Intel® C++ Compiler 8.1 for Windows Release Notes*

For Intel® IA-32 and Itanium® Processors

Contents

Overview

This product provides tools for Windows* software developers to create applications that run at top speeds on all Intel® IA-32 processors and the Intel® Itanium® processors. Optimizations include support for Streaming SIMD Extensions 2 (SSE2) in the Intel® Pentium® 4 and Pentium M processors, Streaming SIMD Extensions 3 (SSE3) in the Intel Pentium 4 processors with SSE3 support, and software pipelining in the Intel Itanium 2 processor. Inter-procedural optimization (IPO) and profile-guided optimization (PGO) can provide greater application performance. Intel Compilers support multi-threaded code development through auto parallelism and OpenMP* support.

Tools provided include the Intel® C++ Compiler 8.1 for Windows, the Intel® Debugger 8.1, the Assembler 7.0 for Itanium-based systems and integration with Microsoft Visual Studio* .NET* 2002 & 2003 and Microsoft Visual C++* 6 (Visual Studio 98). This version of the compiler also includes two new tools, the Intel Compilers code-coverage Tool and the Intel Compilers test-prioritization tool.

There are three variations of the Intel C++ Compiler for Windows.

- One compiler runs on IA-32 systems and produces applications that run on IA-32 systems.
- A second compiler runs on IA-32 systems but produces Itanium-based applications.
- The third compiler runs on Itanium-based systems and produces Itanium-based applications.

The Intel Debugger runs on IA-32 systems and Itanium-based systems as a 32-bit application. It debugs both IA-32 applications and Itanium-based applications. If you produce Itanium-based applications you can run and debug them on an Itanium-based system, or run them on an Itanium-based system while debugging them from an IA-32 development system using the Intel Debugger's remote debugging capability.

The Intel C++ Compiler provides integration into Microsoft's Integrated Development Environment (IDE) provided with Microsoft Visual C++* .NET 2002 and 2003 versions and Microsoft Visual C++ 6, on IA-32 systems. You should expect differences among the binaries generated in the different Microsoft Visual C++ environments. The Intel C++ Compiler generates binaries to target each of these versions with the following command line options: /Qvc6, `/Qvc7` and `/Qvc7.1`.

Product Contents

This product contains the following components:

- Intel® C++ Compiler for 32-bit applications, version 8.1
- Intel C++ Compiler for Itanium-based applications, version 8.1
- Intel C++ Itanium Compiler for Itanium-based applications, version 8.1
- Intel IA-32 Assembler 7.0 to produce Itanium-based applications
- Intel Itanium-based Assembler 7.0 to produce Itanium-based applications
- Intel Debugger 8.1
- Utilities
    - Intel Compilers code-coverage tool
    - Intel Compilers test-prioritization tool
    - Integration into Microsoft's Visual Studio .NET 2002 and 2003 on IA-32 systems
    - Compiler Selection Tool for Microsoft Visual C++ 6.0 (Visual Studio 98) on IA-32 systems
    - Utility icpi to isolate compile/link time errors located at: `<installation_directory>\Compiler80\ia32\bin\icpi.exe` or at `<installation_directory>\Compiler80\Itanium\bin\icpi.exe`
- Documentation and documentation index can be found at `<installation_directory>\Compiler80\docs\ccompindex.htm`.
    - A training tutorial *Enhancing Performance with Intel Compilers* is provided

Note: The `<installation_directory>` defaults to `<Program Files>\Intel\cpp\` . The Version 8.1 compiler installs into the `compiler80` subdirectory, replacing version 8.0.

To receive technical support and product updates for the tools provided in this product you need to register. For how to register, please see Technical Support section below.

What's New in Version 8.1

This section discusses new features and changes in the 8.1 release of the Intel® C++ compiler.

New Predefined Macros

The following new predefined macros are now available:

- `__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).
- `__INTEL_COMPILER_BUILD_DATE` specifies the build date of the compiler in YYYYMMDD format. It matches the build date shown on the version banner. You can use this predefined macro if you have a need to conditionalize code based on a specific Intel Compiler update. The YYYYMMDD string is guaranteed to be an increasing integral value with each new release.

/O3 enables high-level loop and memory optimizations

In this release, specifying -O3 enables additional loop transformations (interchange, distribution, collapsing) and memory access optimizations which can improve performance.

/Qipo Intermediate Language Now Contained Within Object Files

In previous releases, when /Qipo was used, separate .il files were generated to contain intermediate language. The use of these separate files could cause difficulty building with makefiles. In this version, the intermediate language is embedded in the .obj files and no separate .il file is created

New /Qipo*N* Option to Create Multiple Objects

In previous versions, when /Qipo was specified to perform multifile interprocedural optimization, one object file was generated as input to the linker; this is still the default for version 8.1. New in version 8.1 is the ability to request that the compiler create multiple object files for input to the linker; this can, in some cases, reduce link time for large applications. To specify the maximum number of object files to be produced, use the /Qipo*N* form of the option where *N* is the maximum number of object files to be created. For example, /Qipo4 specifies a maximum of 4 object files. The compiler may choose to create fewer files than the maximum depending on the application size. If /Qipo0 is specified, the compiler will choose an appropriate number of object files based on the total application size.

Change in Meaning of /fast

As of the 8.1 release, specifying /fast implies the following options: /O3 /Qipo /QxP

New /Qglobal-hoist[-] Optimization Option

/Qglobal-hoist[-] is an option that controls certain optimizations, load hoisting and speculative loads, that can move memory loads to a point earlier in the program execution than where they appear in the source. In most cases, these optimizations are safe and can improve performance. The default is /Qglobal-hoist, enabling the optimizations.

However, some applications, such as those that use shared or dynamically mapped memory, may fail if a load is moved too early in the execution stream (for example, before the memory is mapped.) If you wish to disable these optimizations, specify /Qglobal-hoist- when compiling the source files which reference the mapped or shared memory.

KMP_SCHEDULE Environment Variable for OpenMP Scheduling Control

A new environment variable, KMP_SCHEDULE, can be used to fine tune the load balancing of parallel loops that are statically scheduled under OpenMP with no chunk size specification. The default

value is KMP_SCHEDULE="static,greedy". This results in (#iterations/#threads) iterations, rounded to the next higher integer, being allocated to most threads, but the final thread(s) may be allocated much fewer, or even zero, iterations. This corresponds to previous compiler behavior. The alternative, KMP_SCHEDULE="static,balanced", results in (#iterations/#threads) iterations, rounded to the next lower integer, being allocated to most threads, with at most one additional iteration being allocated to some threads. Although the largest number of iterations assigned to any thread remains the same, this results in a more even sharing of iterations between threads, which may sometimes lead to a performance improvement.

For example, consider a loop of 9 iterations running on 4 threads:

KMP_SCHEDULE	Number of iterations			
	Thread 0	Thread 1	Thread 2	Thread 3
"static,greedy"	3	3	3	0
"static,balanced"	3	2	2	2

Additional New Options

For information on these options, please see the *New Options* section of the on-disk *Compiler Options Quick Reference Guide*.

- /fixed
- /Qcxx_features
- /Qipo_separate
- /traceback

What's New in Version 8.0

This section discusses new features and changes in the 8.0 release of the Intel® C++ compiler.

Compiler driver name change

If you use the Intel C++ Compiler for Itanium®-based systems, be aware that the driver name for Itanium-based systems has changed from ecl to icl.

Default installation directory change

The default installation directory for the Intel C++ Compiler 8.0 for Windows is now <Program Files>\Intel\cpp\ and the default for the Intel Visual Fortran is <Program Files>\Intel\Fortran\ If you have both Intel Visual Fortran and Intel C++ compilers installed, the respective product files are now installed in separate folders; this is a change from previous releases of Intel compilers. If you are building mixed-language applications from the command line, and you chose not to update the system environment variables on installation, you must invoke the other language's initialization batch file (ifortvars.bat or iccvars.bat) so that the PATH, LIB and INCLUDE environment variables are properly defined.

Note: Do not force installation of Fortran and C++ into the same folder, as this can result in problems building applications using Inter-Procedural Optimization(IPO) and Profile-Guided Optimization(PGO).

Optimization support for Intel® Pentium® 4 Processors with SSE3 Instruction Set

A new generation of Intel® Pentium® 4 processors supports the Streaming SIMD Extensions 3 (SSE3) instruction set, which can improve performance of vectorized loops containing complex data types, float-to-integer conversions, and horizontal adds.

The Intel C++ Compiler 8.0 for Windows adds the ability to optimize for Intel® processors that support SSE3. To do so, specify the `/QxP` or `/QaxP` options. For further details, please consult the sections on optimizations in the *Intel C++ Compiler User's Guide*.

New IA-32 Optimization Options

This release includes two new code generation options. `/QxB` and `/QaxB` direct the compiler to generate code for best performance on the Intel Pentium M processor. The new `/QxN` and `/QaxN` options enable additional optimizations for all Intel Pentium 4 processors. Intel recommends the use of `/QxN` and `/QaxN` for best performance with Pentium 4 processors, and suggests trying `/QxB` or `/QaxB` to see if it helps your application on the Pentium M processor. For more information, please refer to the sections on optimization in the *Intel C++ Compiler User's Guide*

Obsolete and Obsolescent Optimization Options

The optimization options `/Q[a]xi` (optimize for Pentium Pro and Pentium II) and `/Q[a]xM` (optimize for Intel® MMX™ instructions set) are no longer supported by the Intel C++ compiler. If these options are present on the compile command line, an informational message is displayed and the options are ignored. If you use `/Q[a]xi` or `/Q[a]xM`, you should discontinue their use. The default is to generate generic code that will run on Pentium processor as well as newer IA-32 processors.

`The/Q[a]xW` (lower optimization level for Pentium 4) will be removed in a future compiler version. If you use `/Q[a]xW`, use `/Q[a]xN` as a replacement when generating code for Intel Pentium 4 processors.

For more information, please refer to the sections on optimization in the *Intel C++ Compiler User's Guide*.

Read-Only (constant) data behavior change

Intel® C++ Compiler 8.0 for 32-bit applications
The 7.1 version of the Intel C++ Compiler places all constants and string literals in a writeable data section. Starting with the Intel C++ Compiler 8.0 the default behavior will change to match either the behavior of Visual C++ * 6.0 or Visual C++ .NET (2002 or 2003) depending on options used.

- By default or if `-Qvc6` is used, the Intel C++ Compiler 8.1 will place the const data in the read-only data section and string literals in a writeable data section.
- If `-Qvc7` or `-Qvc7.1` is used, the Intel C++ Compiler 8.1 will place string literals in the read-only section also.
- If `-Gf` is used, the Intel C++ Compiler 8.1 will always ensure that string literals are placed in a writeable section if an application depends on string literals being writeable

Intel C++ Compiler 8.0 for Itanium-based applications
The 7.1 version of the Intel C++ Compiler places all dimensioned constants and string literals in a writeable data section. Starting with the Intel C++ Compiler 8.0 the default behavior will change to match the behavior of the Microsoft* C/C++ Compiler.

- By default the dimensioned const data will be placed in a read-only data section and string literals will be placed in the read-only section.
- If `-Gf` is used, the Intel C++ Compiler 8.1 will always ensure that string literals are placed in a writeable section if an application depends on string literals being writeable

Microsoft Visual Studio* .NET* IDE integration

This version of the Intel C++ Compiler for Windows* has significantly improved the integrations with Microsoft's Visual Studio .NET 2002 and 2003. The following features have been added:

- The utility `"Enable or Disable Intel C++ Integration Tool"` has been eliminated. With the improved Visual Studio .NET integrations, you may select to `"Convert to use Intel® C++ Project System"` or `"Convert to use Visual C++ Project System..."` from the Project Menu.
- Microsoft Visual C++ projects are managed by the Microsoft Visual C++ VS .NET integrations until the user requests to `"Convert to use Intel® C++ Project System"` from the project menu.
- The new utility `"icProjConvert80"` has been provided to handle the project conversions of a Visual C++ project to an Intel C++ 8.1 project and back. This utility also manages the conversion of a Visual C++ project that has been modified by the 7.x Intel C++ Integration Tool to an Intel C++ 8.1 project.
- Non-privileged users may utilize the Intel C++ Project System within Microsoft Visual Studio .NET.
- The Intel C++ Options dialog accessible from the `[Tools..Options]` menu displays each supported Intel C++ compiler on a separate page.

Improved Visual Studio .NET command line option compatibility

The Intel C++ Compiler version 8.1 has improved command-line compatibility with VC++ with the `/showIncludes` and `/WL` driver options.

Visual C++ .NET compatible macros

The `__COUNTER__` and `__FUNCSIG__` macros have been predefined.

- `__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__` - 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 option.

Intel® Compilers code-coverage tool

The Intel® Compilers code-coverage tool leverages the Intel Compilers profile-guided optimization technology to present developers a complete picture of the coverage of their application code on a particular workload. To find the application's code coverage the tool combines static profile information generated by the compiler with dynamic profile information generated by running the user's instrumented binaries on the workload. The coverage tool uses this information together with the application sources to create HTML pages with color annotations that highlight the coverage of the code. Navigation is through frames that make it particularly easy to sort the application's files and functions and see the least-covered modules and functions. Developers can then use their favorite browser to display the coverage of their code.

The Intel Compilers code coverage tool can be used in a number of ways to improve development efficiency, reduce defects, and improve application performance. When applied to the profile of the application on its test space, the tool can be used to measure the overall quality of testing based on the coverage information. Similarly, when applied to the profile of a performance workload, the code-coverage information indicates how well the workload exercises the application's critical code. High coverage of performance-critical modules is essential to taking full advantage of the profile-guided optimizations that Intel Compilers offer. The tool also provides an option, useful for both coverage and performance tuning, through which the users can display the dynamic execution count of each basic block of the application. Lastly, the coverage tool provides the ability to compare the profile of two different runs of the application. This feature can be used to find the portion of the application's code that is not covered by the application's tests but is exercised when the application is used outside the test space, such as by a customer.

The Intel Compilers code-coverage tool is supported on Intel® Architecture IA-32 and the Itanium Processor Family on both Windows and Linux* and seamlessly supports C, C++, and Fortran.

Intel Compilers test-prioritization tool

The Intel Compilers test-prioritization tool leverages the Intel Compilers profile-guided optimizations technology to select and prioritize application's tests based on prior execution profiles of the application. Using this tool, users can select and prioritize the tests that are more relevant for any subset of the application's code. When certain modules of an application are changed, the Intel Compilers test-prioritization tool suggests the tests that are most probably affected by the given change set. The tool mines the profile data from previous runs of the application, discovers the dependency between the application's components and its tests, and uses this information to guide the process of testing. The tool can be used for devising an effective hierarchical testing based on the application's code coverage. For instance, the tool may be used to find the smallest subset of the application tests that achieve exactly the same code coverage as the entire set of tests. The tool can also be used to dramatically reduce the turn-around time of testing. Instead of spending a large amount of time and finding a possibly-large number of failures, the tool may enable the users to quickly find a small number of tests that expose the defects associated with the regressions caused by a change set. The tool offers the potential of significant time saving in testing and development of large-scale applications where testing is major bottleneck. The tool can be used to minimize the number of tests that are required to achieve a given overall coverage for any subset of the application. Moreover, when the execution times of the tests are available, the tool may also be used to select and prioritize the tests to achieve certain level of code coverage in a minimum amount of time.

The Intel Compilers test-prioritization tool is supported on Intel Architecture IA-32 and the Itanium Processor Family on both Windows and Linux and seamlessly supports C, C++, and Fortran.

Please refer to the following link for additional details:
http://www.intel.com/software/products/compilers/techtopics/pgt.htm .

`/Qalias_arg` to force Fortran style parameters

By default C assumes that parameters overlap; Fortran semantics assumes parameters do not overlap. This option allows users to assert Fortran semantics for C programs, improving optimization. This option would be particularly useful to users writing numerical code in C.

Versioned Intermediate files (`.il`) during interprocedural optimization (IPO)

Each `.il` file generated by IPO will have a version number. The compiler will only accept `.il` files with matching versions. The version numbers will be automatically generated and updated as part of the build process.

`libguide` can be linked dynamically only

The statically linked library, `libguide` can potentially cause performance issues that are hard to debug. The 8.1 compilers will link `libguide` dynamically regardless of the command line options.

Support for the `/Gh` and the `/GH` options

`/Gh` option is helpful for custom user profiling by calling the `___penter` function. The prototype for `___penter` is not included in any of the standard libraries or Intel libraries. Users do not need to provide a prototype unless they plan to explicitly call `___penter` .

`/GH` option is helpful for custom user profiling by calling the `___pexit` function. The prototype for `___pexit` is not included in any of the standard libraries or Intel libraries. Users do not need to provide a prototype unless they plan to explicitly call `___pexit`. `/GH` is similar to `/Gh`.

Better debug support for `/Qip` and `/Qipo` options

Better debug support is now provided for `/Qip` and the `/Qipo` options. Some information about variables will now be available (although values may not be completely accurate due to optimizations).

`/Qfpstkchk` option

This option would cause extra code to be generated after every function/subroutine call that would assure that the FP stack was in the state the compiler expected. When a customer calls a function that returns an FP value, the FP value is supposed to be returned on the top of the FP stack. If the return value is unused the compiler must just pop the value off the FP stack to keep the FP stack in the correct state. However, if the application has called such a function, but either has left out the function's prototype, or incorrectly prototyped the function such that the compiler doesn't know the function is returning an FP value, then the FP stack will not get popped as needed. This tends to cause the FP stack to fill up over time, and eventually overflow. When the stack overflows this generally results in a NAN value being put into FP calculations, and the programs results differ, or other error manifests itself. Unfortunately the point where the errors manifest can be arbitrarily far away from the point of the actual bug. This option will force an access violation exception immediately after such an incorrect call occurred, thus making it very easy for the user to find these issues.

System Requirements

Minimum Hardware Requirements to Develop IA-32 Applications

- A system based on a 450 MHz Intel® Pentium® II processor or greater, Intel Pentium 4 recommended
- 256 MB of RAM (512 recommended)
- 100 MB of free hard disk space, plus an additional 200 MB during installation for download and temporary files.
- 100 MB of hard disk space for the virtual memory paging file. Be sure to use at least the minimum amount of virtual memory recommended by your operating system.

Minimum Hardware Requirements to Develop Itanium®-based Applications on an IA-32 System

- A system based on a 450 MHz Intel Pentium II processor or greater, Intel Pentium 4 recommended
- 256 MB of RAM (512 recommended)
- 150 MB free hard disk space, plus an additional 200 MB during installation for download and temporary files.
- 100 MB of hard disk space for the virtual memory paging file. Be sure to use at least the minimum amount of virtual memory recommended by your operating system.

Minimum Hardware Requirements to Develop Itanium-based Applications on an Itanium-based System

- A system with an Intel® Itanium® or Itanium 2 processor or greater
- 512 MB of RAM (1 GB recommended)
- 100 MB free hard disk space, plus an additional 200 MB during installation for download and temporary files.

Software Requirements to Develop IA-32 or Itanium-based Applications on an IA-32 System

- Windows* 2000, Windows XP, Windows Server 2003
  Note: Microsoft Windows 98, Windows 98 SE, Windows Millennium Edition and Windows NT* are no longer supported for development, but are supported for application deployment
- Supported Microsoft Visual Studio environments
  - Microsoft Visual C++* 6.0 Professional Edition or higher
  - Microsoft Visual Studio* 6.0 Professional Edition or higher
  - Microsoft Visual Studio .NET 2002 or 2003, Professional Edition or higher
  - Microsoft Visual C++ .NET 2002 or 2003, Standard Edition or higher
- Microsoft Platform SDK if developing Itanium-based applications.
  Note: The DLLs in the Platform SDK directory `c:\Program files\Microsoft SDK\redist\win64` may also be required at runtime.
- Microsoft Macro Assembler (MASM) Version 7.00.9466 or later is required if you want to use options that produce or operate on assembly files.
- The Intel® C++ Compiler is not designed to be used in the Watcom* development environment.

Software Requirements to Develop Itanium-based Applications on an Itanium System

- Windows XP professional 64-bit edition or Windows Server 2003.

- Microsoft Platform SDK.
  Note: The DLL's in the Platform SDK directory `c:\Program files\Microsoft SDK\redist\win64` may also be required at runtime.

NOTES:
Adobe* Acrobat Reader* version 4.0 or later is required to view some of the product documentation.

It is the responsibility of application developers to ensure that the machine instructions contained in the application are supported by the operating system and the processors on which the application is to run. In particular, programs which use Streaming SIMD Extensions require Windows NT 4.0 with Service Pack 6 or higher, Windows 2000 or Windows XP, running on an Intel Pentium III processor. The Streaming SIMD Extensions 2 and Streaming SIMD Extensions 3 of the Intel Pentium 4 processor also require one of these operating systems.

Installation

Pre-Installation Instructions

To install the Intel® C++ Compiler, you need to obtain an account with administrative privileges. But any normal account with at least "Users" or "Debugger Users" or higher user privilege can use Intel C++ Compiler through Visual C++* .NET IDE or command line.

Note: the default installation directory for the Intel C++ Compiler is `C:\Program Files\Intel\cpp\`.

If you wish to use the IA-32 compiler that generates Itanium-based applications you must first install the Microsoft Platform SDK on your IA-32 development system prior to installing this product.

If you wish to use the Itanium-based compiler that generates Itanium-based applications you must first install the Microsoft Platform SDK on your Itanium-based system before installing this product on it.

Note: If you have version 7.0 or 7.1 of Intel C++ Compiler installed, with the Microsoft Visual C++.NET integration enabled, you must remove the Visual C++.NET integration before installing this version of Intel C++ Compiler. To do this, follow these steps:

1. Select Start..Programs..Intel(R) Software Development Tools..Intel(R) C++ Compiler 7.1..Enable or Disable Visual C++.NET 2002 (or 2003) Integration.
2. In the Intel® Package Registration window that appears, select Disable Intel Package and click OK.
3. In the Windows Control Panel, select Add or Remove Programs.
4. Select the Intel C++ Compiler 7.x you wish to modify and click the Change button. When the InstallShield* Wizard appears, click Next.
5. Select Modify and click Next.
6. Click on the "+" symbols, as needed, to expand the entries under "Intel(R) C++ Compiler for 32-bit applications", "Microsoft Visual Studio Integration".
7. Change the entries for "For Microsoft Visual C++.NET 2002" and "For Microsoft Visual C++.NET 2003" to "This feature will not be installed", signified by a red "X". You will see only the entries corresponding to the editions of Microsoft Visual C++.NET installed on your system. Click Next.
8. Click Next, Install and then Finish.

The **recommended installation order** is as follows:

1. Install Visual C++ .NET 2002 or Visual C++ .NET 2003 (Standard edition or above) or Visual C++ 6.0 Professional edition or above
2. Install Microsoft Platform SDK if needed
3. If you have the Intel C++ Compiler 7.x installed on your system, please uninstall the VC++ .NET Integration as described in the note above.
4. Uninstall other versions of the Intel C++ Compiler 8.x
5. Install this version of the Intel C++ Compiler 8.1

Note: To learn how to uninstall the Intel C++ Compiler, please read the Uninstalling the Compiler section below.

The Intel C++ Compiler 8.1 can coexist with the Intel C++ Compiler 6.0.1 and 7.1 product.

Installation

1. Installing the license

   The Intel C++ Compiler uses Macrovision Corporation's FLEXlm* electronic licensing technology. License management is transparent. The installation program of the Intel C++ Compiler 8.1 checks for a valid license before installing any component of the product. Also, the license must remain in place on the system in order to use the Intel C++ Compiler 8.1 to compile and build programs.

   Note: Your existing license for the Intel C++ Compiler for Linux* will work with the 8.1 compiler provided your support services have not expired.

   The FLEXlm license daemon for Intel software, used for floating and node-locked licenses only, is available for multiple platforms. The daemon may be installed on any supported platform accessible on your local network. The compiler CD contains license daemons for most platforms. If you do not have the CD, or need a license daemon for an additional platform, please contact Intel® Premier Support.

2. Here is how to setup the license file before installation.
   - If you have an electronically downloaded version of the Intel C++ Compiler 8.1, the license will be sent to you via email. Please follow the instructions in the email to install the license file.

     If you have a CD version of the Intel C++ Compiler 8.1, a valid license is included on the CD and the installation program can locate it automatically. But, in order to obtain access to technical support and to be able to download and execute product updates, as **a CD-ROM user** you must do the following:

     1. **Register your product:** First, locate the serial number found on the inside flap of the product box. Then, visit the web site http://www.intel.com/software/products/registrationcenter/ and follow the instructions. After the registration you will receive an email within 24 hours containing a new license.
     2. **Install the new license:** The new license in the email typically entitles you to one year of support services that allow you to download and execute product updates and obtain full technical support. The email also contains the instructions on how to install the license. Please follow the instructions to finish the new license installation.

     Note:
     The license file must have an extension "`.lic`".

The default license directory is `C:\Program Files\Common Files\Intel\Licenses\`.

For details about the support service license, please see http://www.intel.com/software/products/compilers/cwin/pricelist.htm .

3. Installing the Intel C++ Compiler

   After you have downloaded the package, simply run the downloaded executable (For example `w_cc_p_8.1.xxx.exe`).

Install time license checking
Before installing any component, the installation program of Intel C++ Compiler 8.1 checks for a valid license. It searches for a valid license file at folders pointed by "INTEL_LICENSE_FILE" first. If there's no valid license, you will be prompted to enter a valid license file that you have just created in the previous step.

After the license check, simply follow the setup program's prompts to complete the installation. The installation program will install the corresponding license to `C:\Program Files\Common Files\Intel\Licenses` on both IA-32 systems and Itanium-based systems.

If you have any problems running the compiler, please make sure a valid license file (*.lic) is located in the license directory. If you still have problems, please submit an issue to Premier Support. See the Technical Support section of this document for details.

The Intel C++ Compiler 8.1 license can coexist with previous versions of the Intel Compiler license.

Note about Intel Debugger Installation

If you install both Intel® Visual Fortran and Intel C++, you will have two copies of the Intel® Debugger. Whichever version was installed most recently is the one that will be used by default.

Uninstalling the Compiler

To uninstall the Intel C++ Compiler 8.1 for Windows completely, you need to uninstall the following with `"Add/Remove Programs"` from the `"Control Panel"`.

- Intel C++ Compiler 8.1 for Windows
- Intel® License Manager for FLEXlm if installed

Note: uninstalling the Intel C++ Compiler does not delete the corresponding license file.

Known Limitations

- Installation related limitations
  - After installation from an electronic download, an unpacked copy of the installation files remains in a temporary directory, by default `C:\Documents and Settings\username\Local Settings\Temp\IntelC++Compiler80`. If desired, the user may delete these files.

    - If you install the Intel® C++ Compiler 8.1 through Microsoft Terminal Services Client*, you need to log off after finishing the installation and re-log on in order for environment variables to be set correctly when using the Intel C++ Compiler.
    - If you have the Intel C++ Compiler 6.0.1 on your system, before installing the 8.1, please uninstall the VC++ .NET 2002 Integration Tool of the Intel C++ Compiler 6.0.1 with the "Modify" feature provided in the program `"Modify or Remove Intel® C++ and EDB"`.
      When you uninstall the Intel C++ Compiler 8.1, please reinstall the VC++ .NET 2002 Integration Tool of the Intel C++ Compiler 6.0.1.
    - If you have both the Intel C++ Compiler 7.1 and 8.1 installed to a system with Visual C++ * 6.0 installed, uninstalling the 7.1 will also remove the "Intel C++ Compiler Selection Tool" for 8.1 if it's installed. Please repair or reinstall 8.1 to fix the problem.
    - If you install the Intel C++ Compiler 8.1 with VS .NET integration to a different location than where it was previously installed on your system (install-uninstall-reinstall), the VS .NET will be unable to locate the Intel C++ Compiler 8.1 within the IDE.
      The work around is to delete the following registry keys:
        - HKEY_CURRENT_USER\software\intel\ide\C++\80
        - HKEY_CURRENT_USER\software\intel\compilers\c++\80
- Limitations on Visual C++ .NET language support
  There are a number of new features introduced by Visual C++ .NET that are not support by this version of Intel C++ Compiler for Windows. Please read the *User's Guide* for additional information.
    - Attributed code is not supported
    - Event handling (new keywords) is not supported
    - Managed extensions for C++ is not supported
- Limitations on integration with Visual C++ .NET
    - The Intel C++ compiler only supports non VC++ .NET specific project types such as `"Win32 Console Project", "Win32 Application"`, etc.
      Project types with VC++ .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)`
- Limitation with the Intel® Debugger (`IDB`)
    - The Intel Debugger is not supported on Windows systems with only Visual Studio* .NET (2002 or 2003) Standard Edition installed.
    - The Intel Debugger for Itanium-based system does not have a Graphical User Interface (GUI) in this release of the product. It may provide a GUI in the near future.
- There will be an increase in compile time when `-Zi` is used together with inlining. Inlining can happen if the user specifies `-ipo, -ip` or compiles a `C++/C99` program at option levels `-O1` or above. This is due to the generation of debug information. For many applications, this combination of compiler options will not increase compile time or compile-time memory use.

Issues relating to Multiple Object File Interprocedural Optimization

The following issues are expected to be resolved in a future update:

`/Qipo1` doesn't guarantee only one object file

Specifying an explicit number of files object files to be generated with `/QipoN` doesn't turn off the compiler heuristic for dividing and object file into two object files when it gets too big.

Because of this, you cannot currently turn off IPO multiple objects by explicitly specifying one object file using `/Qipo1`.

`/Qqipo_separate` is not recognised by `xilink`

The `/Qipo_separate` option is not recognized by `xilink`. This causes IPO compilations using this option to fail.

Explicit naming of obj and asm files ignored with `/Qipo` multiple objects

When using /Q`ipo_c` or `/Qipo_S` (explicit .obj or .asm files, respectively), options to explicitly name these files are ignored by the compiler for when generating multiple objects.

asm files generated by `/Qipo_S` fail to assemble with multiple object IPO

The assembler complains that routines being called haven't been defined.

Limited Debug Information with Automatic CPU Dispatching (`/Qax*`)

Compilation using `/Qax{W|N|B|P}` results in two copies of generated code for each function. One for IA-32 generic code and one for CPU specific code. The symbol for each function then refers to an Auto CPU Dispatch routine that decides at run-time which one of the generated code sections to execute. Debugger breakpoints that are set on these functions by name cause the application to stop in the dispatch routine. This may cause unexpected behavior when debugging. This issue may be addressed in a future version of the Intel Debugger and Compilers.

Cannot Debug or View Traceback for IA-32 Programs Built with `/Oy-`

Compilation using `/Oy-` specifies that the IA-32 EBP register be used as a general purpose register, eliminating its use as a frame pointer. Debuggers and traceback handlers may not be able to properly unwind through a stack that contains a call to a function that is compiled in this manner.

Other Issues

Please click on the following links to see additional notes and known issues in the latest version of each tool.

- Intel® C++ Compiler to produce IA-32 applications
  Note, this file is available only if the compiler for 32-bit applications is installed.
- Intel® C++ Compiler to produce Itanium®-based applications
  Note, this file is available only if the compiler for Itanium-based applications is installed.

Technical Support

Your feedback is very important to us. To receive technical support for the tools provided in this product and technical information including FAQ's and product updates, you need to be registered for an Intel® Premier Support account on our secure web site, https://premier.intel.com. Please register at

http://support.intel.com/support/performancetools/support.htm and click on "Registration Center".

Note:

- Registering for support varies for release product or pre-release products (alpha, beta, etc) - only released products have support web pages on http://support.intel.com.
- If you are having trouble registering or unable to access your Premier Support account, contact developer.support@intel.com. Please do not email your technical issue to developer.support@intel.com as it is not a secure medium.
- If you have forgotten your password, please email a request to: quad.support@intel.com. Please do not email your technical issue to this email address as it is not a secure medium.

For information about the Intel® C++ Compiler's Users Forums, FAQ's, tips and tricks, and other support information, please visit: http://support.intel.com/support/performancetools/c/windows/. For general support information please visit http://www.intel.com/software/products/support/.

Submitting Issues

Steps to submit an issue:

1. Go to https://premier.intel.com/.
2. Type in your Login and Password. Both are case-sensitive.
3. Click the "Submit" button.
4. Read the Confidentiality Statement and click the "I Accept" button.
5. Click on the "Go" button next to the "Product" drop-down list.
6. Click on the "Submit Issue" link in the left navigation bar.
7. Choose "Development Environment (tools,SDV,EAP)" from the "Product Type" drop-down list.
8. If this is a software or license-related issue, choose "Intel C++ Compiler, Windows*" from the "Product Name" drop-down list.
9. Enter your question and complete the fields in the windows that follow to successfully submit the issue.

Guidelines for problem report or product suggestion:

1. Describe your difficulty or suggestion.
   For problem reports please be as specific as possible, so that we may reproduce the problem. For compiler problem reports, please include the compiler options and a small test case if possible.
2. Describe your system configuration information.
   Run the compiler (`icl`) from the command window like below:
   (**Note: in order to display the correct Package ID, the command should be invoked from the drive where the compiler is installed.**)
   `>> icl` on an IA32-based system or Itanium-based system
   and copy the "Package ID" (e.g. **`w_cc_b_8.1.xxx`**) from the output into the corresponding Premier Support field. Please include any other specific information that may be relevant to helping us to reproduce and address your concern.
3. If you were not able to install the compiler or cannot get the Package ID, enter the filename you downloaded as the package ID.

Resolved Issues

Please review <package ID>_README (e.g. w_cc_p[c]_8.1.xxx_README), available for download from Intel® Premier Support, https://premier.intel.com, to see which issues have been resolved in the latest version of the compiler.

Documentation

You can view the Intel® compiler and related HTML-based documentation with your Web browser, which provide full navigation, index look-up, search, and hyperlink capabilities.

The documentation index is provided for easy access of all the documents. The Document index is available from the Intel C++ Compiler program folder and is located at: <installation_directory>\Compiler80\docs\ccompindex.htm. A training tutorial *Enhancing Performance with Intel Compilers* is also available from the Intel C++ Compiler program folder. *The Intel® Debugger Manual* is available from the Intel® Debugger program folder.

The document *Intel® C++ Compiler User's Guide* is now organized into separate parts:

- An Options Quick Reference Guide
- User's Guide for Building Applications
- User's Guide for Optimizing Applications
- Reference Information

Note:

In the documentation index file (ccompindex.htm), if you find that certain links do not work, please access the following Web page to download a patch for Internet Explorer* applicable to your operating system:

http://support.microsoft.com/support/KB/articles/Q811/6/30.asp

For more information on problems opening HTML Help files using the windows.showhelp attribute, see *Microsoft Knowledge Base Article 822989*.

Additional Information

Related Products and Services

Information on Intel software development products is available at http://www.intel.com/software/products.

Some of the related products include:

- The Intel® Software College provides training for developers on leading-edge software development technologies. Training consists of online and instructor-led courses covering all Intel architectures, platforms, tools, and technologies.
- The VTune™ Performance Analyzer enables you to evaluate how your application is utilizing the CPU and helps you determine if there are modifications you can make to improve your application's performance.
- The Intel® C++ and Fortran Compilers are an important part of making software run at top speeds with full support for the latest Intel IA-32 and Itanium® processors.

- The Intel® Performance Library Suite provides a set of routines optimized for various Intel processors. The Intel® Math Kernel Library, which provides developers of scientific and engineering software with a set of linear algebra, fast Fourier transforms and vector math functions optimized for the latest Intel Pentium® and Intel Itanium processors. The Intel® Integrated Performance Primitives consists of cross-platform tools to build high performance software for several Intel architectures and several operating systems.

Copyright and Legal Information