



**OpenEye**  
Scientific Software

## **OpenEye Toolkit QuickStart – C#**

*Release 2012.Feb.1*

**OpenEye Scientific Software, Inc.**

January 11, 2012



# CONTENTS

<b>1</b>	<b>Front Matter</b>	<b>1</b>
<b>2</b>	<b>Installation</b>	<b>3</b>
2.1	Introduction . . . . .	3
2.2	Prerequisites . . . . .	3
2.3	Licensing . . . . .	3
2.4	Download the OpenEye C# distribution . . . . .	4
2.5	Installation . . . . .	4
2.6	The Global Assembly Cache . . . . .	4
<b>3</b>	<b>Using VS2008</b>	<b>5</b>
3.1	Creating a new project . . . . .	5
3.2	Adding references to the OpenEye assemblies . . . . .	6
3.3	Changing the build target . . . . .	6
3.4	Adding code to the project . . . . .	7
	<b>Index</b>	<b>11</b>



# FRONT MATTER

Copyright 1997-2012 OpenEye Scientific Software, Santa Fe, New Mexico. All rights reserved.

All rights reserved. This material contains proprietary information of OpenEye Scientific Software. Use of copyright notice is precautionary only and does not imply publication or disclosure.

The information supplied in this document is believed to be true but no liability is assumed for its use or the infringement of the rights of others resulting from its use. Information in this document is subject to change without notice and does not represent a commitment on the part of OpenEye Scientific Software.

This package is sold/licensed/distributed subject to the condition that it shall not, by way of trade or otherwise, be lent, re-sold, hired out or otherwise circulated without OpenEye Scientific Software's prior consent, in any form of packaging or cover other than that in which it was produced. No part of this manual or accompanying documentation, may be reproduced, stored in a retrieval system on optical or magnetic disk, tape, CD, DVD or other medium, or transmitted in any form or by any means, electronic, mechanical, photocopying recording or otherwise for any purpose other than for the purchaser's personal use without a legal agreement or other written permission granted by OpenEye.

This product should not be used in the planning, construction, maintenance, operation or use of any nuclear facility nor the flight, navigation or communication of aircraft or ground support equipment. OpenEye Scientific Software, shall not be liable, in whole or in part, for any claims arising from such use, including death, bankruptcy or outbreak of war.

Windows is a registered trademark of Microsoft Corporation. Apple, OS X, and Macintosh are registered trademarks of Apple Computer, Inc. AIX and IBM are registered trademarks of International Business Machines Corporation. UNIX is a registered trademark of the Open Group. RedHat is a registered trademark of RedHat, Inc. Linux is a registered trademark of Linus Torvalds. SPARC is a registered trademark of SPARC International Inc.

SYBYL is a registered trademark of TRIPOS, Inc. MDL is a registered trademark and ISIS is a trademark of Accelrys, Inc. SMILES, SMARTS, and SMIRKS may be trademarks of Daylight Chemical Information Systems. Macromodel is a trademark of Schrodinger, Inc.

Python is a trademark of the Python Software Foundation. Java is a trademark or registered trademark of Sun Microsystems, Inc. in the U.S. and other countries.

Other products and software packages referenced in this document are trademarks and registered trademarks of their respective vendors or manufacturers.



# INSTALLATION

## 2.1 Introduction

The C#/.Net version of the OpenEye Toolkits is a P/Invoke wrapper created using SWIG. It is important to note that this is not a new version of the toolkits written in C# nor is it a C# interpretation of the toolkit API, but a rather faithful reproduction of the toolkit API in C#. And while this manual is specific to C#, the version of the toolkits works with all .Net languages include Visual Basic .Net, F# and IronPython.

A few of the idiomatic differences include:

- There are relatively few classes in OEChem and they have a rather shallow inheritance hierarchy.
- Most OEChem algorithms are in C++ free functions. These are mapped to static methods in the class *OEChem*.
- C# doesn't support all the operator overloading of C++, so some C++ operators are mapped to methods. For example, "operator bool()" is mapped to a member function called "IsValid()", "operator()" is mapped to "Call()".

## 2.2 Prerequisites

In order to use the OpenEye toolkits with C#, you need a copy of Visual Studio. We currently support VS2008 and VS2010, including the free "Express" versions.

## 2.3 Licensing

A license file from OpenEye Scientific Software is required to run any OpenEye application. A license file can be requested/obtained by contacting OpenEye at [business@eyesopen.com](mailto:business@eyesopen.com).

At startup, the application looks for a valid license in the following default locations:

- In a file specified by the environment variable **OE\_LICENSE**.
- In a file named `oe_license.txt` in the directory specified by the environment variable **OE\_DIR**.
- In a file named `oe_license.txt` in the user's platform-specific local OpenEye application data directory. The location of this directory is detailed below:
  - Linux/UNIX:  
~USERNAME/.OpenEye
  - Mac OS X:  
/Users/USERNAME/Library/OpenEye

- Microsoft Windows 2000/XP

C:\Documents and Settings\USERNAME\Application Data\OpenEye

- Microsoft Windows Vista/7

C:\Users\USERNAME\AppData\Local\OpenEye

- In a file named `oe_license.txt` in the current working directory

## 2.4 Download the OpenEye C# distribution

The distribution is obtained from <http://www.eyesopen.com/downloads>. Once you have a license file with the “clr” feature for the toolkits, you can download a distribution. Picking a distribution is relatively easy. There are two 32-bit downloads, one for VS2008 and one for VS2010.

64-bit versions are provided, but these will only run on 64-bit versions of Windows 7, Windows Vista and Windows Server.

## 2.5 Installation

Each distribution is provided as a Windows installer. After running the installer, on the Start menu you will find a link to the documentation in HTML and PDF format. There is also a link to take you directly to the Examples folder. Inside the Examples folder, you will find a pair of folders for each toolkit. One for the main examples and one for the documentation examples. Each sub-folder contains a solution (.sln) file appropriate for your chosen Visual Studio version.

**Note:** The installer does not install anything to the Global Assembly Cache (GAC). There is no requirement for having the DLLs installed in the GAC, but just in case, see the section *Global Assembly Cache*.

**Note:** You can install more than one version of the OpenEye .Net toolkits. So if you have both VS2008 and VS2010 or if you want both 32-bit and 64-bit versions, they can co-exist with no problems.

## 2.6 The Global Assembly Cache

If you desire to install all the OpenEye assemblies into the Global Assembly Cache, there is a convenience script included in C:\OpenEye\OpenEye.Net<VERSION>\Scripts. Double-click on **Install\_to\_GAC.bat** to install all the assemblies.

If you need to uninstall the assemblies for some reason, simply double-click on **Uninstall\_from\_GAC.bat**.

# USING VS2008

## 3.1 Creating a new project

To create a new project, choose from the File menu, “File->New->Project...”. This will bring up a dialog like *New VS2008 Project 1*. There are several settings we want to choose here:

1. Under “Project Types:”, open the “Visual C#” node and choose “Windows”.
2. In the “Templates” section, choose “Console Application”.
3. In the upper right-hand corner, choose “.Net Framework 3.5” from the pull-down menu.
4. Type a name for your project, like “OESample1”.
5. Click “OK”

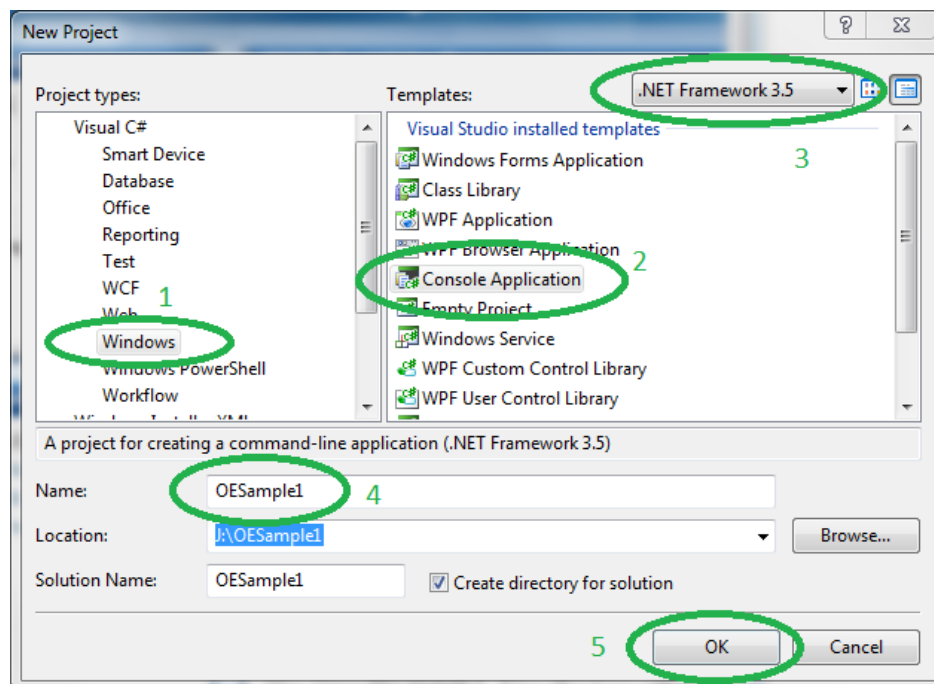


Figure 3.1: New VS2008 Project 1

## 3.2 Adding references to the OpenEye assemblies

In order to link in the OpenEye libraries, we need to add a reference to the project to any/all of the OpenEye assemblies we will be using.

In the Solution Explorer, right-click on the “References” node inside the “OESample1” project, and choose “Add Reference...”. This will bring up a dialog like *Add Reference*.

In this dialog, we want to:

- Click the “Browse” tab at the top.
- Browse to the directory where you installed the OpenEye .Net libs. By default, this will be “C:\OpenEye\OpenEye.Net<VERSION>\Libs”, where version is the specific version/architecture you are building with.
- Scroll down and select “OpenEye.OEChem.dll” and hit “OK”

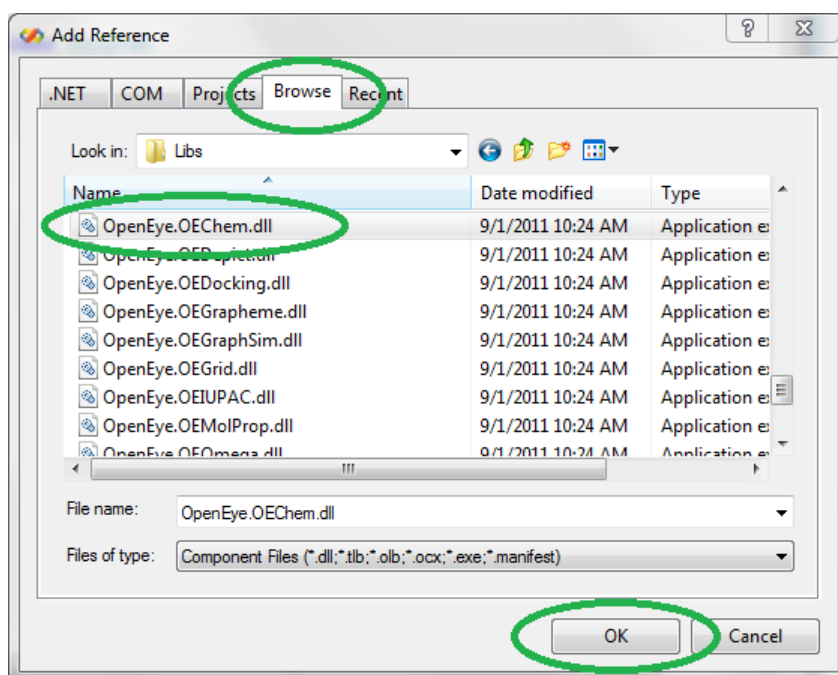


Figure 3.2: AddReference

Additional toolkits can be chosen as you add more functionality to your program. For this example, we are only using OEChem.

## 3.3 Changing the build target

By default, a new C# project is set up to target “AnyCPU”, but since we are linking in native code, we need to change the build target to either “x86” or “x64”, depending on which version of the OpenEye toolkits you are using. For this we will assume we are using the x86 version.

In the menu bar, you will see two pull-down menus that control the build target. One lets you choose between Debug and Release builds while the other switches the platform (architecture).

We need to create an “x86” platform to replace the “AnyCPU” target.

Click on the drop down and choose “Configuration Manager...”

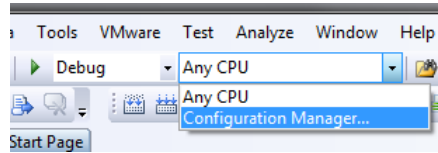


Figure 3.3: Launching the Configuration Manager

You should see a dialog like *The Configuration Manager*.

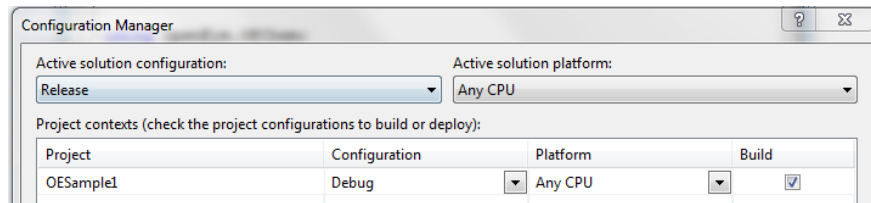


Figure 3.4: The Configuration Manager

In the upper right corner of the Configuration Manager Dialog, click on the “Active solution platform:” pulldown and choose “<New...>”.

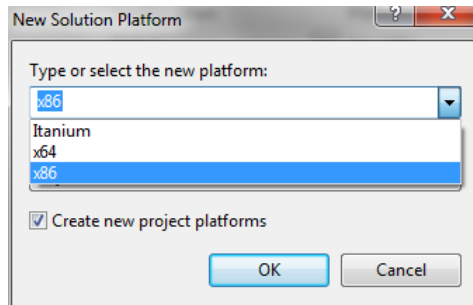


Figure 3.5: Creating new solution platform

In the top pull-down, choose “x86” and leave the bottom pull-down (“Copy settings from”), set to “AnyCPU”. Click “OK.” See *Creating new solution platform*.

This should leave you with the Configuration Manager window looking something like *x86 platform created*

Click “Close”. Now it is time to add some code to our program.

## 3.4 Adding code to the project

At the top of the main Visual Studio window, you should see your new platform choice in place.

In the project browser, double-click “Program.cs” to open it in the editor, if it is not already open.

At the top of the file, we will add *using* statements for OEChem:

```
using OpenEye.OEChem;
```

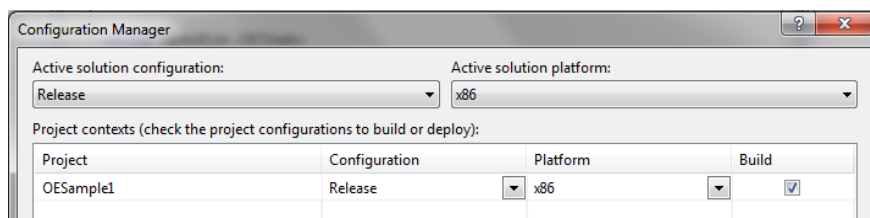


Figure 3.6: x86 platform created

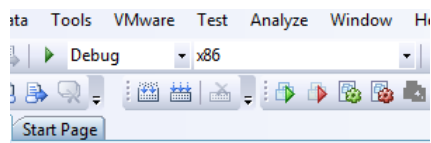


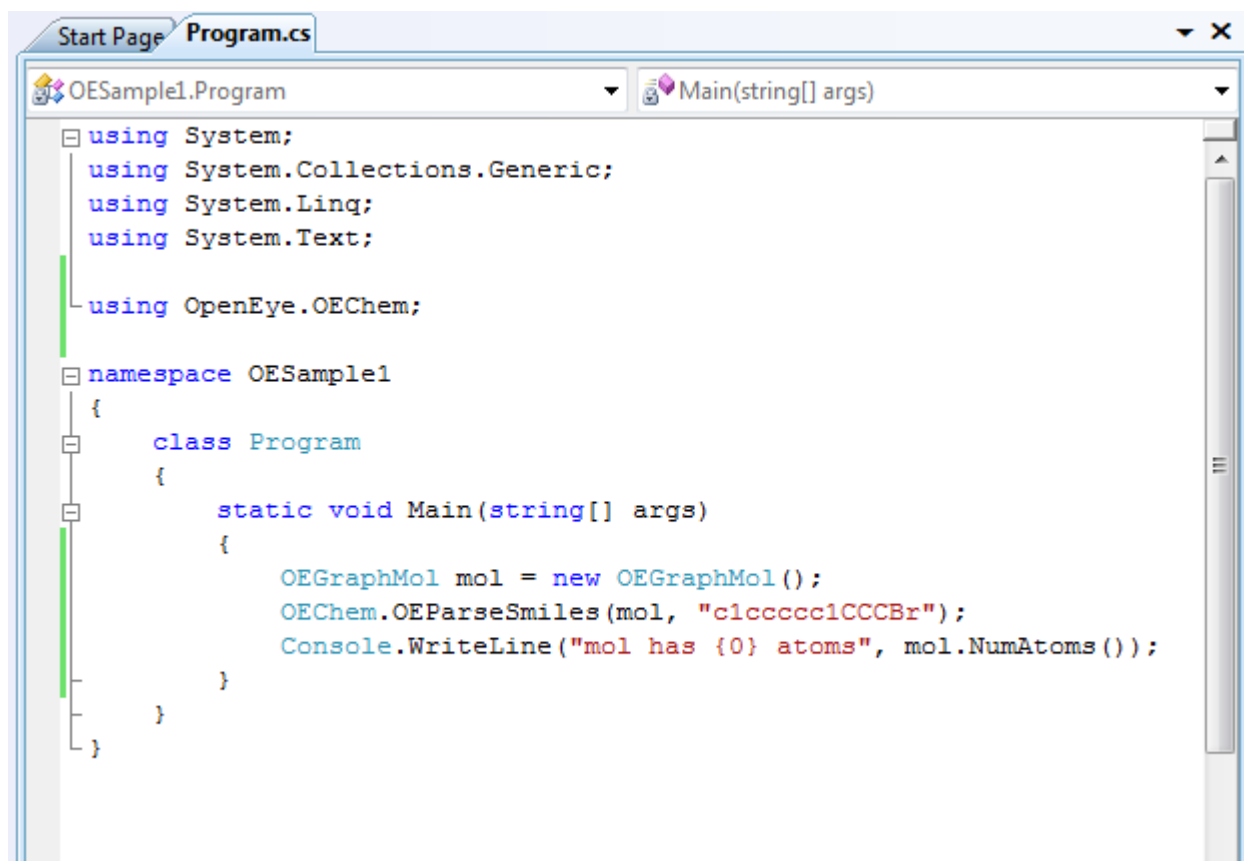
Figure 3.7: Debug|x86 chosen

and then inside the “Main”, we add this code:

```
OEGraphMol mol = new OEGraphMol();  
OEChem.OEParseSmiles(mol, "c1ccccc1CCBr");  
Console.WriteLine("mol has {0} atoms", mol.NumAtoms());
```

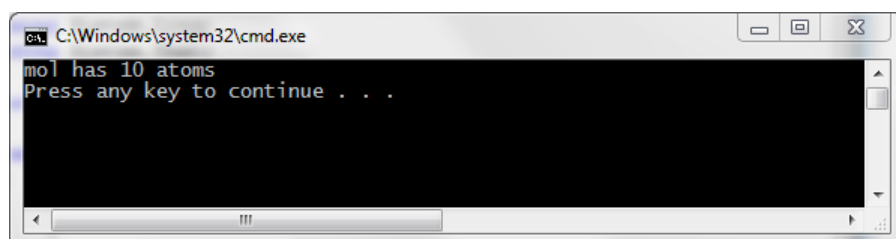
When you are finished, your sample should look something like *Editing in VS2008*.

To test our example, we can run it from within VS2008 by hitting <Ctrl-F5>. If you don't have any syntax errors, you should see a Console window appear with output that looks like:



```
Start Page Program.cs
OESample1.Program Main(string[] args)
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using OpenEye.OEChem;
namespace OESample1
{
    class Program
    {
        static void Main(string[] args)
        {
            OEGraphMol mol = new OEGraphMol ();
            OEChem.OEParseSmiles (mol, "c1cccc1CCBr");
            Console.WriteLine ("mol has {0} atoms", mol.NumAtoms ());
        }
    }
}
```

Figure 3.8: Editing in VS2008



```
C:\Windows\system32\cmd.exe
mol has 10 atoms
Press any key to continue . . .
```

Figure 3.9: Running the example in VS2008



# INDEX

## E

environment variable  
    OE\_DIR, 3  
    OE\_LICENSE, 3

## O

OE\_DIR, 3  
OE\_LICENSE, 3