



White Paper

**Jennifer L. Jiang**

Compiler Technical Consulting  
and Support Team

Intel Corporation

# Intel® C++ Compiler for Windows\* Compatibility with Versions of Microsoft Visual C++\*



# Introduction

The Intel® C++ Compiler 9.1 for Windows\* supports the Microsoft\* Visual C++\* extensions to the C and C++ languages. This document describes the important differences of the Intel C++ Compiler 9.1 and Visual C++ extensions.

## Overview

The Intel® C++ Compiler 9.1 for Windows is integrated into the following Microsoft development environments: Microsoft Visual C++\* 6.0, Microsoft Visual C++ .NET\* 2002, Microsoft Visual C++ .NET 2003, and Microsoft Visual C++ 2005. The Intel C++ Compiler also generates code that is optimized for IA-32, Intel® processors with Intel® Extended Memory 64 Technology (Intel® EM64T), and Intel® Itanium® 2 processors.

The Intel C++ Compiler is source- and binary- (native code) compatible with the Microsoft Visual C++ Compiler. This allows you to harness the performance-enhancement features of the Intel C++ Compiler by rebuilding only part of an application with the Intel Compiler. If required, you can mix and match object files and libraries built with the two compilers.

This document explains key processes for obtaining information on the compatibility between the Intel C++ Compiler and Microsoft Visual C++ 6.0, Microsoft Visual C++ .NET 2002 and 2003, and Visual C++ 2005.

## General Compatibility

### Compatibility Options

The following Intel® Compiler options provide compatibility for different versions of Microsoft Visual C++ on IA-32 systems:

- `/Qvc6` Microsoft Visual C++ 6.0
- `/Qvc7` Microsoft Visual C++ .NET 2002
- `/Qvc7.1` Microsoft Visual C++ .NET 2003
- `/Qvc8` Microsoft Visual C++ 2005

In a few cases, the Microsoft Compiler compiles source code without generating any errors while the Intel Compiler generates errors for the same source code. To compile without errors, use the `/Qms[0,1,2]` option with the Intel Compiler. The default setting for the `/Qms` option is `/Qms1`. The meanings for `/Qms [0, 1, 2]` follow:

- `/Qms0` instructs the compiler to disable Microsoft compatibility bugs.
- `/Qms1` instructs the compiler to enable most Microsoft compatibility bugs (default).
- `/Qms2` instructs the compiler to implement full Microsoft compatibility.

The macro has been predefined in Intel® C++ Compiler for Windows 8.1 or later.

`__INTEL_MS_COMPAT_LEVEL__` specifies the level of Microsoft compatibility provided. The value of this macro is controlled by the setting of the `-Qms` option. If no `/Qms` option is provided, the value of this macro is 1 (the default setting).

### Microsoft Visual C++\* Pragmas Support

The Intel C++ Compiler supports the following pragmas:

- `#pragma auto_inline([on|off])`  
Specifying off excludes any functions within the range as candidates for automatic inline expansion.
- `#pragma conform(name [, show] [, {on|off}] [[, {push|pop}] [, identifier]])`

Specifies the run-time behavior of the `/Zc:forScope` compiler option, which allows for the loop's initializer to remain in scope until the local scope ends.

The Intel Compiler accepts the following pragmas, which sometimes appear in header files. Though they do not result in error messages, the pragmas have no effect.

- `component`
- `function`
- `include_alias`
- `inline_depth`
- `inline_recursion`
- `intrinsic`
- `setlocale`

**Differences in Treatment of Inline Assembly Labels**

The Intel C++ Compiler treats the inline assembly labels in a case-sensitive manner. In contrast, the Microsoft Visual C++ Compiler treats them in a case-insensitive manner.

**Differences with Macro Expansions**

The two compilers act differently when code passes a macro as a parameter to another macro using the token concatenation operator to generate new code.

The Intel C++ Compiler does not expand the macro used as a parameter before concatenation. Nor does it expand macros passed to `#include` directives.

The Microsoft Visual C++ Compiler performs an extra preprocessing scan to expand those macros.

**Differences in Precompiled Header (PCH) Support**

The precompiled header (PCH) information generated by the Intel C++ Compiler is not compatible with the PCH information generated by the Microsoft Visual C++ compiler. The specific incompatibilities are as follows:

- The Intel Compiler does not recognize the PCH file generated by the Microsoft compiler. If no suitable or recognized PCH file exists, compilation proceeds without use of PCH files.
- The Microsoft compiler aborts with an error message when it tries to use a PCH file generated by the Intel compiler.
- The Intel C++ Compiler does not support PCH generation and use in the same translation unit.
- The Intel C++ Compiler does not generate PCH information beyond the point where a declaration is seen in the primary translation unit. When the `/Yu` option is specified, the Microsoft Visual C++ compiler ignores all text, including declarations preceding the `#include` statement of the specified file.
- PCH files coexistence: the `/Qpch` option causes the Intel C++ Compiler to name its PCH files with a `.pch` filename suffix and reduce build time. The `/Qpch` option is on by default; use `/Qpch-` to turn it off.

**Differences in enum type**

Microsoft Visual C++ always considers `enum` bit fields to be signed, even if not all values of the `enum` type can be represented by the bit field. On the other hand, the Intel C++ Compiler considers an `enum` bit field to be unsigned, unless the `enum` type has at least one `enum` constant with a negative value.

## Microsoft Visual C++ 6.0 Compatibility

The Intel C++ Compiler is fully source- and binary-compatible with the Microsoft Visual C++ 6.0 Compiler when `"/Qvc6"` is specified. You can build either the full project files or part of the project files with the Intel C++ Compiler.

**Integration with Visual C++ 6.0 IDE**

The Intel C++ Compiler is integrated into Microsoft Visual C++ 6.0 IDE through the Intel® C++ Compiler Selection Tool that is available from the [Tools] > [Intel (R) C++ Compiler Selection Tool] menu. Binaries built with the Intel C++ Compiler can be debugged from within the Microsoft Visual C++ 6.0 IDE. From Microsoft Visual C++ 6.0 IDE, programs can be built by the Intel C++ Compiler to target either IA-32-based systems or Itanium®-based systems.

**Visual C++ 6.0 Processor Pack Compatibility**

If your program uses the Single Instruction Multiple Data (SIMD) C++ data types, they may be affected by the recent changes in the binary conventions for compatibility with the Microsoft Visual C++ 6.0 Processor Pack. These data types include:

- `__m64`
- `__m128`
- `__m128d`
- `__m128i`

The `/Qmssp[-]` option, available in the Intel C++ Compiler for IA-32 system, is ON by default. It enables binary compatibility with Microsoft Visual C++ 6.0 Processor Pack.

**Intel® Itanium®-based Processor Targeting**

The Intel C++ Compiler has been integrated into Microsoft Visual Studio\* 6.0 IDE to develop applications for Itanium-based systems using the Intel C++ Compiler Selection Tool. The latest Platform SDK or Visual Studio 2005 Team System is required for developing the Itanium-based application.

You can also develop Itanium-based applications from the command window with the Intel C++ Compiler.

Options	Description
<b>/Fd</b>	Name of the PDB file used for debug information for specified source files
<b>/Gi</b>	Enables incremental compilation
<b>/ZI</b>	Edits and continues debugging (similar effect as the /Zi option)
<b>/Gm</b>	Enables minimal rebuild
<b>/Yd</b>	A Microsoft precompiled header-specific option that puts debug information in every object
<b>/Zmn</b>	Controls maximum memory allocated by the compiler

Table 1. Unsupported Compiler Options

### Unsupported Compiler Options

The Intel C++ Compiler supports most of the options of the Microsoft Visual C++ 6.0 Compiler. However, it does not support the options listed in the table below. Most of the unsupported options are useful during development time and are typically not required to build a working application.

### Unsupported Pragmas

Refer to the earlier discussion in this paper, Microsoft Visual C++\* pragmas support.

## Microsoft Visual C++ .NET\* and Visual C++ 2005 Compatibility

The Intel C++ Compiler is fully source- and binary- (native code only) compatible with the Microsoft Visual C++ .NET 2002/2003 compiler and the Visual C++ 2005 compiler when the option `"/Qvc7"` or `"/Qvc7.1"` or `"/Qvc8"` is specified. Binaries built with the Intel C++ Compiler can be debugged from within the Microsoft Visual C++ .NET IDE or Visual Studio 2005 IDE.

With regard to security checks, `/GS` is supported by the Intel C++ Compiler, but it is disabled by default; in contrast, in Visual C++ 2005, it is enabled by default. Please specify `/GS` to enable the security checks with the Intel C++ Compiler. The Intel C++ Compiler 9.1 for Windows also added the support for safe exception handler feature with `/Qsafeseh` for 32-bit binary and is on by default.

Manifest is a Visual Studio 2005 feature that describes run-time dependencies of an application. The Intel C++ Compiler supports generation of manifest files, which are typically placed alongside the EXE or DLL with a file type of `.manifest`. Manifest files can also be embedded in the EXE or DLL, but this is supported only when incremental linking is disabled.

### Integration with Visual C++ .NET and Visual Studio\* 2005 IDEs

The Intel C++ Compiler 9.1 for Windows is integrated into Microsoft Visual C++ .NET 2002 and 2003, and Visual Studio 2005 IDEs through the Visual Studio Integration Partner (VSIP).

Microsoft Visual C++ .NET 2002/2003 and Visual Studio 2005 maintain a `.vcproj` file for the Visual C++ project, and the Intel C++ project system maintains a corresponding file `.icproj` that contains the configuration data specific to the Intel C++ Compiler. In order to use the Intel C++ Compiler, you need to convert your project or solution to use the Intel C++ project system first. After converting, the new file `.icproj` will be created for every `.vcproj` project.

There are two ways to convert a Microsoft Visual C++ .NET or Visual C++ 2005 project or solution to use the Intel C++ project system:

- From the Intel C++ Compiler's command window, use `"ICProjConvert90.exe"`.
- From the Microsoft Visual C++ .NET or 2005 IDE, right-click on a project or multiple projects or the solution and select the menu item `"Convert to use Intel C++ Project System"` from the pop-up menu.

If you'd like to share the `.vcproj` file among your development group, but not everyone in your group has the Intel C++ Compiler installed, you should specify `/sharevcproj` when using `ICProjConvert90.exe`. If you convert within the Microsoft Visual C++ .NET or 2005 IDE, you should make sure the field `"Share Visual C++ project files"` is set to `"Yes"` on the [Tool] > [Options] > [Intel® C++] > [General page]. This field is set to `"Yes"` by default. Please note that the `.sln` file can not be shared because it is modified when converting to Intel C++ Project.

Options	Description
<code>/AI&lt;dir&gt;</code>	Adds to assembly search path
<code>/clr</code>	Compiles for the common language runtime (managed C++)
<code>/FU&lt;file&gt;</code>	Forces use of assembly/module
<code>/Fx</code>	Merges injected code to file
<code>/MPn</code>	Utilizes n processors for compilation
<code>/openmp</code>	Use /Qopenmp instead
<code>/w&lt;l&gt;&lt;n&gt;</code>	Sets warning level 1-4 for n
<code>/wo&lt;n&gt;</code>	Issues warning n once
<code>/Zm&lt;n&gt;</code>	Maximum memory allocation (percentage of the default)
<code>/homeparams</code>	Force parameters passed in registers to be written to the stack
<code>/doc [file]</code>	Process XML documentation comments and optionally name the .xdc file

**Table 2. Unsupported Compiler Options of Visual C++ .NET and Visual C++ 2005**

The following capabilities are provided by the integration:

- Converting new or existing Visual C++ projects or solutions to use the Intel C++ Project System
- Selecting either the Intel C++ Compiler or the Microsoft Visual C++ Compiler to build each configuration of a project or individual files of a project
- Optimizing your application using compilation options specific to the Intel C++ Compiler
- Choosing which version of the Intel C++ Compiler to use if multiple versions are installed on your system

The Intel C++ Compiler only supports native C++ project types provided by Visual C++ .NET and Visual Studio 2005. The project types with .NET attributes such as the ones below cannot be converted to an Intel C++ project:

- Empty Project (.NET)
- Class Library (.NET)
- Console Application (.NET)
- Windows Control Library (.NET)
- Windows Forms Application (.NET)
- Windows Service (.NET)

The Intel C++ Compiler 9.1 for Windows does not support all the features of Microsoft Visual C++ .NET 2002/2003 and Visual C++ 2005. The details are discussed below.

## A New Pragma, Macros and Options to Increase Compatibilities

### New Pragma

The Intel C++ Compiler supports this new pragma from Microsoft Visual C++ .NET:

```
#pragma conform(name [, show]
[, {on|off}] [[, {push|pop}] [,
indentifier]])
```

This pragma specifies the run-time behavior of the `/Zc:for` the Scope compiler option to allow for the loop's initializer to remain in scope until the local scope ends.

### New Macros

The `__COUNTER__` and `__FUNCSIG__` macros have been predefined in the Intel C++ Compiler 8.1 for Windows or later..

- `__COUNTER__` expands to an integer, starting with 0 and incrementing by 1 every time it is used. `__COUNTER__` remembers its state when using precompiled headers. If the last `__COUNTER__` value was 4 after building a precompiled header (PCH), it will start with 5 on each PCH use
- `__FUNCSIG__` is valid only within a function and returns the signature of the enclosing function (as a string). `__FUNCSIG__` is not expanded if you use the `/EP` or `/P` compiler options.

## Unsupported Major Features

- Attributed code
- Managed extensions for C++ (new pragmas, keywords, and command-line options)
- Event handling (new keywords)

- `__abstract` keyword
- `__box` keyword
- `__delegate` keyword
- `__gc` keyword
- `__identifier` keyword
- `__nogc` keyword
- `__pin` keyword
- `__property` keyword
- `__sealed` keyword
- `__try_cast` keyword
- `__w64` keyword

### Unsupported Preprocessor Features

- `#import` directive changes for attributed code
- `#using` directive
- `managed`, `unmanaged` pragmas
- `_MANAGED` macro
- `runtime_checks` pragma

### Mixing Unmanaged Code Compiled with the Intel® C++ Compiler with Managed Code

If you use the managed extensions to the C++ language in Microsoft Visual C++ .NET, you can use the Intel C++ Compiler for your non-managed code for better application performance. Make sure managed keywords do not appear in your non-managed code.

For information on how to mix unmanaged code and managed code, refer to the article, "An Overview of Managed/Unmanaged Code Interoperability", at the Microsoft Web site: <http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dndotnet/html/manunmancode.asp>.

**Note:** This URL is available at the Microsoft MSDN\* Web site. If it changes, please search for the article.

You can find more information on the Intel C++ Forum at <http://softwareforums.intel.com/ids>.

### Intel® EM64T and Itanium-based Processor Targeting

The Intel C++ Compiler for Windows 9.1 now supports targeting for Intel EM64T and Itanium-based systems from within the Visual Studio 2005 IDE, but you can develop for Intel EM64T and Itanium-based applications from the command window as well.

#### Note:

- To build for Intel EM64T within Visual Studio 2005 IDE, you need to install the Visual Studio 2005 Standard edition or above, and should have the "X64 Compiler and Tools" installed. Also you should use `/favor:EM64T` for best performance on Intel EM64T when using Visual C++ 2005, use `/QxP` when using the Intel C++ Compiler for Intel EM64T.

Some nice features offered by the Intel C++ Compiler version 9.x are that it supports inline asm, MMX/x87 instructions in the inline asm, and MMX™ instructions through intrinsics.

- To build Itanium-based applications within the Visual Studio 2005 IDE, you need to install the Visual Studio 2005 Team System, and you should have the "Itanium Compiler and Tools" component installed.

The Intel C++ Compiler for Windows 9.1 does not support targeting for Intel EM64T and Itanium-based systems from within the Visual C++ .NET 2002 or 2003 IDE.

# Compatibility in OpenMP\* Support

Visual C++ 2005 now supports the OpenMP\* 2.0 API specification. The Intel C++ Compiler for Windows has been supporting OpenMP since version 7.x. The Intel C++ Compiler now supports the OpenMP 2.5 API specification.

The following list briefly summarizes OpenMP support in the Intel C++ Compiler for Windows:

1. OpenMP 2.5 API specification
2. Taskqueuing extension

**Note:** The taskqueuing extension implemented by the Intel C++ compiler extends OpenMP to parallelize a broader range of applications. It allows the user to exploit irregular parallelism with units of work that are not pre-computed at the start of the worksharing construct. Unlike `single`, `for`, and `sections` constructs, all work units are known at the time the construct begins execution.

The taskqueuing pragmas are `taskq` and `task`. The `taskq` pragma specifies the environment within which the enclosed units of work (`tasks`) are to be executed. From among all the threads that encounter a `taskq` pragma, one is chosen to execute it initially. The `task` pragma specifies a unit of work, potentially executed by a different thread.

3. Implemented extension functions to the OpenMP run-time and implemented extension environment variables
4. Support for debugging OpenMP code

5. Provides an OpenMP stub library for debugging purposes that links in the OpenMP stub APIs but still runs sequentially; this is supported through `/Qopenmp_stub` option

6. Support for the Intel® Thread Checker and Intel® Thread Profiler via `/Qtcheck` and `/Qopenmp_profile`

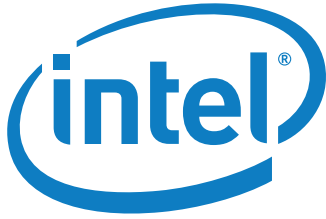
Because the OpenMP specification does not define the interoperability of multiple implementations, the implementations of the OpenMP by Visual C++ 2005 and Intel C++ Compiler for Windows are not interoperable. To avoid possible linking and run time problems, please follow the rules below:

1. Avoid using multiple copies of OpenMP runtime library from different compilers. Because each OpenMP runtime library assumes it has all the resources of the system, it ends up with oversubscription of threads to processors.
2. The best choice is to compile all the OpenMP sources with one compiler, or at least the parallel region and entire call tree under it. The routines exposed to users should not have orphaned OpenMP constructs.
3. Use dynamic library for OpenMP.

## References

You can find useful information about compiler compatibility in the following documents, each of which is available from the Intel C++ Compiler 9.1 for Windows web site: <http://www.intel.com/cd/software/products/asm-na/eng/compilers/cwin/index.htm>

- Intel® C++ Compiler 9.1 for Windows Release Notes
- Intel® C++ Compiler 9.1 for Windows User's Guide
- Optimizing Applications with the Intel® C++ and Fortran Compilers for Windows and Linux



For product and purchase information visit:  
[www.intel.com/software/products](http://www.intel.com/software/products)

Intel, the Intel logo, Intel. Leap ahead. and Intel. Leap ahead. logo, Pentium, Intel Core, and Itanium are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

\*Other names and brands may be claimed as the property of others.

INFORMATION IN THIS DOCUMENT IS PROVIDED IN CONNECTION WITH INTEL PRODUCTS. NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS DOCUMENT. EXCEPT AS PROVIDED IN INTEL'S TERMS AND CONDITIONS OF SALE FOR SUCH PRODUCTS, INTEL ASSUMES NO LIABILITY WHATSOEVER, AND INTEL DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY, RELATING TO SALE AND/OR USE OF INTEL PRODUCTS INCLUDING LIABILITY OR WARRANTIES RELATING TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT. Intel products are not intended for use in medical, life saving, life sustaining applications. Intel may make changes to specifications and product descriptions at any time, without notice.

Copyright © 2006, Intel Corporation. All Rights Reserved.

0505/DXM/ITF/PDF 300348-002