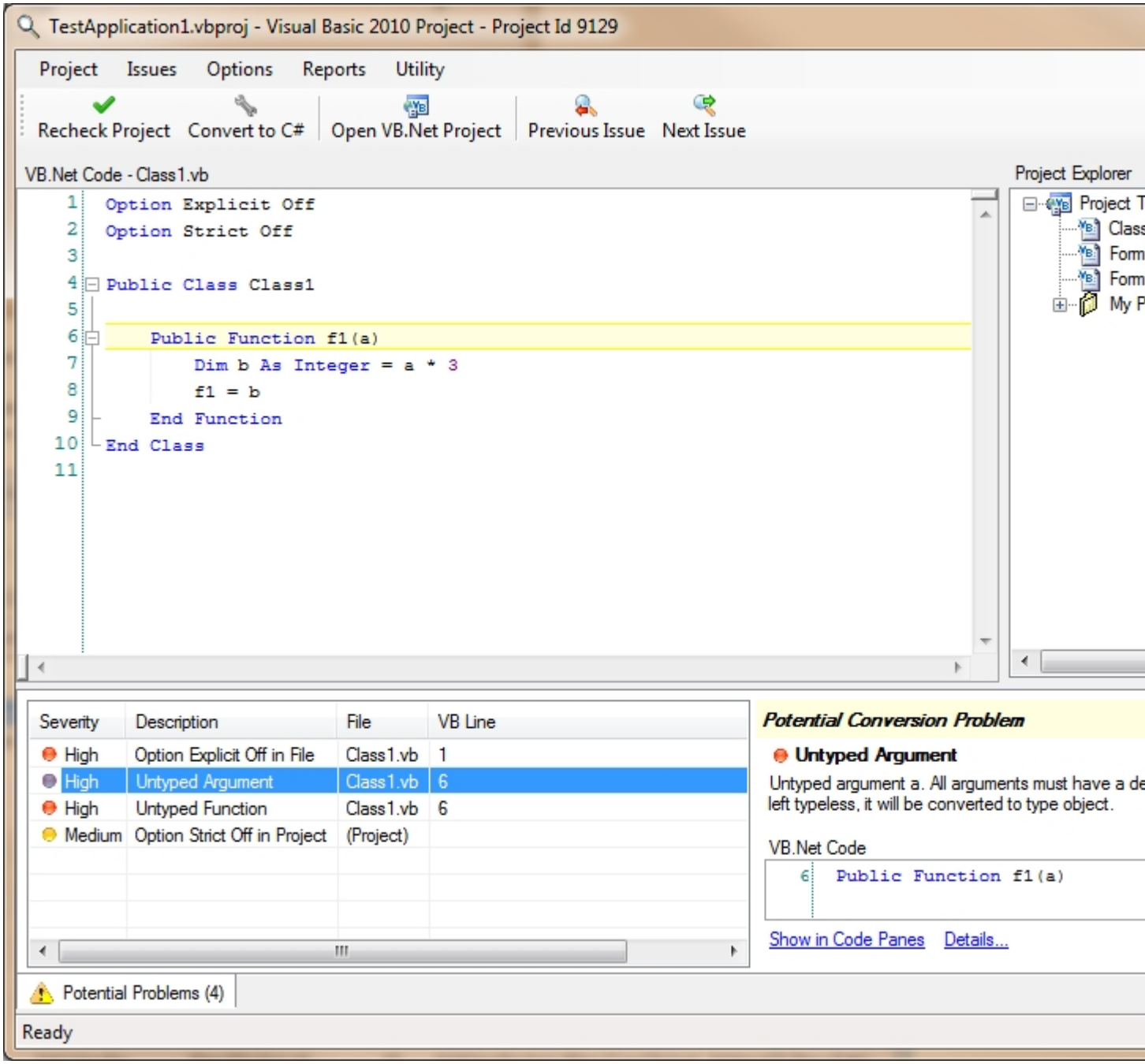


Your project will be first be checked for potential problems before converting and you will be shown a screen similar to the one below:



Press "Review Potential Problems" to see the potential conversion problems (recommended), or you can click "Proceed With Conversion" to immediately start the conversion.

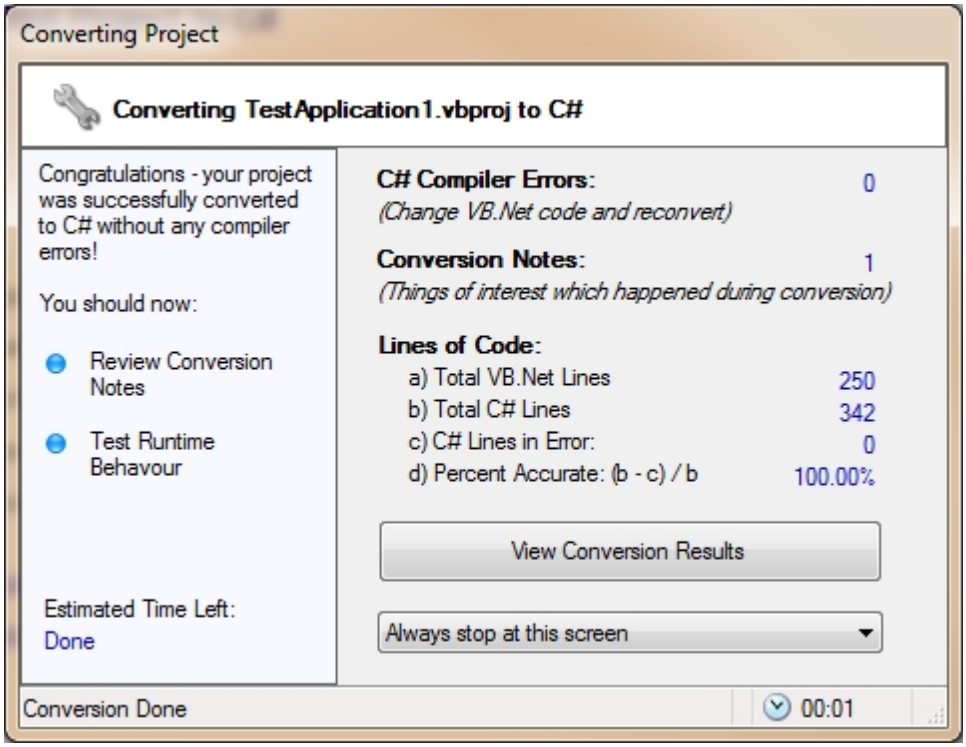
It is highly recommended to fix all high severity potential problems before attempting a conversion. Medium severity problems are often successfully handled during conversion, but high severity problems will almost always result in C# compiler errors and can interfere with the automatic C# compiler error fixing feature of the converter.



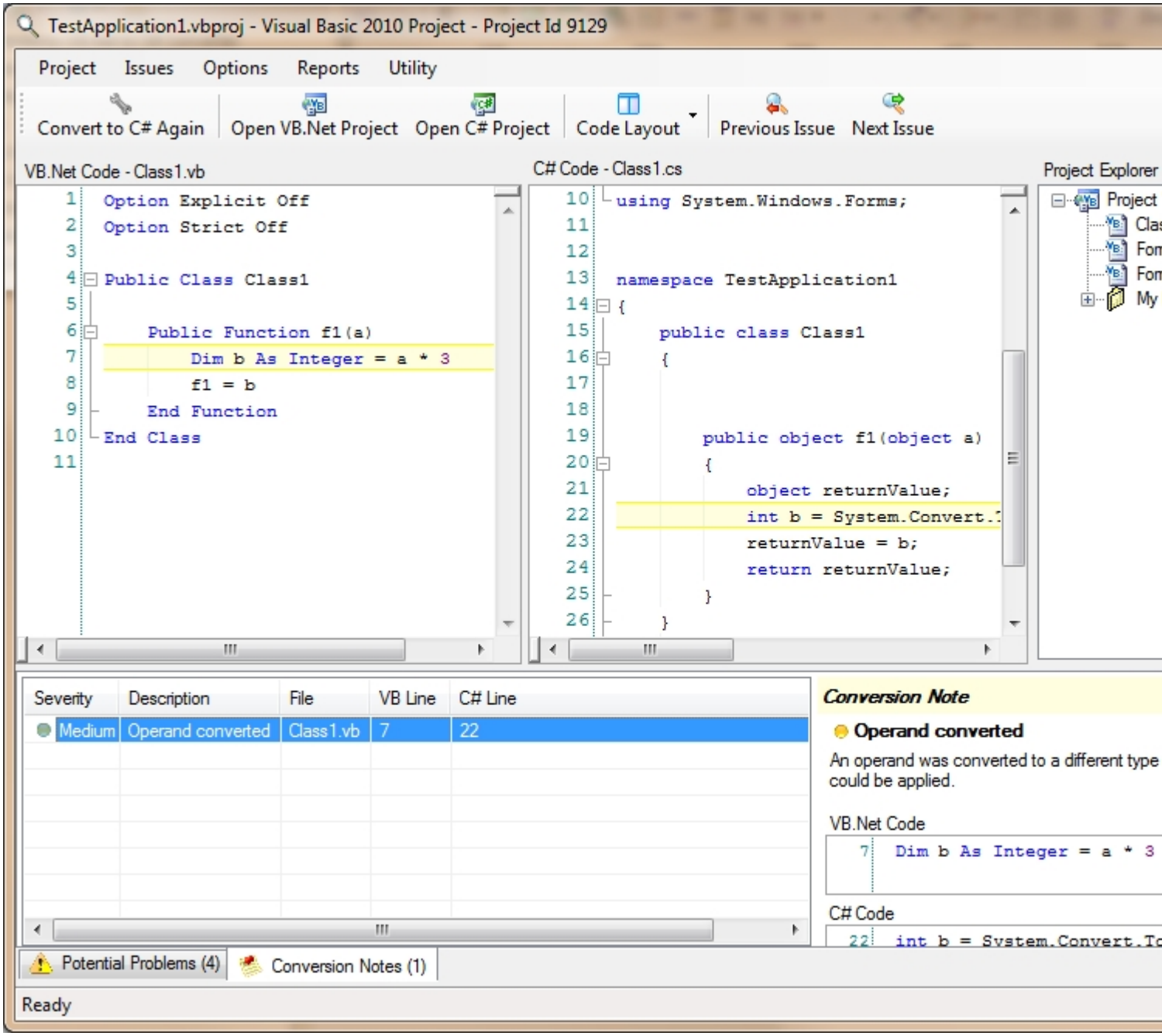
When you are done fixing potential conversion problems and are ready to convert, repeat the above steps but press "Proceed with Conversion".

Convert Project to C# and Review Conversion Notes

After proceeding with the conversion, you will see a conversion summary screen similar to the one below, which will list the statistics of the conversion, such as number of lines of code, and any conversion notes and C# compiler errors encountered.



Press "View Conversion Results" to see side by side VB.Net and C# code listings, and to review any potential problems, conversion notes, and compiler errors encountered:



You can double-click a conversion note, potential problem, or compiler error to see its corresponding VB.Net and C# code, or press the previous and next issue buttons to navigate through the issues encountered. Pay particular attention to any high severity conversion notes issued and try to correct the issues in the VB.Net code if possible and reconvert.

Fix C# compiler errors (by changing VB.Net code if possible) and reconvert

The converter fixes many C# compiler errors for you, but there may be some the converter can't automatically fix. If your project encounters C# compiler errors, you should attempt to change the source VB.Net code if possible to avoid the errors and convert again.

Fixing C# errors manually may expose other C# compiler errors the converter hasn't had a chance to identify and correct, since compiler errors tend to be produced in waves starting with the most severe. These additional errors would normally be corrected by the converter in its normal processing, so you can save a huge amount of work by changing the VB.Net code and converting again rather than changing the C# code directly.

Fix Runtime Problems

Once your project compiles successfully, the runtime behavior should be thoroughly tested.

One potential issue to check is relative path names. For example, if your initial VB.Net code was:

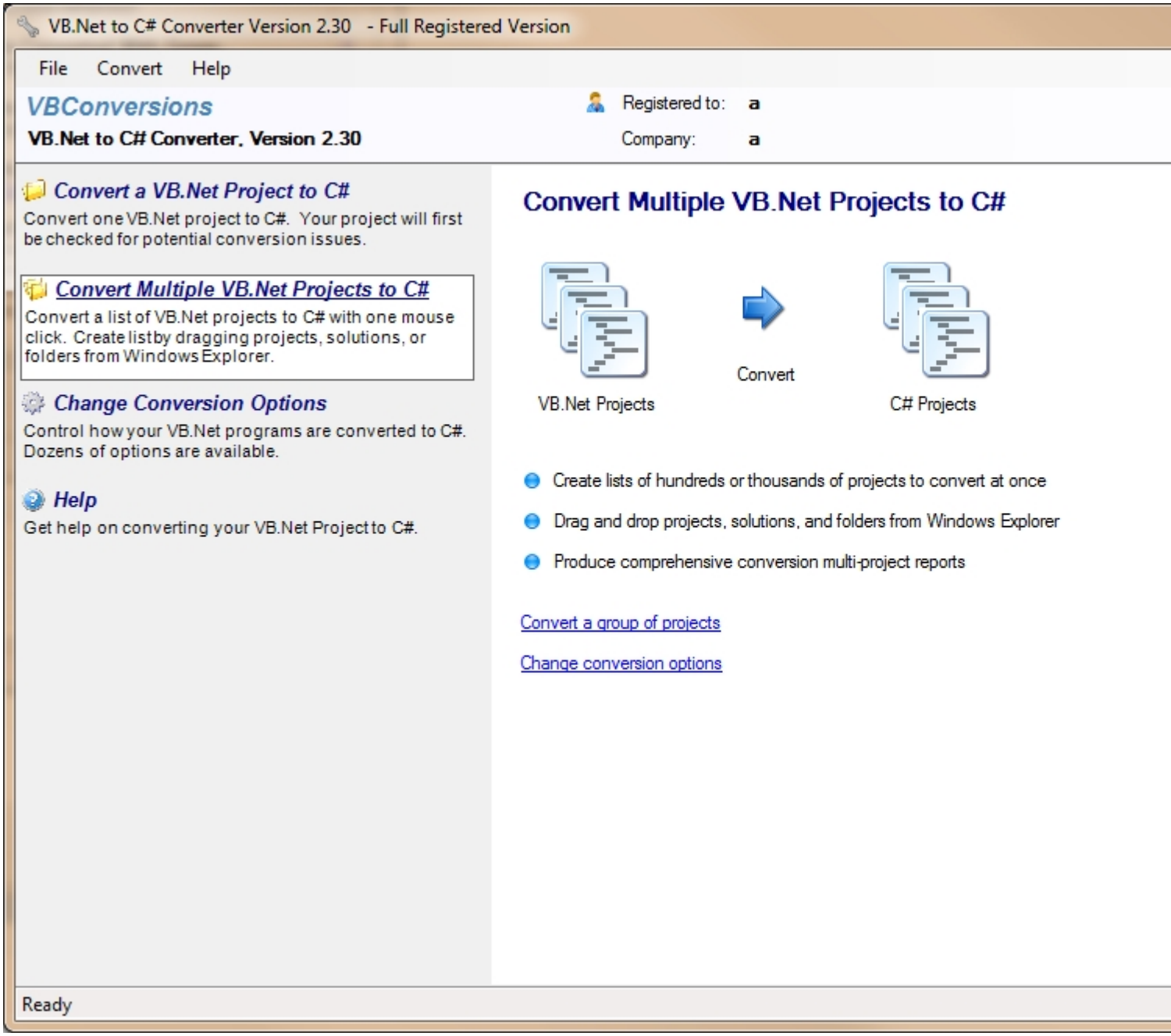
```
Dim sr As New System.IO.StreamReader("..\test.txt")
```

Your converted C# code is now in a different directory and the path may be invalid or the file not found at the new relative location. You should change the code to use a different relative path name or explicitly point to the appropriate directory.

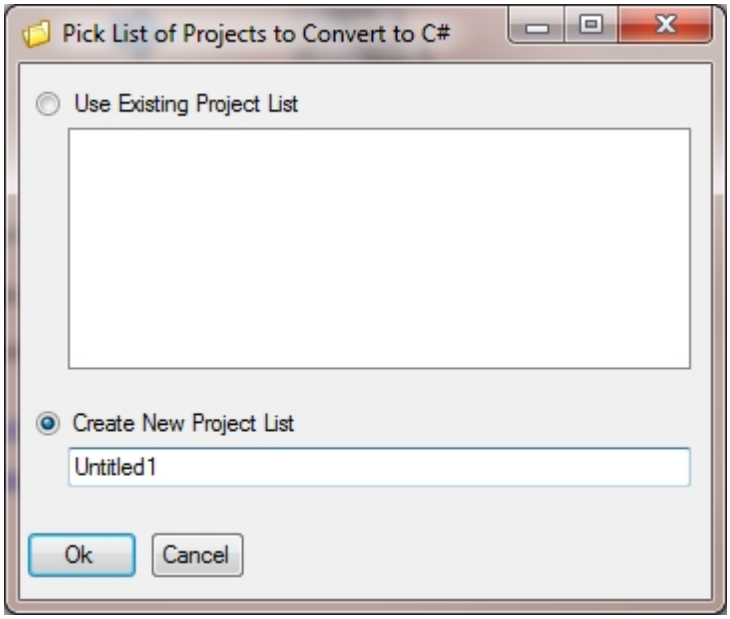
How to Convert Multiple VB.Net Projects to C#

You can convert multiple projects at once using the VBConversions VB.Net to C# Converter. You simply create a *project list*, add projects to the list, then convert the whole list of projects at once. You may also run over a dozen printable reports to help you manage converting multiple projects. VBConversions itself uses a project list of over 3000 programs to test each release.

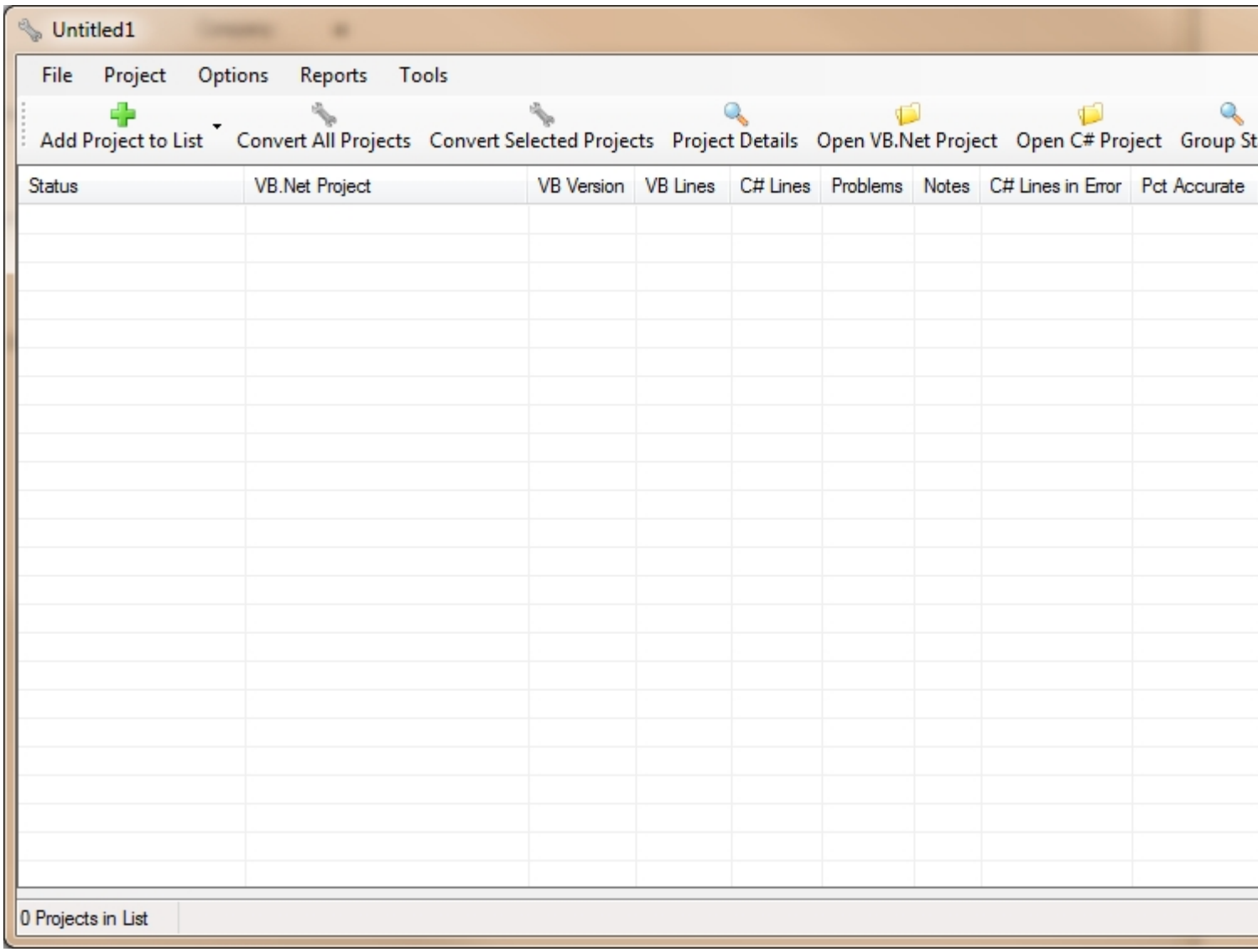
To convert multiple projects, pick "Convert Multiple VB.Net Projects to C#" from the main menu:



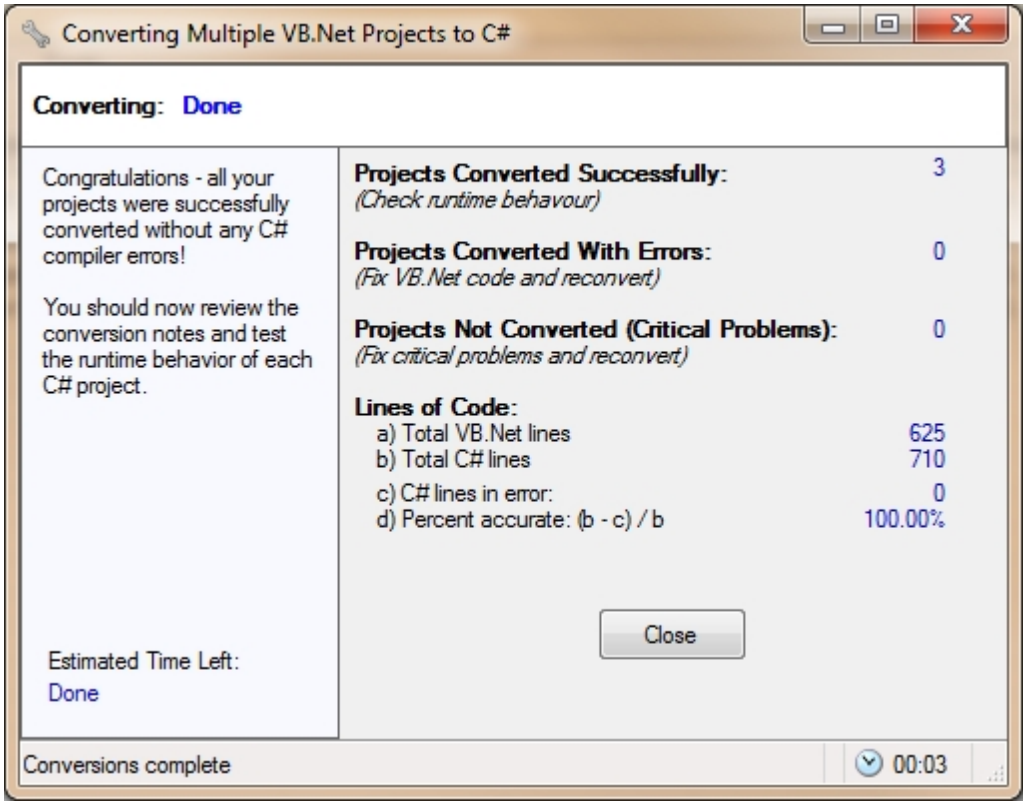
Type in the name of your new project list to create and press Ok. The next time you enter this screen, your new project list will automatically appear in the list of Existing Project Lists.



Next you can add projects to you project list. The easiest way to add them is to simply drag and drop folders, project files, or solutions from Windows Explorer, but you can also add projects via the menu options and toolbar buttons as well. When done adding projects to the list, you can convert all the projects at once to C#. The are also 18 different printable reports to help you manage converting multiple projects.



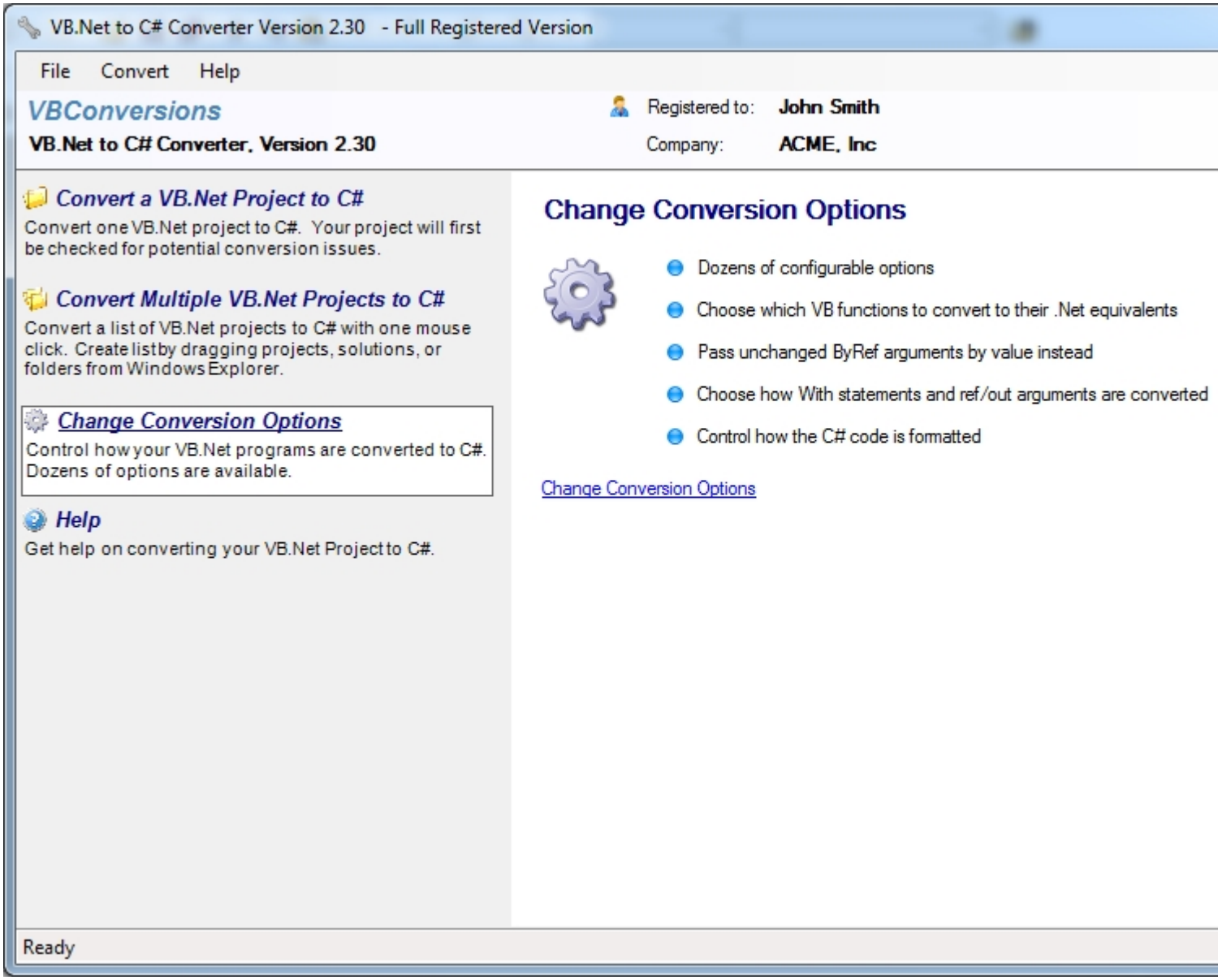
When "Convert All" or "Convert Selected" is pressed, the following screen is displayed showing the progress of the conversions. When done, close this window and you will be taken back to your project list where you can fix problems with individual projects or run summary reports on the whole list.



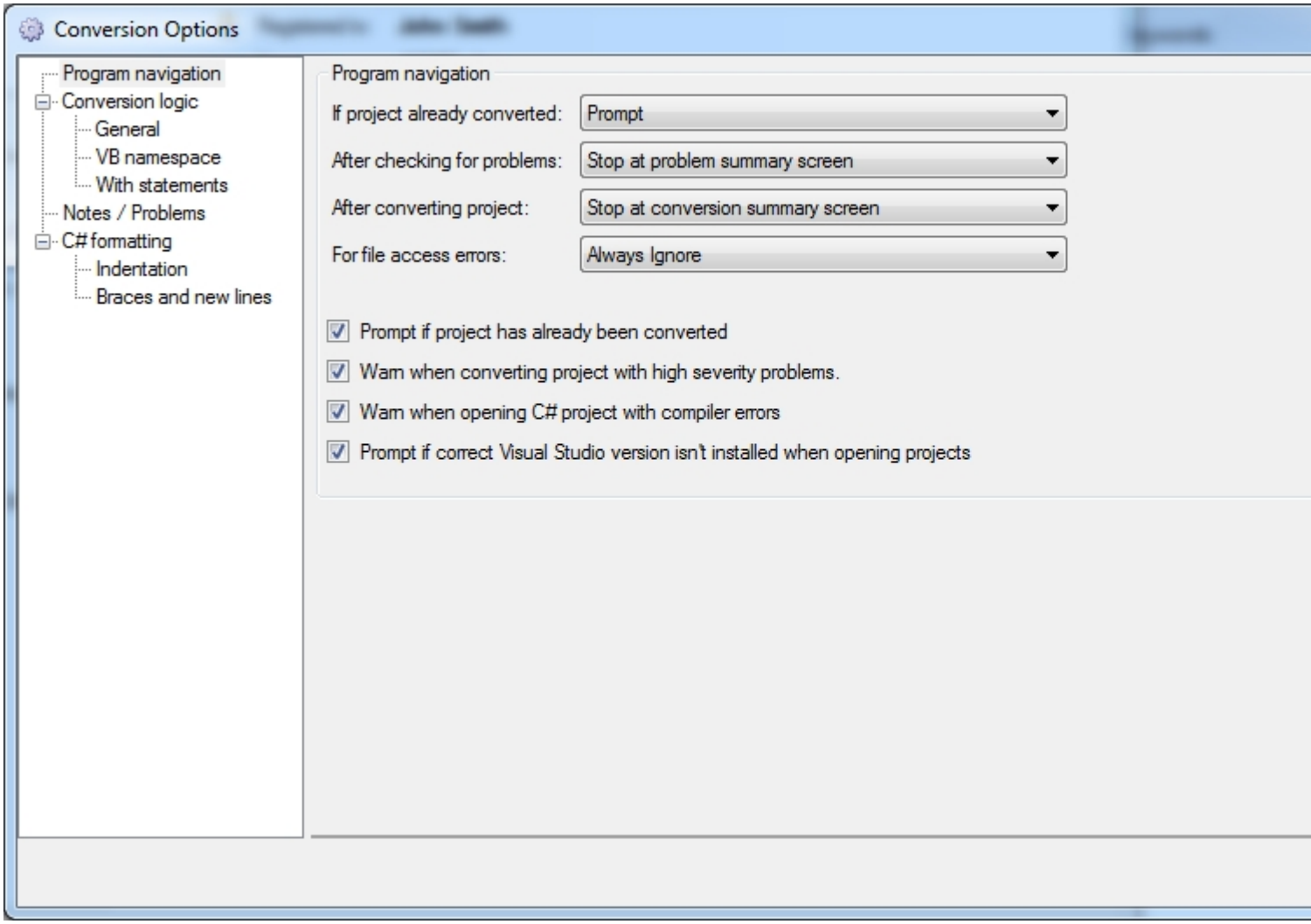
Conversion Options

Conversion Options

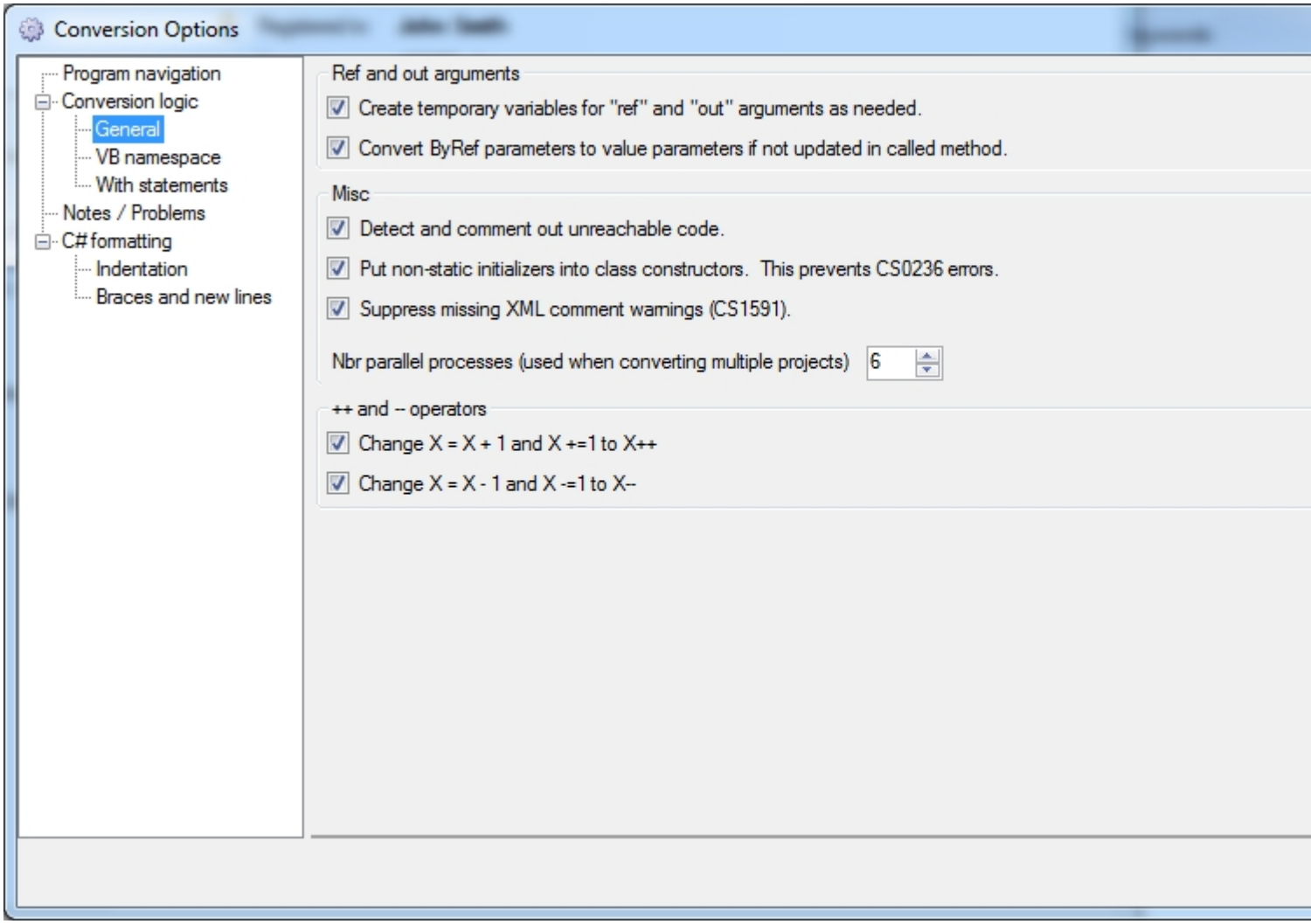
Your project is unique, and you can configure dozens of conversion options to suit the needs of your particular application. To see the conversion options available, choose "Change Conversion Options" from the main screen.



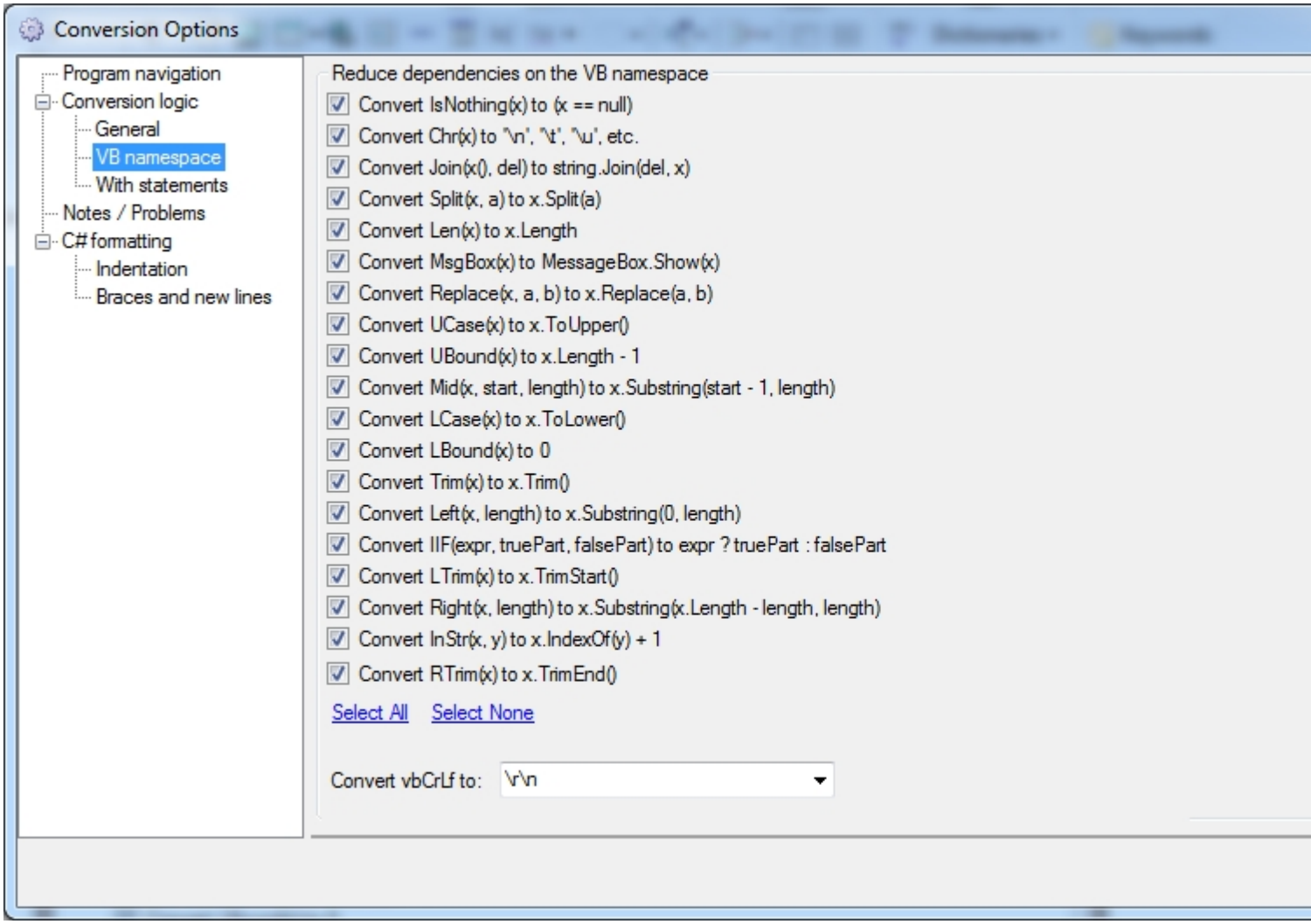
The program navigation can be customized to turn warnings off and on, and to go from picking a project to convert to browsing the converted code with one click. It is highly recommended, however, than you stop at the problem summary screen and review any potential conversion problems before proceeding.



General conversion options control how ref and out arguments are to be converted, the number of parallel processes to use when converting multiple projects, and other miscellaneous conversion options. Be careful when changing the default options in the "Ref and Out Arguments" and "Misc" sections, as this can have a negative effect on conversion accuracy.



VB Namespace options allow you to specify which VB specific functions you want converted to their .Net equivalents. In general, you should leave all these options checked, unless a particular item is causing problems in your conversion.

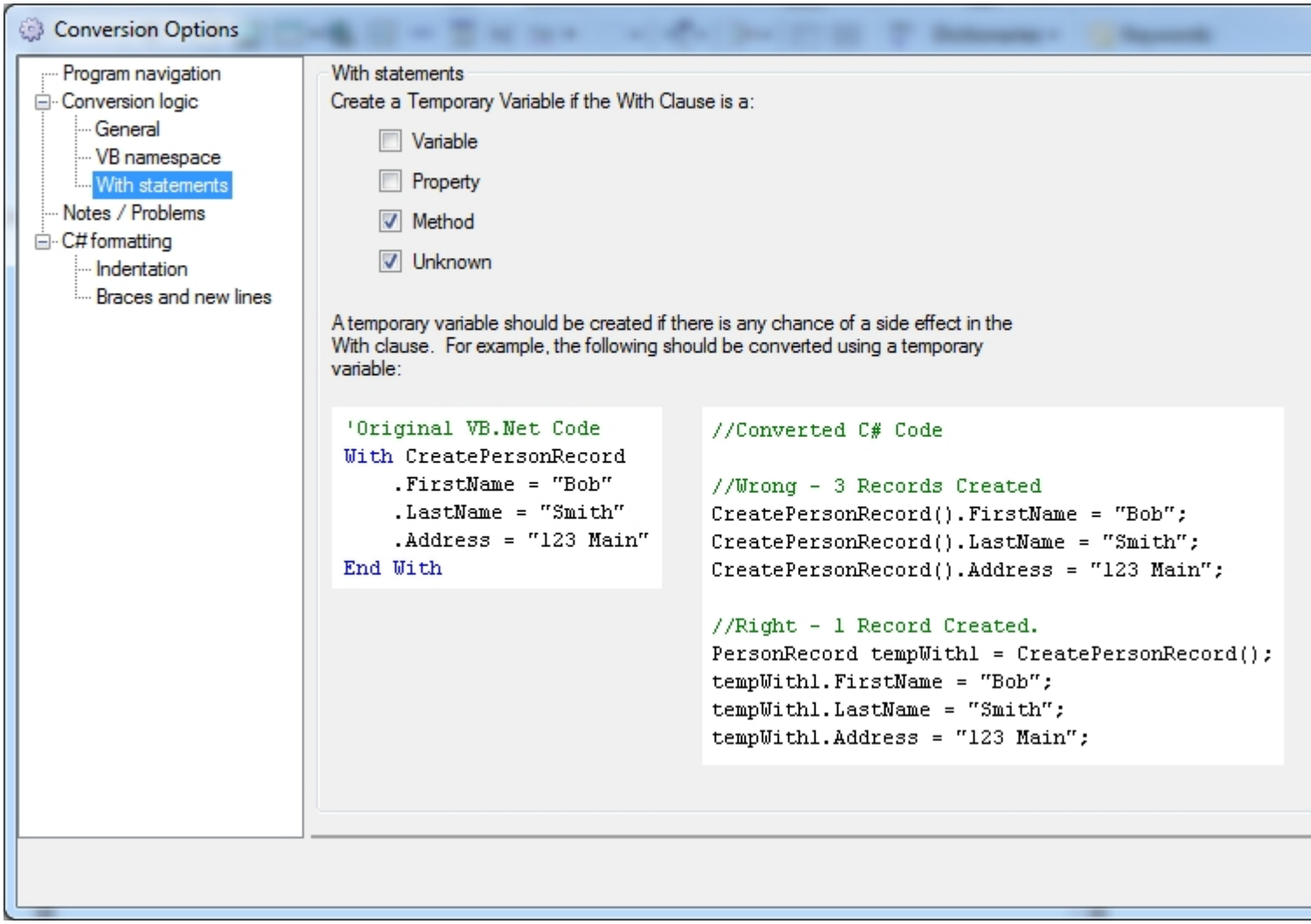


The "With" options control how VB With clauses are converted. They can be converted two ways:

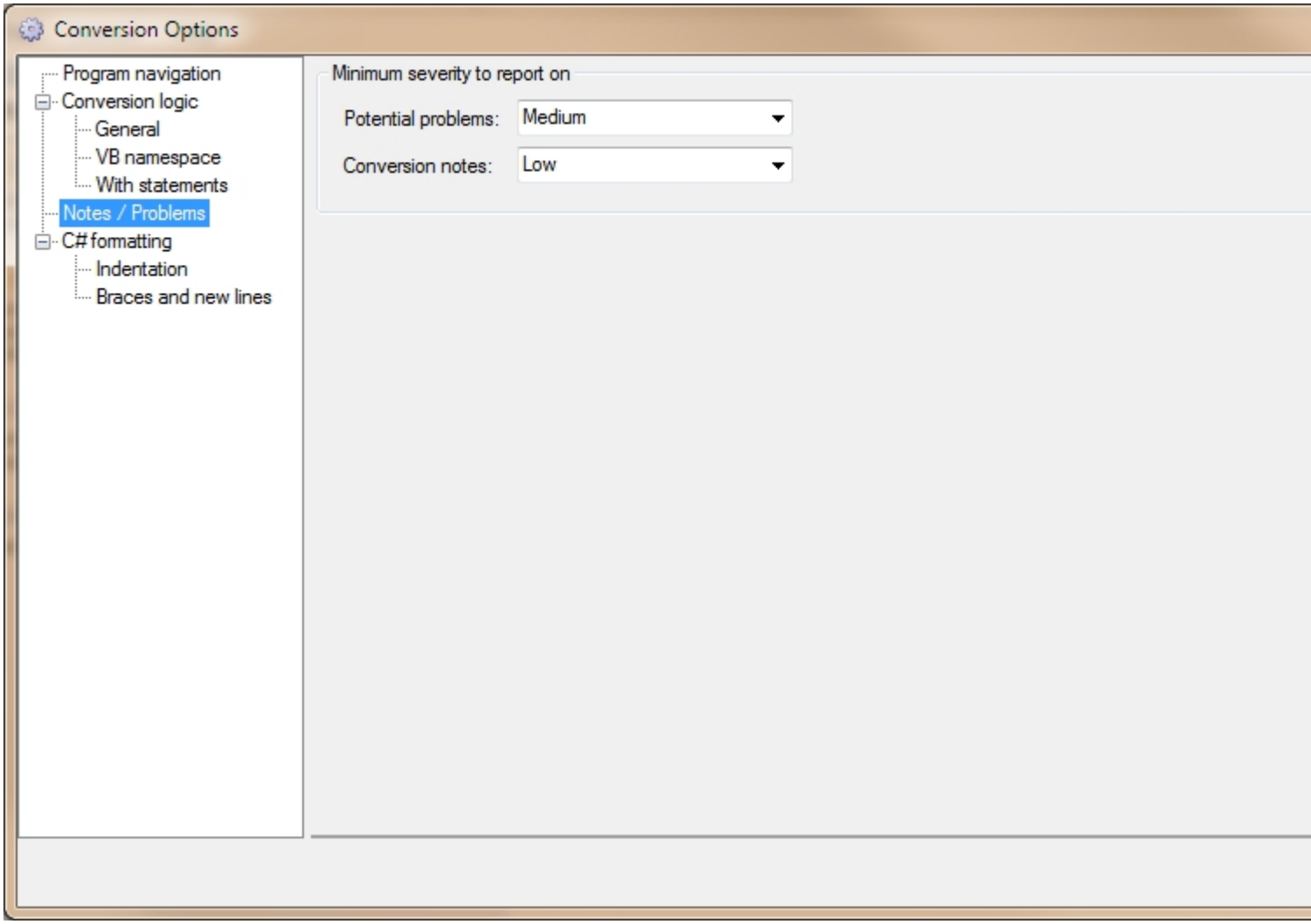
1) The With clause can be repeated for each reference in the with block (the first example below), or

2) Assign the With clause to a temporary variable and reference the temporary variable in the with block. This is essential if the With clause contains side effects (such as adding a record to a database as in the example below).

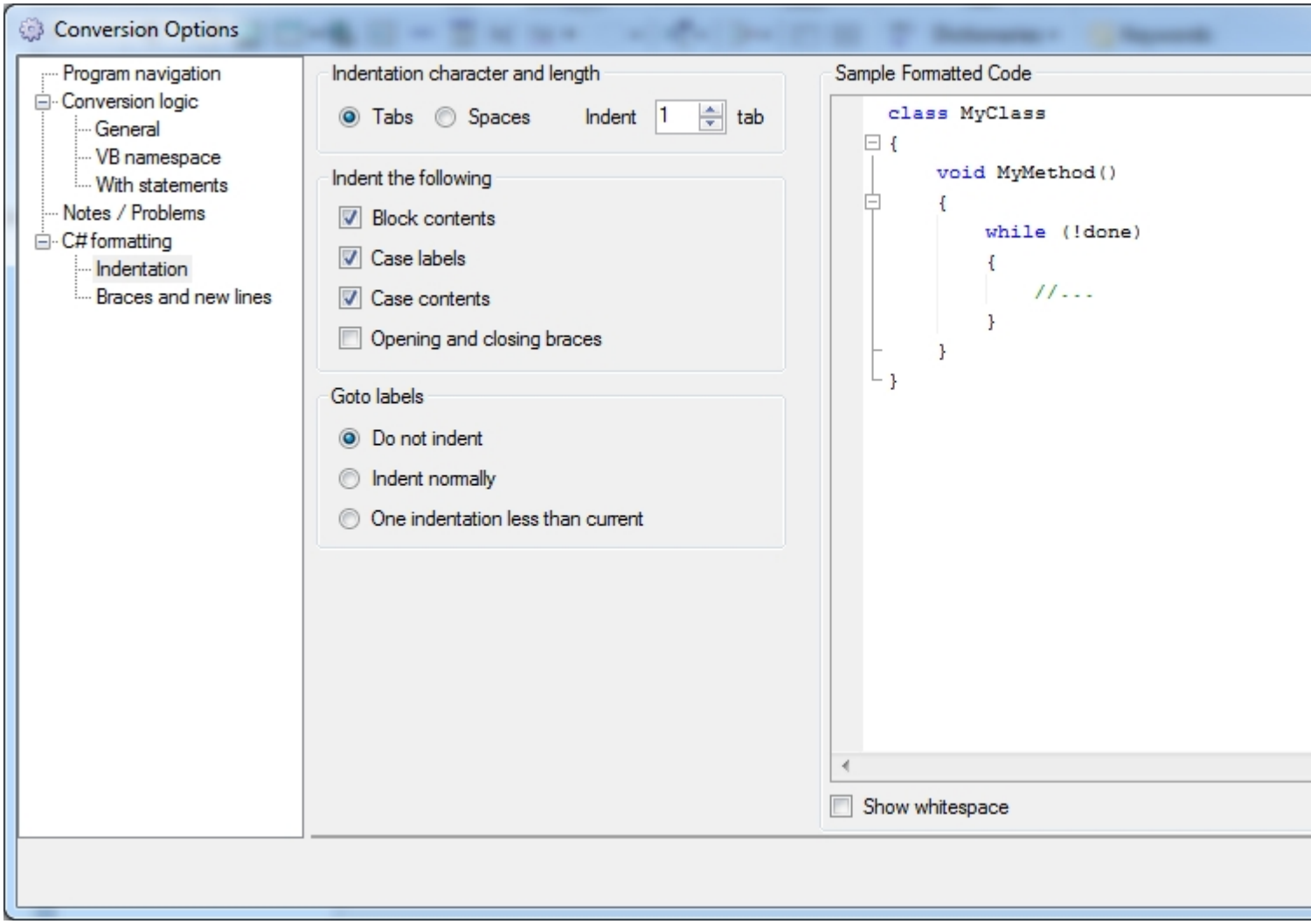
By default, VBConversions will use temporary variables to convert With clauses if a method is being called in the With clause or if it can't determine if a method is being called.



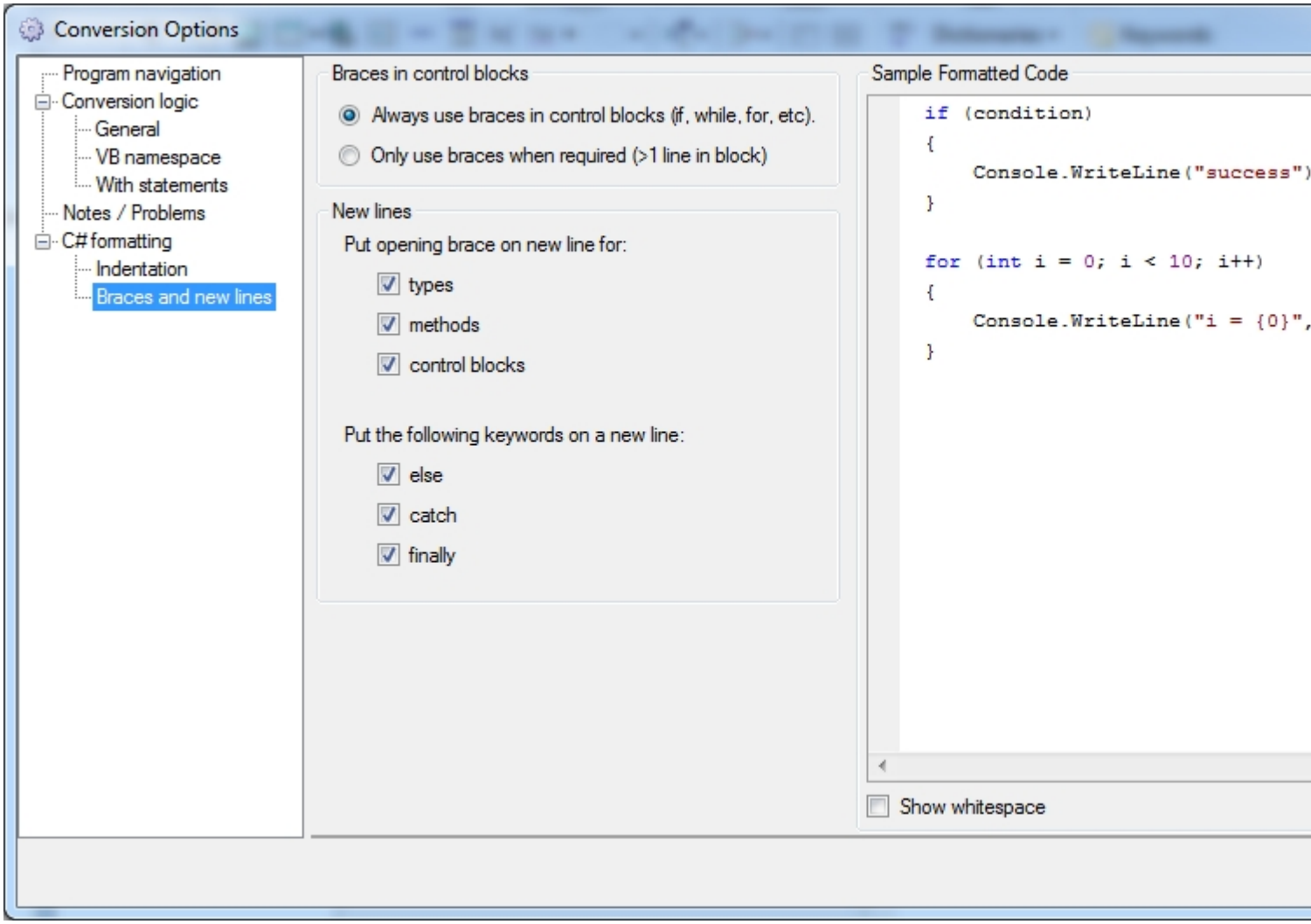
The Notes / Problems options screen allows you to specify the minimum conversion note and potential problem severities to report on. It is not recommended to change either value to High, as you may not be notified of potentially serious conversion issues.



The C# Indentation options screen allows you to control how the C# code output by the converter is indented. As you change the options, you can immediately see how it would affect formatting by viewing the sample C# code in the right hand pane.



The C# Braces and new lines options screen allows you to control how braces and new lines are formatted in the C# code output by the converter. As you change the options, you can immediately see how it would affect formatting by viewing the sample C# code in the right hand pane.



Contacting VBConversions

You may contact VBConversions for support at admin@vbconversions.com

To purchase the VBConversions VB.Net to C# Converter click here