

Intel® C++ Composer XE 2011 for Windows* Installation Guide and Release Notes

Document number: 321414-003US
17 November 2010

Table of Contents

1	Introduction	3
1.1	Change History	3
1.2	Product Contents	3
1.3	System Requirements.....	3
1.3.1	IA-64 Architecture (Intel® Itanium®) Development Not Supported.....	5
1.4	Documentation.....	5
1.5	Samples.....	6
1.6	Japanese Language Support.....	6
1.7	Technical Support.....	6
2	Installation.....	6
2.1	Pre-installation Steps	6
2.1.1	Configure Visual Studio for 64-bit Applications.....	6
2.1.2	Installation on Microsoft Windows Vista* and Windows 7*	7
2.2	Installation	7
2.2.1	Activation of Purchase after Evaluation Using the Intel Activation Tool	7
2.2.2	Using a License Server.....	8
2.2.3	Prompt for Administrator Permission with Microsoft Visual Studio 2005*	8
2.3	Changing, Updating and Removing the Product	8
2.4	Installation Folders.....	8
2.5	Installation Known Issues.....	10
2.5.1	Documentation Issue with Multiple Visual Studio Versions.....	10
3	Intel® C++ Compiler	10
3.1	Compatibility	10
3.2	New and Changed Features	10

3.2.1	Static Security Analysis Feature (formerly Source Checker) Requires Intel® Inspector XE	11
3.2.2	Microsoft* Visual Studio 2010* Support.....	11
3.2.3	Intel® C++ Project File Compatibility.....	11
3.3	New and Changed Compiler Options	12
3.3.1	Deprecated Options	12
3.4	Other Changes	13
3.4.1	Build Environment Command Script Change	13
3.4.2	OpenMP* Legacy Libraries Removed	13
3.4.3	OpenMP* Libraries Default to Dynamic Linking.....	13
3.4.4	Using Intel C++ Projects with a Source Control System.....	13
3.5	Known Issues	14
3.5.1	Compiler Known Issues	14
3.5.2	Visual Studio Known Issues.....	14
3.5.3	Intel® Cilk™ Plus Known Issues.....	15
3.5.4	Guided Auto-Parallel Known Issues.....	16
3.5.5	Static Security Analysis Known Issues.....	16
4	Intel® Integrated Performance Primitives	17
4.1	Intel® Integrated Performance Primitives 7.0.....	17
4.1.1	New and Changed Features	17
4.1.2	Directory Structure Changed.....	18
4.2	Intel® IPP Cryptography Libraries are Available as a Separate Download.....	18
4.3	SPIRAL Generated Functions are Available as a Separate Download	18
4.4	Intel® IPP Code Samples	19
5	Intel® Math Kernel Library	19
5.1	Changes in This Version	19
5.1.1	Changes in Initial Release	19
5.1.2	Changes in Update 1	21
5.2	Attributions.....	21
6	Intel® Threading Building Blocks	22
6.1	Known Issues	22
6.1.1	Library Issues	22
7	Intel® Parallel Debugger Extension	23

7.1	New Features	23
7.2	Known Issues	23
7.3	Documentation.....	24
8	Disclaimer and Legal Information.....	24

1 Introduction

This document describes how to install the product, provides a summary of new and changed product features and includes notes about features and problems not described in the product documentation.

Intel® C++ Composer XE 2011 is the next release of the product formerly called Intel® C++ Compiler Professional Edition.

1.1 Change History

This section highlights important changes in product updates.

Update 1 (12.0.1)

- [Intel® Math Kernel Library updated to 10.3 Update 1](#)
- Corrections to reported problems

Product Release (12.0.0)

- Initial product release

1.2 Product Contents

*Intel® C++ Composer XE 2011 Update1 for Windows** includes the following components:

- Intel® C++ Compiler XE 12.0 Update1 for building applications that run on IA-32 or Intel® 64 architecture systems running the Windows* operating system
- Intel® Integrated Performance Primitives 7.0 Update 1
- Intel® Math Kernel Library 10.3 Update1
- Intel® Threading Building Blocks 3.0 Update 3
- Intel® Parallel Debugger Extension Update1
- Integration into Microsoft* development environments
- Sample programs
- On-disk documentation

1.3 System Requirements

For an explanation of architecture names, see <http://software.intel.com/en-us/articles/intel-architecture-platform-terminology/>

- A PC based on an IA-32 or Intel® 64 architecture processor supporting the Intel® Streaming SIMD Extensions 2 (Intel® SSE2) instructions (Intel® Pentium® 4 processor or later), or compatible non-Intel processor
 - For the best experience, a multi-core or multi-processor system is recommended
- 1GB RAM (2GB recommended)
- 4GB free disk space for all product features and all architectures
- Microsoft Windows XP*, Microsoft Windows Vista*, Microsoft Windows 7*, Microsoft Windows Server 2003*, Microsoft Windows Server 2008* or Microsoft Windows HPC Server 2008* (embedded editions not supported)
 - Microsoft Windows Server 2008 or Windows HPC Server 2008 requires Microsoft Visual Studio 2008* SP1. Other versions of Visual Studio listed below are not supported on Windows Server 2008 or Windows HPC Server 2008.
- To use the Microsoft Visual Studio development environment or command-line tools to build IA-32 or Intel® 64 architecture applications, one of:
 - Microsoft Visual Studio 2010* with C++ and “X64 Compiler and Tools” components installed [\[1\]](#)
 - Microsoft Visual Studio 2008* Standard Edition (or higher edition) with C++ and “X64 Compiler and Tools” components installed [\[1\]](#)
 - Microsoft Visual Studio 2005* Standard Edition (or higher edition) with C++ and “X64 Compiler and Tools” components installed [\[1\]](#)
- To use command-line tools only to build IA-32 architecture applications, one of:
 - Microsoft Visual C++ 2010* Express Edition
 - Microsoft Visual C++ 2008* Express Edition
 - Microsoft Visual C++ 2005* Express Edition and Microsoft Windows SDK for Windows 2008 and .NET Framework 3.5*
- To use command-line tools only to build Intel® 64 architecture applications, one of:
 - Microsoft Windows Software Development Kit Update for Windows 7*
 - Microsoft Windows SDK for Windows 2008 and .NET Framework 3.5*
- To read the on-disk documentation, Adobe Reader* 7.0 or later

Notes:

1. Microsoft Visual Studio 2005 and 2008 Standard Edition installs the “x64 Compiler and Tools” component by default – the Professional and higher editions require a “Custom” install to select this. Microsoft Visual Studio 2010 includes x64 support by default.
2. The default for the Intel® compilers is to build IA-32 architecture applications that require a processor supporting the Intel® SSE2 instructions - for example, the Intel® Pentium® 4 processor. A compiler option is available to generate code that will run on any IA-32 architecture processor. However, if your application uses Intel® Integrated Performance Primitives or Intel® Threading Building Blocks, executing the application will require a processor supporting the Intel® SSE2 instructions.
3. Applications can be run on the same Windows versions as specified above for development. Applications may also run on non-embedded 32-bit versions of Microsoft Windows earlier than Windows XP, though Intel does not test these for compatibility.

Your application may depend on a Win32 API routine not present in older versions of Windows. You are responsible for testing application compatibility. You may need to copy certain run-time DLLs onto the target system to run your application.

1.3.1 IA-64 Architecture (Intel® Itanium®) Development Not Supported

This product version does not support development on or for IA-64 architecture (Intel® Itanium®) systems. The version 11.1 compiler remains available for development of IA-64 architecture applications.

1.4 Documentation

Product documentation can be found in the `Documentation` folder as shown under [Installation Folders](#).

Optimization Notice

Intel® compilers, associated libraries and associated development tools may include or utilize options that optimize for instruction sets that are available in both Intel® and non-Intel microprocessors (for example SIMD instruction sets), but do not optimize equally for non-Intel microprocessors. In addition, certain compiler options for Intel compilers, including some that are not specific to Intel micro-architecture, are reserved for Intel microprocessors. For a detailed description of Intel compiler options, including the instruction sets and specific microprocessors they implicate, please refer to the “Intel® Compiler User and Reference Guides” under “Compiler Options.” Many library routines that are part of Intel® compiler products are more highly optimized for Intel microprocessors than for other microprocessors. While the compilers and libraries in Intel® compiler products offer optimizations for both Intel and Intel-compatible microprocessors, depending on the options you select, your code and other factors, you likely will get extra performance on Intel microprocessors.

Intel® compilers, associated libraries and associated development tools may or may not optimize to the same degree for non-Intel microprocessors for optimizations that are not unique to Intel microprocessors. These optimizations include Intel® Streaming SIMD Extensions 2 (Intel® SSE2), Intel® Streaming SIMD Extensions 3 (Intel® SSE3), and Supplemental Streaming SIMD Extensions 3 (Intel® SSSE3) instruction sets and other optimizations. Intel does not guarantee the availability, functionality, or effectiveness of any optimization on microprocessors not manufactured by Intel. Microprocessor-dependent optimizations in this product are intended for use with Intel microprocessors.

While Intel believes our compilers and libraries are excellent choices to assist in obtaining the best performance on Intel® and non-Intel microprocessors, Intel recommends that you evaluate other compilers and libraries to determine which best meet your requirements. We hope to win your business by striving to offer the best performance of any compiler or library; please let us

know if you find we do not.

Notice revision #20101101

1.5 Samples

Samples for each product component can be found in the `Samples` folder as shown under [Installation Folders](#).

1.6 Japanese Language Support

Intel compilers provide support for Japanese language users. Error messages, visual development environment dialogs and some documentation are provided in Japanese in addition to English. By default, the language of error messages and dialogs matches that of your operating system language selection. Japanese-language documentation can be found in the `ja_JP` subdirectory for documentation and samples.

If you wish to use Japanese-language support on an English-language operating system, or English-language support on a Japanese-language operating system, you will find instructions at <http://software.intel.com/en-us/articles/changing-language-setting-to-see-english-on-a-japanese-os-environment-or-vice-versa-on-windows/>

1.7 Technical Support

If you did not register your compiler during installation, please do so at the [Intel® Software Development Products Registration Center](#). Registration entitles you to free technical support, product updates and upgrades for the duration of the support term.

For information about how to find Technical Support, Product Updates, User Forums, FAQs, tips and tricks, and other support information, please visit <http://www.intel.com/software/products/support/>

Note: If your distributor provides technical support for this product, please contact them for support rather than Intel.

2 Installation

2.1 Pre-installation Steps

2.1.1 Configure Visual Studio for 64-bit Applications

If you are using Microsoft Visual Studio 2005* or 2008 and will be developing 64-bit applications (for the Intel® 64 architecture) you may need to change the configuration of Visual Studio to add 64-bit support.

If you are using Visual Studio 2005/2008 Standard Edition, or Visual Studio 2010 Professional Edition or higher, no configuration is needed to build Intel® 64 architecture applications. For other editions:

1. From Control Panel > Add or Remove Programs, select “Microsoft Visual Studio 2005” (or 2008) > Change/Remove. The Visual Studio Maintenance Mode window will appear. Click Next.
2. Click Add or Remove Features
3. Under “Select features to install”, expand Language Tools > Visual C++
4. If the box “X64 Compiler and Tools” is not checked, check it, then click Update. If the box is already checked, click Cancel.

Note that Visual C++ Express Edition does not support 64-bit development.

2.1.2 Installation on Microsoft Windows Vista* and Windows 7*

Microsoft Visual Studio 2005* users should install *Visual Studio 2005 Service Pack 1 (VS 2005 SP1)* as well as the *Visual Studio 2005 Service Pack 1 Update for Windows Vista*, which is linked to from the VS 2005 SP1 page. After installing these updates, you must ensure that Visual Studio runs with Administrator permissions, otherwise you will be unable to use the Intel compiler. For more information, please see Microsoft's Visual Studio on Windows Vista page (<http://msdn2.microsoft.com/en-us/vstudio/aa948853.aspx>) and related documents.

2.2 Installation

The installation of the product requires a valid license file or serial number. If you are evaluating the product, you can also choose the “Evaluate this product (no serial number required)” option during installation.

If you received your product on DVD, insert the first product DVD in your computer's DVD drive; the installation should start automatically. If it does not, open the top-level folder of the DVD drive in Windows Explorer and double-click on setup.exe.

If you received your product as a downloadable file, double-click on the executable file (.EXE) to begin installation. Note that there are several different downloadable files available, each providing different combinations of components. Please read the download web page carefully to determine which file is appropriate for you.

You do not need to uninstall previous versions or updates before installing a newer version – the new version will coexist with the older versions

2.2.1 Activation of Purchase after Evaluation Using the Intel Activation Tool

Note for evaluation customers a new tool Intel Activation Tool “ActivationTool.exe” is included in this product release and installed at “[Common Files]\Intel\Parallel Studio XE 2011\Activation\”.

If you installed the product using an Evaluation license or SN, or using the “Evaluate this product (no serial number required)” option during installation, and then purchased the product, you can activate your purchase using the Intel Activation Tool at Start > All Programs > Intel

Parallel Studio XE 2011 > Product Activation. It will convert your evaluation software to a fully licensed product.

2.2.2 Using a License Server

If you have purchased a “floating” license, see <http://software.intel.com/en-us/articles/licensingsetting-up-the-client-floating-license/> for information on how to install using a license file or license server. This article also provides a source for the Intel® License Server that can be installed on any of a wide variety of systems.

2.2.3 Prompt for Administrator Permission with Microsoft Visual Studio 2005*

If you are installing on Microsoft Windows Vista* or later versions of Microsoft Windows and are using Microsoft Visual Studio 2005, Windows may display a dialog similar to the following:



If this is displayed, it is important that you click the Continue button and leave the “Always show this message” box checked. If you select “Exit Visual Studio” instead, or do nothing (this message times out after two minutes), the compiler integration will not install completely. For more information, see [Installation on Microsoft Windows Vista*](#).

2.3 Changing, Updating and Removing the Product

Use the Windows Control Panel “Add or Remove Products” applet to change which product components are installed or to remove the product.

When installing an updated version of the product, you do not need to remove the older version first. You can have multiple versions of the compiler installed and select among them. If you remove a newer version of the product you may have to reinstall the integrations into Microsoft Visual Studio from the older version.

2.4 Installation Folders

The installation folder arrangement is shown in the diagram below. Not all folders will be present in a given installation.

- C:\Program Files\Intel\ComposerXE-2011

- o bin
 - ia32
 - ia32_intel64
 - intel64
- o compiler
 - include
 - ia32
 - intel64
 - lib
 - ia32
 - intel64
- o debugger
- o Documentation
- o Help
- o ipp
 - bin
 - demo
 - include
 - interfaces
 - lib
 - tools
- o mkl
 - benchmarks
 - bin
 - examples
 - include
 - interfaces
 - lib
 - tests
 - tools
- o tbb
 - bin
 - examples
 - include
 - lib
- o redist
- o Samples
- o VS Integration

Where the folders under `bin`, `include` and `lib` are used as follows:

- `ia32`: Files used to build applications that run on IA-32
- `intel64`: Files used to build applications that run on Intel® 64

- `ia32_intel64`: Compilers that run on IA-32 to build applications that run on Intel®64

If you are installing on a system with a non-English language version of Windows, the name of the `Program Files` folder may be different. On Intel® 64 architecture systems, the folder name is `Program Files (X86)` or the equivalent.

By default, updates of a given version will replace the existing directory contents. When the first update is installed, the user is given the option of having the new update installed alongside the previous installation, keeping both on the system. If this is done, the top-level folder name for the older update is changed to `ComposerXE-2011.nnn` where `nnn` is the update number.

2.5 Installation Known Issues

2.5.1 Documentation Issue with Multiple Visual Studio Versions

If you have both Microsoft Visual Studio* 2005 and 2008 installed on your system and integrate Intel® C++ Composer XE 2011 into both versions, removing the integration from one of the versions will remove the integrated Intel® C++ Composer XE 2011 documentation from both.

To re-install the documentation:

1. Use the Control Panel to select the product.
 - For Windows XP* users: Select **Control Panel > Add/Remove Programs**.
 - For Windows 7* users: Select **Control Panel > Programs and Features**.
 - For Windows Vista* users: Select **Control Panel > Programs**.
2. With the product selected, click the **Change/Remove** button and choose Modify mode.
3. In the **Select Components** dialog box, unselect “Integrated Documentation;” this will **remove** the documentation.
4. Repeat steps 1 and 2.
5. In the **Select Components** dialog box, select “Integrated Documentation” to **install** documentation again

3 Intel® C++ Compiler

This section summarizes changes, new features and late-breaking news about the Intel C++ Compiler.

3.1 Compatibility

In version 11, the IA-32 architecture default for code generation has changed to assume that Intel® Streaming SIMD Extensions 2 (Intel® SSE2) instructions are supported by the processor on which the application is run. [See below](#) for more information.

3.2 New and Changed Features

The following features are new or significantly enhanced in Intel® C++ Compiler XE 12.0. For more information on these features, please refer to the documentation viewable from within Visual Studio.

- Intel® Cilk™ Plus language extensions for the Intel® C++ Compiler make it easy to add parallelism to both new and existing software.
- Guided Auto-Parallelism
- Features from C++0x
 - rvalue references
 - Standard atomics
 - Support of C99 hexadecimal floating point constants when in “Windows C++” mode
 - Right angle brackets
 - Extended friend declarations
 - Mixed string literal concatenations
 - Support for `long long`
 - Variadic macros
 - Static assertions
 - Auto-typed variables
 - Extern templates
 - `__func__` predefined identifier
 - Declared type of an expression (`decltype`)
 - Universal character name literals
 - Strongly-typed enums
 - Lambdas
- An option to use math library functions that are faster but return results with less precision or accuracy
- An option to use math library functions that return consistent results across different models and manufacturers of processors

3.2.1 Static Security Analysis Feature (formerly Source Checker) Requires Intel® Inspector XE

The “Source Checker” feature, from compiler version 11.1, has been enhanced and renamed “Static Security Analysis”. The compiler options to enable Static Security Analysis remain the same as in compiler version 11.1 (for example, `/Qdiag-enable:sc`), but the results are now written to a file that is interpreted by Intel® Inspector XE rather than being included in compiler diagnostics output.

3.2.2 Microsoft* Visual Studio 2010* Support

This release supports Microsoft Visual Studio 2010. The Intel® C++ integration of Visual Studio 2010 is very different in comparison to the integration of Visual Studio 2005 or 2008. Please see this article for some details: <http://software.intel.com/en-us/articles/what-are-the-differences-in-ide-integration-of-visual-studio-2010-and-visual-studio-2005-2008/>

3.2.3 Intel® C++ Project File Compatibility

The Intel C++ project file (`.icproj`) format changed in version 12.0. If you open a project created with an earlier version of Intel C++, you will get a message indicating that the project needs to be converted. A version 12 project cannot be used by an earlier version of the Intel

C++ integration but you can use older versions of the compiler that you have installed through Tools > Options > Intel C++ > Compilers.

3.3 New and Changed Compiler Options

For details on these and all compiler options, see the Compiler Options section of the on-disk documentation.

- /Qansi-alias-check
- /Qcilk-serialize
- /Qdiag-sc-dir
- /Qfp-trap
- /Qfp-trap-all
- /Qguide
- /Qguide-data-trans
- /Qguide-file
- /Qguide-file-append
- /Qguide-opts
- /Qguide-par
- /Qguide-vec
- /Qimf-absolute-error
- /Qimf-accuracy-bits
- /Qimf-arch-consistency
- /Qimf-max-error
- /Qimf-precision
- /Qintel-extensions
- /Qopt-args-in-regs
- /Qopt-matmul
- /Qpatchable-addresses
- /Qprof-value-profiling
- /Qprofile-functions
- /Qprofile-loops
- /Qprofile-loops-report
- /Qpar-runtime-control[n]
- /Qpar-runtime-control-
- /Qregcall
- /Qsimd[-]
- /Qzero-initialized-in-bss

For a list of deprecated compiler options, see the Compiler Options section of the documentation.

3.3.1 Deprecated Options

Use following method to find all the deprecated compiler options:

- 1) Open a command prompt from Start menu: Start > All Programs > Intel Parallel Studio XE 2011 > Command Prompt > Parallel Studio XE with Intel Compiler XE 2011 v12.0 [Update xx] > IA-32 Visual Studio xxx mode
- 2) Run command:

```
>> icl /? deprecate
```

3.4 Other Changes

3.4.1 Build Environment Command Script Change

The command window script used to establish the build environment allows the optional specification of the version of Microsoft Visual Studio to use. If you are not using the predefined Start menu shortcut to open a build environment window, use the following command to establish the proper environment:

```
"<install-dir>\bin\compilervars.bat" arch [vs]
```

Where *arch* is one of followings as appropriate for the target architecture you want to build for:

- ia32
- ia32_intel64
- intel64

vs is optional and can be one of followings. If *vs* is not specified, the version of Visual Studio specified at installation time for command-line integration is used by default.

- vs2010
- vs2008
- vs2005

If you also have Intel® Visual Fortran Composer XE 2011 installed, this command will also establish the environment for using that compiler.

The script file names iclvars.bat and ifortvars.bat have been retained for compatibility with previous releases.

3.4.2 OpenMP* Legacy Libraries Removed

The OpenMP “legacy” libraries have been removed in this release. Only the “compatibility” libraries are provided.

3.4.3 OpenMP* Libraries Default to Dynamic Linking

As of version 11.0, OpenMP applications link to the dynamic OpenMP libraries by default. To specify static linking of the OpenMP libraries, specify `/Qopenmp-link:static` . The static libraries are deprecated and may be removed in a future major release.

3.4.4 Using Intel C++ Projects with a Source Control System

If your project is managed under a source control system, for example, Microsoft Visual Source Safe* or Microsoft Visual Studio Team Foundation Server*, there are additional steps you must

follow in order to use the Intel C++ project system with your project. A detailed article on this topic is available at <http://software.intel.com/en-us/articles/tips-on-using-the-intel-c-compiler-with-source-code-control-software/>

3.5 Known Issues

3.5.1 Compiler Known Issues

3.5.1.1 *Command-Line Diagnostic Issue for Filenames with Japanese Characters*

The filename in compiler diagnostics for filenames containing Japanese characters may be displayed incorrectly when compiled within a Windows command shell using the native Intel® 64 architecture compiler. It is not a problem when using Visual Studio or when using the Intel® 64 architecture cross-compiler or IA-32 architecture compiler

3.5.2 Visual Studio Known Issues

3.5.2.1 *Visual Studio 2010 sets default of /fp:precise*

A project created in or converted to Visual Studio 2010 will have the command line option `/fp:precise` set by default. This option sets the “floating point model” to improve consistency for floating point operations by disabling certain optimizations, reducing performance. To set the option back to the Intel default of `/fp:fast`, change the project property C++ > Optimization > Floating Point Model to Fast.

3.5.2.2 *Custom Build Rules in Microsoft Visual Studio 2005**

The Intel® C++ integration into Microsoft Visual Studio 2005* includes support for the Visual C++ Custom Build Rule functionality with some limitations. Custom Build Rules allow you to add custom tool invocations to your build process. See the Visual C++ documentation for a detailed explanation of Custom Build Rules.

You must create your custom build rules before converting your Visual C++ project to the Intel project system. If you need to modify your custom build rules after converting your project to the Intel project system, you must convert your project to the Visual C++ project system, make your custom build rule changes, and then convert the project back to the Intel project system.

The Macro lists that are available when you change the string properties defined for a Custom Build Rule don't contain Intel-specific macros (such as `$(icInstallDir)`; `$(icIDEInstallDir)`; `$(icProjectExt)`; `$(icProjectFileName)`). However, you can use the Intel-specific macros in Custom Build Rule property values and they will be expanded correctly.

The Intel® C++ integration into Visual Studio 2005 does not currently support the Visual C++ Tool Build Order functionality which is available from the Visual C++ Tool Build Order dialog box. That is, you cannot change the order in which the build steps are run.

3.5.2.3 Language packs of Visual Studio 2010

If you install a new language pack of Visual Studio 2010 after installing the Intel C++ Composer XE 2011, you may not see the Intel C++ Compiler specific options in the Project Property dialog. Please try the following to fix the issue:

- 1) if directory "<program files>\MSBuild\Microsoft.Cpp\v4.0\Platforms\[Win32|x64]\PlatformToolsets\Intel C++ Compiler XE 12.0\1033" exists, copy all files to "<program files>\MSBuild\Microsoft.Cpp\v4.0\Platforms\[Win32|x64]\PlatformToolsets\Intel C++ Compiler XE 12.0\<locale-ID>".
- 2) if directory "<program files>\MSBuild\Microsoft.Cpp\v4.0\Platforms\[Win32|x64]\PlatformToolsets\v100\1033\" exists, copy all files to "<program files>\MSBuild\Microsoft.Cpp\v4.0\Platforms\[Win32|x64]\PlatformToolsets\v100\<locale-ID>".

* The <locale-ID> is the language pack.

Another method is to uninstall the Intel C++ Composer XE 2011 and reinstall the Intel C++ Composer XE 2011.

3.5.3 Intel® Cilk™ Plus Known Issues

- 1) Microsoft C++ Structured Exception Handling (SEH) will fail if an SEH exception is thrown after a steal occurs and before the corresponding `_Cilk_sync`.
- 2) A `cilk_spawn` under the `if` in an `if/else` conditional statement may result in a compilation error.

The Intel C++ Compiler will complain about the following code:

```
if (expr)
    cilk_spawn a();
else
    b();

test.cpp
test.cpp(12): error: expected a statement
    else
    ^
```

The work-around is to add `{}` around `"cilk_spawn"` like below:

```
if (expr) {
    cilk_spawn a();
} else
    b();
```

3.5.4 Guided Auto-Parallel Known Issues

Guided Auto Parallel (GAP) analysis for single file, function name or specific range of source code does not work when Whole Program Interprocedural Optimization (/Qipo) is enabled. The workaround is to disable /Qipo – in Visual Studio, this is Project > [projectname] Properties > C++ > Optimization > Interprocedural Optimization > No.

3.5.5 Static Security Analysis Known Issues

3.5.5.1 Excessive false messages on C++ classes with virtual functions

Note that use of the Static Security Analysis feature also requires the use of Intel® Inspector XE.

Static security analysis reports a very large number of incorrect diagnostics when processing any program that contains a C++ class with virtual functions. In some cases the number of spurious diagnostics is so large that the result file becomes unusable.

If your application contains this common C++ source construct, add the following command line switch to suppress the undesired messages: /Qdiag-disable:12020,12040 (Windows) or -diag-disable 12020,12040 (Linux). **This switch must be added at the link step because that is when static security analysis results are created.** Adding the switch at the compile step alone is not sufficient. In Microsoft Visual Studio, add this to the property page Linker > Command Line.

If you are using a build specification to perform static security analysis, add the -disable-id 12020,12040 switch to the invocation of the inspxe-runsc, for example,
inspxe-runsc -spec-file mybuildspec.spec -disable-id 12020,12040

If you have already created a static security analysis result that was affected by this issue and you are able to open that result in the Intel® Parallel Inspector XE GUI, then you can hide the undesired messages as follows:

- The messages you will want to suppress are “Arg count mismatch” and “Arg type mismatch”. For each problem type, do the following:
- Click on the undesired problem type in the Problem filter. This hides all other problem types.
- Click on any problem in the table of problem sets
- Type control-A to select all the problems
- Right click and select *Change State -> Not a problem* from the pop-up menu to set the state of all the undesired problems
- Reset the filter on problem type to All
- Repeat for the other unwanted problem type
- Set the Investigated/Not investigated filter to *Not investigated*. You may have to scroll down in the filter pane to see it as it is near the bottom. This hides all the undesired messages because the “Not a problem” state is considered a “not investigated” state.

4 Intel® Integrated Performance Primitives

This section summarizes changes, new features and late-breaking news about this version of Intel® Integrated Performance Primitives (Intel® IPP). For detailed information about IPP see the following links:

- **New features:** see the information below and visit the main Intel IPP product page on the Intel web site at: <http://www.intel.com/software/products/ipp>.
- **Documentation, help, and samples:** see the documentation links on the IPP product page at: <http://www.intel.com/software/products/ipp>.

4.1 Intel® Integrated Performance Primitives 7.0

4.1.1 New and Changed Features

- Additional optimizations for the 256-bit AVX SIMD instruction set (available on Intel® processors code named "Sandy Bridge") have been incorporated.
- Further AES-NI optimizations have been applied to the cryptography domain ([separate download, see below](#)) and data compression (CRC32 for `ipp_bzip2`), substantially improving performance on those processors that support the AES-NI instructions.
- Microsoft* Visual Studio 2010 is now supported by the Windows edition of the IPP library; meaning, IPP help files and project files are compatible with the Visual Studio 2010 IDE. The Visual Studio 2005 and 2008 IDEs are also supported.
- Support for the JPEG-XR (HD Photo) forward and inverse transforms for 16s, 32s and 32f data types and variable length code (VLC) encode and decode functions for 32s data types has been added.
- A JPEG-XR (HD Photo) codec is now included in the IPP UIC sample framework for grayscale, RGB and RGBA images with 8, 16, and 32-bit integer and 16 and 32-bit floating point pixel depths.
- The DMIP sample now includes a Microsoft DSL (Domain Specific Language) add-on for use with Visual Studio 2008.
- A new *interfaces* directory has been added that contains high-level application code, in the form of source and pre-built binaries. Several popular data compression libraries (e.g., `bzip2`, `zlib` and `gzip`) have been modified for use with the IPP library and can be found in the *interfaces* directory for immediate use.
- There is a new `ipp_lzopack` (data compression) library, located in the *interfaces* directory mentioned above, as part of this release.
- Multi-threading is now part of the `ipp_zlib` library (by use of the OpenMP multi-threading library).
- A new directory hierarchy has been established to simplify integration of the Intel IPP library with the Intel Compiler products. This change may require that you update your build scripts, makefiles and/or Visual Studio project files.
- Directories formerly designated as "em64t" are now designated by the "intel64" tag. This change may require that you update your build scripts, makefiles and/or Visual Studio project files.

- Library filenames have been normalized to be consistent between 32-bit and 64-bit architectures (i.e., the "em64t" tag has been removed from all 64-bit library file names). This change may require that you update your build scripts, makefiles and/or Visual Studio project files.
- The domain-specific "emerged" and "merged" static library files have been combined for simpler reference (e.g., ippsemerged.lib + ippmerged_t.lib ⇒ ippsemerged_t.lib) and the single-threaded static libraries are now designated by a "_l" suffix (multi-threaded static libraries continue to be designated with a "_t" suffix). This change may require that you update your build scripts, makefiles and/or Visual Studio project files.
- The speech recognition functions (ippSR domain) are not part of this release; this domain will continue to be supported in the IPP 6.1 product.
- Intel® Itanium® architecture (IA-64) support is not included in this release. Intel® IPP 6.1 is the latest release for the IA-64 architecture.
- The SPIRAL generated functions (ippGEN domain) are now distributed as a separate download. See [instructions below](#) for more information.

4.1.2 Directory Structure Changed

A new directory hierarchy has been established to simplify integration with the Intel® compiler products. This may require action on your part in order to be able to continue building applications that use Intel® Integrated Performance Primitives.

- If you used the “Select Build Components” dialog for the Intel® performance libraries with earlier versions of Intel® Parallel Composer or Intel® C++ Compiler, right click on your C++ project in Microsoft Visual Studio, select Intel Parallel Composer > Select Build Components, then click OK to update the paths.
- If you use the Tools > Options > Intel Parallel Composer > Compilers dialog to change the version of compiler you are using, you must also update the Build Components paths as above.

4.2 Intel® IPP Cryptography Libraries are Available as a Separate Download

The Intel® IPP cryptography libraries are available as a separate download. For download and installation instructions, please read

<http://software.intel.com/en-us/articles/download-ipp-cryptography-libraries/>

4.3 SPIRAL Generated Functions are Available as a Separate Download

The *SPIRAL generated functions*, a special subset of the Signal Processing domain of the Intel IPP library that was added in version 6.0 of the library, have been moved to a separately installed add-on library. This change was made to reduce the overall download size of the IPP library. The functions contained within the ippGEN domain all begin with the ippg prefix and are listed below using an abbreviated naming format.

- ippgenGetLibVersion
- ippgDCT4_[32f|64f]
- ippgDCT4Init_[32f|64f]

- `ippgDCT4InitAlloc_[32f|64f]`
- `ippgDCT4Free_[32f|64f]`
- `ippgDCT4GetSize_[32f|64f]`

- `ippgDFTFwd_CToC_[32fc|64fc]`
- `ippgDFTFwd_CToC_*_[32fc|64fc]`
- `ippgDFTInv_CToC_[32fc|64fc]`
- `ippgDFTInv_CToC_*_[32fc|64fc]`

- `ippgHartley_[32f|64f]`
- `ippgHartley_*_[32f|64f]`

You will find the download for the SPIRAL generated functions alongside the compiler download at the [Intel® Software Development Products Registration Center](http://www.intel.com/software/products/registration).

4.4 Intel® IPP Code Samples

The Intel® IPP code samples are organized into downloadable packages at <http://www.intel.com/software/products/ipp>

The samples include source code for audio/video codecs, image processing and media player applications, and for calling functions from C++, C# and Java*. Instructions on how to build the sample are described in a readme file that comes with the installation package for each sample.

5 Intel® Math Kernel Library

This section summarizes changes, new features and late-breaking news about this version of the Intel® Math Kernel Library.

5.1 Changes in This Version

5.1.1 Changes in Initial Release

- 1) BLAS
 - New functions for computing 2 matrix-vector products at once: [D/S]GEM2VU, [Z/C]GEM2VC
 - New functions for computing mixed precision general matrix-vector products: [DZ/SC]GEMV
 - New function for computing the sum of two scaled vectors: *AXPBY
 - Intel® AVX optimizations in key functions: SMP LINPACK, level 3 BLAS, DDOT, DAXPY
- 2) LAPACK
 - New C interfaces for LAPACK supporting row-major ordering
 - Integrated Netlib LAPACK 3.2.2 including one new computational routine (*GEQRF) and two new auxiliary routines (*GEQR2P and *LARFGP) and the earlier LAPACK 3.2.1 update
 - Intel® AVX optimizations in key functions: DGETRF, DPOTRF, DGEQRF
- 3) PARDISO

- Improved performance of factor and solve steps in multi-core environments
 - Introduced the ability to solve for sparse right-hand sides and perform partial solves— produces partial solution vector
 - Improved performance of the out-of-core (OOC) factorization step
 - Support for zero-based (C-style) array indexing
 - Zeros on the diagonal of the matrix are no longer required in sparse data structures for symmetric matrices
 - New ILP64 PARDISO interface allows the use of both LP64 and ILP64 versions when linked to the LP64 libraries
 - The memory required for storing files on the disk in OOC mode can now be estimated just after reordering
- 4) Sparse BLAS
- Format conversion functions now support all data types (single and double precision for real and complex data) and can return sorted or unsorted arrays
- 5) FFTs
- New MPI FFTW 3.3alpha1 wrappers cover new cluster functionality
 - Improved load-balancing of cluster FFTs provides improved performance
 - Intel AVX optimizations in all 1D/2D/3D FFTs
 - Improved performance of 2D and 3D mixed-radix FFTs for single and double precision data for all systems supporting the SSE4.2 instruction set
 - Support for split-complex data represented as two real arrays introduced for 2D/3D FFTs
 - Support for 1D complex-to-complex transforms of large prime lengths
 - Introduced Hybrid parallelism (MPI + OpenMP*) on cluster 1D complex transforms and increased performance on vector lengths which are a multiple of the number of MPI processes
- 6) VML
- A new function for computing $(ax+b)/(cy+d)$ where a, b, c, and d are scalars, and x and y are real vectors: `v[s/d]LinearFrac()`
 - Intel AVX optimizations for real functions
 - A new mode for setting denormals to zero, overflow support for complex vectors, and for every VML function a new function with an additional parameter for setting the accuracy mode
- 7) VSL
- A set of new Summary Statistics functions was added covering basic statistics, covariance and correlation, pooled, group, partial, and robust covariance/correlation, quantiles and streaming quantiles, outliers detection algorithm, and missing values support
 - Performance optimized algorithms: MI algorithm for support of missing values, TBS algorithm for computation of robust covariance, BACON algorithm for detection of outliers, ZW algorithm for computation of quantiles (streaming data case), and 1PASS algorithm for computation of pooled covariance
 - Improved performance of SFMT19937 Basic Random Number Generator (BRNG)
 - Intel® AVX optimizations: MT19937 and MT2203 BRNGs

- 8) Documentation: Product documentation is available in the Microsoft Help Viewer* 1.x format that integrates with Microsoft Visual Studio* 2010
- 9) Added runtime dispatching dynamic libraries allowing link to a single interface library which loads dependent libraries dynamically at runtime depending on runtime CPU detection and/or library function calls
- 10) A new directory structure has been established to simplify integration of Intel MKL with the Intel® Parallel Studio XE family of products and directories formerly designated as "em64t" are now designated by the "intel64" tag
- 11) Intel® Itanium® architecture (IA-64) support is not included in this release. Intel® MKL 10.2 is the latest release for IA-64
- 12) The sparse solver functionality has been fully integrated into the core Intel MKL libraries and the libraries with "solver" in the filename have been removed from the product

5.1.2 Changes in Update 1

- 1) PARDISO/DSS: Added true F90 overloaded API (see the Intel MKL reference manual for more information)
- 2) PARDISO: Improved the statistical reporting to be more reader friendly
- 3) Sparse BLAS: Improved performance of ?BSRMM functions on the latest Intel® processors
- 4) FFTs: Support for negative strides
- 5) FFT examples: Added examples for split-complex FFTs in C and Fortran using both the DFTI and FFTW3 interfaces
- 6) VML: Improved performance of real in-place Add/Sub/Mul/Sqr functions on systems supporting SSE2 and SSE3
- 7) Poisson Library: Changed the default behavior of the Poisson library functions from sequential to threaded operation
- 8) Bug fixes: <http://software.intel.com/en-us/articles/intel-mkl-103-bug-fixes/>

5.2 Attributions

As referenced in the End User License Agreement, attribution requires, at a minimum, prominently displaying the full Intel product name (e.g. "Intel® Math Kernel Library") and providing a link/URL to the Intel® MKL homepage (www.intel.com/software/products/mkl) in both the product documentation and website.

The original versions of the BLAS from which that part of Intel® MKL was derived can be obtained from <http://www.netlib.org/blas/index.html>.

The original versions of LAPACK from which that part of Intel® MKL was derived can be obtained from <http://www.netlib.org/lapack/index.html>. The authors of LAPACK are E. Anderson, Z. Bai, C. Bischof, S. Blackford, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, and D. Sorensen. Our FORTRAN 90/95 interfaces to LAPACK are similar to those in the LAPACK95 package at <http://www.netlib.org/lapack95/index.html>. All interfaces are provided for pure procedures.

The original versions of ScaLAPACK from which that part of Intel® MKL was derived can be obtained from <http://www.netlib.org/scalapack/index.html>. The authors of ScaLAPACK are

L. S. Blackford, J. Choi, A. Cleary, E. D'Azevedo, J. Demmel, I. Dhillon, J. Dongarra, S. Hammarling, G. Henry, A. Petitet, K. Stanley, D. Walker, and R. C. Whaley.

PARDISO in Intel® MKL is compliant with the 3.2 release of PARDISO that is freely distributed by the University of Basel. It can be obtained at <http://www.pardiso-project.org>.

Some FFT functions in this release of Intel® MKL have been generated by the SPIRAL software generation system (<http://www.spiral.net/>) under license from Carnegie Mellon University. Some FFT functions in this release of the Intel® MKL DFTI have been generated by the UHFFT software generation system under license from University of Houston. The Authors of SPIRAL are Markus Puschel, Jose Moura, Jeremy Johnson, David Padua, Manuela Veloso, Bryan Singer, Jianxin Xiong, Franz Franchetti, Aca Gacic, Yevgen Voronenko, Kang Chen, Robert W. Johnson, and Nick Rizzolo.

6 Intel® Threading Building Blocks

For information on changes in this version of Intel® Threading Building Blocks, please read the file `CHANGES` in the TBB documentation directory.

6.1 Known Issues

Please note the following with respect to this particular release of Intel Threading Building Blocks.

6.1.1 Library Issues

- To allow more accurate results to be obtained with Intel® Thread Checker or Intel® Thread Profiler, download the latest update releases of these products before using them with Intel Threading Building Blocks.
- If you are using Intel Threading Building Blocks and OpenMP* constructs mixed together in rapid succession in the same program, and you are using Intel compilers for your OpenMP* code, set `KMP_BLOCKTIME` to a small value (e.g., 20 milliseconds) to improve performance. This setting can also be made within your OpenMP* code via the `kmp_set_blocktime()` library call. See the Intel compiler OpenMP* documentation for more details on `KMP_BLOCKTIME` and `kmp_set_blocktime()`.
- In general, non-debug ("release") builds of applications or examples should link against the non-debug versions of the Intel Threading Building Blocks libraries, and debug builds should link against the debug versions of these libraries. On Windows systems, compile with `/MD` and use Intel Threading Building Blocks release libraries, or compile with `/MDd` and use debug libraries; not doing so may cause run-time failures. See the Tutorial in the product "Documentation" sub-directory for more details on debug vs. release libraries.

7 Intel® Parallel Debugger Extension

This section summarizes changes, new features and late-breaking news about this version of the Intel® Parallel Debugger Extension.

7.1 New Features

- Intel® Cilk™ Plus support
 - Serialized execution of an Intel® Cilk™ Plus program at debug-time, without re-compilation.
 - Intel® Cilk™ Plus worker threads are clearly marked in the Visual Studio Debugger
 - Display of stealing points and current Cilk Plus workers on Cilk Plus Thread stack window
- Threads Window
 - Improved Data Sharing Detection
 - Support for OpenMP* 3.0
 - Support for Windows OS synchronization functions
 - Improved data sharing detection analysis performance

7.2 Known Issues

- If you are using Microsoft Visual Studio 2005, there are six Intel-specific exceptions that must be enabled manually. Select `Debug > Exceptions`, expand the `Win32 Exceptions` tree, and enable items:

```
a1a01db0 Intel Parallel Debugger Extension Exception 0
a1a01db1 Intel Parallel Debugger Extension Exception 1
a1a01db2 Intel Parallel Debugger Extension Exception 2
a1a01db3 Intel Parallel Debugger Extension Exception 3
a1a01db4 Intel Parallel Debugger Extension Exception 4
a1a01db5 Intel Parallel Debugger Extension Exception 5
```

This needs to be done once per project.

- Disabling the Intel Debugging exceptions during a debug session may cause Visual Studio (up to Visual Studio 2008, SP1) to hang.
- Use of the Intel Parallel Debugger Extension requires that the OpenMP library be linked dynamically, which is the default. If you wish to use the Parallel Debugger Extension, do not use `/Qopenmp-link:static` to specify static linking of the OpenMP Library.
- Be sure to enable the parallel debug instrumentation (switch `/debug:parallel`) before you start parallel debugging: `Project > [project name] Properties > Configuration Properties > C/C++ > Debug > Enable Parallel Debug Checks: Yes (/debug:parallel)`. Otherwise, the debugger will not detect data sharing events nor break on re-entrant calls.

- Microsoft Visual Studio 2010 users: If you have specified that the project be built with Microsoft Visual C++, and then change the project to use the Intel C++ compiler, you must close and re-open the solution for the Parallel Debug Environment property setting "Auto" to be recognized.
- If you are using Microsoft Visual Studio 2008 and debugging 64-bit applications, you must have Visual Studio 2008 Service Pack 1 installed.
 - You can debug 64-bit applications under Visual Studio 2005 and 2008 without Service Packs only if they are linked to the low memory area. If not linked to the low memory area, you will not see any events until the debuggee terminates. After termination, all events are displayed in the event window. In order to debug 64-bit applications properly, set the base address to `0x10000` in `Project > Properties > Linker > Advanced`.
- Function local or heap variables are displayed as "???" in the data sharing event window.
- The SSE Registers window does not work for 64-bit applications - the window shows "???"
- Filters on static local variables are not set correctly via context menu.
- Reentrant call detection stops in Disassembly view. It does not work correctly for static functions. When used in design mode, the function should be preceded by a suitable context operator, for example, `{, , myapp.exe} my_extern_function`
- The debugger extension windows remain empty when their placement is changed from "docked" to "floating". The workaround is to either keep them docked or to restart the debug session after the placement was changed.
- The debugger extension requires the application to be started from Visual Studio. It does not work when attaching to an existing process.
- Windows settings are restored to default (Hexadecimal) when the window is hidden or closed and reopened again.

7.3 Documentation

Intel Parallel Debugger Extension Documentation can be accessed via the Help menu of Microsoft Visual Studio or by pressing the function key F1 after activation of a Parallel Debugger Extension window. Help is also available by clicking the link "HTML version" inside `debugger-documentation.htm`.

8 Disclaimer and Legal Information

INFORMATION IN THIS DOCUMENT IS PROVIDED IN CONNECTION WITH INTEL(R) 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. UNLESS OTHERWISE AGREED IN WRITING BY INTEL, THE INTEL PRODUCTS ARE NOT DESIGNED NOR INTENDED FOR ANY APPLICATION IN WHICH THE FAILURE OF THE INTEL PRODUCT COULD CREATE A SITUATION WHERE PERSONAL INJURY OR DEATH MAY OCCUR.

Intel may make changes to specifications and product descriptions at any time, without notice. Designers must not rely on the absence or characteristics of any features or instructions marked "reserved" or "undefined." Intel reserves these for future definition and shall have no responsibility whatsoever for conflicts or incompatibilities arising from future changes to them. The information here is subject to change without notice. Do not finalize a design with this information.

The products described in this document may contain design defects or errors known as errata which may cause the product to deviate from published specifications. Current characterized errata are available on request.

Contact your local Intel sales office or your distributor to obtain the latest specifications and before placing your product order.

Copies of documents which have an order number and are referenced in this document, or other Intel literature, may be obtained by calling 1-800-548-4725, or go to:
<http://www.intel.com/design/literature.htm>

Intel processor numbers are not a measure of performance. Processor numbers differentiate features within each processor family, not across different processor families. Go to:

<http://www.intel.com/products/processor%5Fnumber/>

Celeron, Centrino, Intel, Intel logo, Intel386, Intel486, Intel Atom, Intel Core, Itanium, MMX, Pentium, VTune, and Xeon are trademarks of Intel Corporation in the U.S. and other countries.

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

Copyright © 2010 Intel Corporation. All Rights Reserved.