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

Document number: 321412-003US
12 January 2011

Table of Contents

1 Introduction

This document describes how to install the product, provides a summary of new and changed 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 2 (2011.2)

- Intel® Math Kernel Library updated to 10.3 Update 2

- Intel® Integrated Performance Primitives 7.0 Update 2
- Intel® Threading Building Blocks 3.0 Update 5
- 3 intrinsics changed in immintrin.h
- Utility “inspxe-runsc” changed
- Corrections to reported problems

Update 1 (2011.1)

- Intel® Math Kernel Library updated to 10.3 Update 1
- Corrections to reported problems

Product Release (2011.0)

- Initial product release

1.2 Product Contents

*Intel® C++ Composer XE 2011 Update2 for Linux** includes the following components:

- Intel® C++ Compiler XE 12.0 Update 2 for building applications that run on IA-32 and Intel® 64 architecture systems running the Linux* operating system
- Intel® Debugger 12.0 Update 2
- Intel® Integrated Performance Primitives 7.0 Update 1
- Intel® Math Kernel Library 10.3 Update 2
- Intel® Threading Building Blocks 3.0 Update 5
- Integration into the Eclipse* development environment
- 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/

Requirements to develop IA-32 architecture applications

- 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)
    - Development for a target different from the host may require optional library components to be installed from your Linux Distribution.
    - For the best experience, a multi-core or multi-processor system is recommended
- 1GB of RAM (2GB recommended)
- 2GB free disk space for all features
- One of the following Linux distributions (this is the list of distributions tested by Intel; other distributions may or may not work and are not recommended - please refer to Technical Support if you have questions):
    - Asianux* 3.0

    - Fedora* 12, 13
    - Red Hat Enterprise Linux* 4, 5, 6
    - SUSE LINUX Enterprise Server* 10, 11
    - Ubuntu* 10.04
    - Debian* 5.0
- Linux Developer tools component installed, including gcc, g++ and related tools
- Library libunwind.so is required in order to use the `-traceback` option. Some Linux distributions may require that it be obtained and installed separately.
- If developing on an Intel® 64 architecture system, some Linux distributions may require installation of one or more of the following additional Linux components: ia32-libs, lib32gcc1, lib32stdc++6, libc6-dev-i386, gcc-multilib

Requirements to develop Intel® 64 architecture applications

- A PC based on an Intel® 64 architecture processor (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 of RAM (2GB recommended)
- 2GB free disk space for all features
- 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 for the installed distribution of Linux
- One of the following Linux distributions (this is the list of distributions tested by Intel; other distributions may or may not work and are not recommended - please refer to Technical Support if you have questions):
    - Asianux* 3.0
    - Fedora* 12, 13
    - Red Hat Enterprise Linux* 4, 5, 6
    - SUSE LINUX Enterprise Server* 10.2, 11.1 SP1
    - Ubuntu* 10.04
    - Debian* 5.0
- Linux Developer tools component installed, including gcc, g++ and related tools
- Library libunwind.so is required in order to use the `-traceback` option. Some Linux distributions may require that it be obtained and installed separately.

Additional requirements to use the Graphical User Interface of the Intel® Debugger

- Java* Runtime Environment (JRE) 5.0 (also called 1.5) or 6.0 (1.6) – 5.0 recommended
    - A 32-bit JRE must be used on an IA-32 architecture system and a 64-bit JRE must be used on an Intel® 64 architecture system

Notes

- The Intel compilers are tested with a number of different Linux distributions, with different versions of gcc. Some Linux distributions may contain header files different from those we have tested, which may cause problems. The version of glibc you use must be

consistent with the version of gcc in use. For best results, use only the gcc versions as supplied with distributions listed above.

- 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.
- Compiling very large source files (several thousands of lines) using advanced optimizations such as -O3, -ipo and -openmp, may require substantially larger amounts of RAM.
- The above lists of processor model names are not exhaustive - other processor models correctly supporting the same instruction set as those listed are expected to work. Please refer to Technical Support if you have questions regarding a specific processor model
- Some optimization options have restrictions regarding the processor type on which the application is run. Please see the documentation of these options for more information.

1.3.1 Red Hat Enterprise Linux* 4 Support Deprecated

In a future major release of Intel® C++ Composer XE, support will be removed for installation and use on Red Hat Enterprise Linux 4. Intel recommends migrating to a newer version of these operating systems.

1.3.2 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 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-linux/

1.6 Technical Support

Register your license 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

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, mount the DVD, change the directory (`cd`) to the top-level directory of the mounted DVD and begin the installation using the command:

```
./install.sh
```

If you received the product as a downloadable file, first unpack it into a writeable directory of your choice using the command:

```
tar -xzvf name-of-downloaded-file
```

Then change the directory (`cd`) to the directory containing the unpacked files and begin the installation using the command:

```
./install.sh
```

Follow the prompts to complete 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.

2.1.1 Activation of Purchase after Evaluation Using the Intel Activation Tool

Note for evaluation customers a new tool Intel Activation Tool "`Activate`" is included in this product release and installed at `/opt/intel/ActivationTool/Activation/` directory.

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 `/opt/intel/ActivationTool/Activation/Activate`. It will convert your evaluation software to a fully licensed product. To use the tool:

```
$ /opt/intel/ActivationTool/Activation/Activate [SN_Num_here]
```

2.1.2 Silent Install

For information on automated or “silent” install capability, please see http://software.intel.com/en-us/articles/intel-compilers-for-linux-silent-installation-guides/.

2.1.3 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.

2.1.4 Eclipse* Integration Installation

Please refer to the section below on Eclipse Integration

2.1.5 Known Installation Issues

- If you have enabled the Security-Enhanced Linux (SELinux) feature of your Linux distribution, you must change the `SELINUX` mode to `permissive` before installing the Intel C++ Compiler. Please see the documentation for your Linux distribution for details. After installation is complete, you may reset the `SELINUX` mode to its previous value.
- On some versions of Linux, auto-mounted devices do not have the "exec" permission and therefore running the installation script directly from the DVD will result in an error such as:

  ```
  bash: ./install.sh: /bin/bash: bad interpreter: Permission denied
  ```

  If you see this error, remount the DVD with exec permission, for example:

  ```
  mount /media/<dvd_label> -o remount,exec
  ```

  and then try the installation again.
- The product is fully supported on Ubuntu and Debian Linux distributions for IA-32 and Intel® 64 architecture systems as noted above under System Requirements. Due to a restriction in the licensing software, however, it is not possible to use the Trial License feature when evaluating IA-32 components on an Intel® 64 architecture system under Ubuntu or Debian. This affects using a Trial License only. Use of serial numbers, license files, floating licenses or other license manager operations, and off-line activation (with serial numbers) is not affected. If you need to evaluate IA-32 components of the product on an Intel® 64 architecture Ubuntu or Debian system, please visit the Intel® Software Evaluation Center (http://www.intel.com/cd/software/products/asmo-na/eng/download/eval/) to obtain an evaluation serial number.

2.2 Installation Folders

The compiler installs, by default, under `/opt/intel` – this is referenced as `<install-dir>` in the remainder of this document. You are able to specify a different location, and can also perform a “non-root” install in the location of your choice.

The directory organization has changed since the Intel® Compilers 11.1 release.

Under `<install-dir>` are the following directories:

- `bin` – contains symbolic links to executables for the latest installed version
- `lib` – symbolic link to the lib directory for the latest installed version
- `include` – symbolic link to the include directory for the latest installed version
- `man` – symbolic link to the directory containing man pages for the latest installed version
- `ipp` – symbolic link to the directory for the latest installed version of Intel® Integrated Performance Primitives
- `mkl` – symbolic link to the directory for the latest installed version of Intel® Math Kernel Library
- `tbb` – symbolic link to the directory for the latest installed version of Intel® Threading Building Blocks
- `composerxe` – symbolic link to the `composerxe-2011` directory
- `composerxe-2011` – directory containing symbolic links to subdirectories for the latest installed Intel® Composer XE 2011 compiler release
- `composerxe-2011-<n>.<pkg>` - physical directory containing files for a specific compiler version. `<n>` is the update number, and `<pkg>` is a package build identifier.

Each `composerxe-2011` directory contains the following directories that reference the latest installed Intel® Composer XE 2011 compiler:

- `bin` – directory containing scripts to establish the compiler environment and symbolic links to compiler executables for the host platform
- `pkg_bin` – symbolic link to the compiler `bin` directory
- `include` – symbolic link to the compiler `include` directory
- `lib` – symbolic link to the compiler `lib` directory
- `ipp` – symbolic link to the `ipp` directory
- `mkl` – symbolic link to the `mkl` directory
- `tbb` – symbolic link to the `tbb` directory
- `debugger` – symbolic link to the `debugger` directory
- `eclipse_support` – symbolic link to the `eclipse_support` directory
- `man` – symbolic link to the `man` directory
- `Documentation` – symbolic link to the `Documentation` directory
- `Samples` – symbolic link to the `Samples` directory

Each `composerxe-2011-<n>.<pkg>` directory contains the following directories that reference a specific update of the Intel® Composer XE 2011 compiler:

- `bin` – all executables
- `compiler` – shared libraries and header files
- `debugger` – debugger files

- `Documentation` – documentation files
- `man` – `man` pages
- `eclipse_support` – files to support Eclipse integration
- `ipp` – Intel® Integrated Performance Primitives libraries and header files
- `mkl` – Intel® Math Kernel Library libraries and header files
- `tbb` – Intel® Threading Building Blocks libraries and header files
- `Samples` – Product samples and tutorial files

If you have both the Intel C++ and Intel Fortran compilers installed, they will share folders for a given version and update.

This directory layout allows you to choose whether you want the latest compiler, no matter which version, the latest update of the Intel® Composer XE 2011 compiler, or a specific update. Most users will reference `<install-dir>/bin` for the `compilervars.sh [.csh]` script, which will always get the latest compiler installed. This layout should remain stable for future releases.

2.3 Removal/Uninstall

Removing (uninstalling) the product should be done by the same user who installed it (root or a non-root user). If `sudo` was used to install, it must be used to uninstall as well. It is not possible to remove the compiler while leaving any of the performance library or Eclipse* integration components installed.

1. Open a terminal window and set default (`cd`) to any folder outside `<install-dir>`
2. Type the command: `<install-dir>/bin /uninstall_cproc.sh` (substitute `intel64` or `ia64` for `ia32` as desired)
3. Follow the prompts
4. Repeat steps 2 and 3 to remove additional platforms or versions

If you have the same-numbered version of Intel® Fortran Compiler installed, it may also be removed.

If you have added the Intel C++ Eclipse integration to an instance of Eclipse in your environment, you will need to update your Eclipse configuration by removing the Intel integration extension site from your Eclipse configuration. To do this, Go to Help > About Eclipse and click on "Installation Details". Select "Intel(R) C++ Compiler XE 12.0 for Linux* OS " under "Installed Software" and click on "Uninstall..." Click "Finish". When asked to restart Eclipse, select "Yes".

3 Intel® C++ Compiler

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

3.1 Compatibility

In version 11.0, the IA-32 architecture default for code generation 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.

- 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 Three intrinsics changed in update 2

Three intrinsics (_rdrand16_step(), _rdrand32_step(), _rdrand64_step()) have been changed in update 2. The documentation has not been updated with these new changes. These intrinsic return a hardware-generated random value and are declared in the “immintrin.h” header file.

These three intrinsics are mapped to a single RDRAND instruction, generate random numbers of 16/32/64 bit wide random integers.

Syntax

1. extern int _rdrand16_step(unsigned short *random_val);
2. extern int _rdrand32_step(unsigned int *random_val);
3. extern int _rdrand64_step(unsigned __int64 *random_val);

Description

The intrinsics perform one attempt to generate a hardware generated random value using the instruction RDRAND. The generated random value is written to the given memory location and the success status is returned: 1 if the hardware returned a valid random value and 0 otherwise.

Return

A hardware-generated 16/32/64 random value.

Constraints

The _rdrand64_step() intrinsic can be used only on systems with the 64-bit registers support.

3.2.2 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, `-diag-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.

3.2.2.1 The command line utility “inspxe-runsc” changed since update 2

This utility is distributed with Intel® Composer XE 2011 and has been changed since update 2. This change only affects users who use Composer XE 2011 to perform Static Security Analysis (SSA). Those that do not use SSA and those that perform SSA without using this utility are unaffected. SSA is only available to users of Intel® Parallel Studio XE 2011 or Intel® C++ Studio XE 2011, so users who do not have those products are unaffected.

Inspxe-runsc executes a **build specification,** a description of how an application is built. Usually build specification files are generated by observing a build as it executes and recoding the compilations and links that are performed. Inspxe-runsc repeats these actions using the Intel compiler in SSA mode. SSA results are generated at the link step so a build specification that describes a build with more than one link step will generate more than one SSA result when inspxe-runsc is invoked.

The versions of inspxe-runsc included in Composer XE 2011 and Composer XE 2011 Update 1 generate all the SSA results in a single directory. In the multiple link case this violated the rule that all the SSA results for one and only one project must be created in the same directory. The updated version of inspxe-runsc respects this rule by generating results for each link step in a separate directory. The name of that directory is formed from the name of the file being linked.

Thus if a build specification describes a project that builds two executables, file1.exe and file2.exe, then earlier versions of inspxe-runsc would create two results, one for file1 and one for file2, say r000sc and r001sc, in the same directory. The new version of inspxe-runsc will also create two results, but the one for file1 will be created in “My Inspector XE results – file1\r000sc” and the one for file2 will be created in “My Inspector XE results – file2\r000sc”. The directories containing the results are both created in the same parent directory.

Inspxe-runsc has a command line switch, -result-dir (-r), that specifies where results are to be created. The meaning of this switch has changed. Previous this would name the directory where the result itself, say r000sc, would be created. Now it names the parent directory where the “My Inspector XE Results - name” directory or directories will be created. So the directory named in the –r switch is effectively two levels up from the results themselves.

The change to inspxe-runsc effectively moves the result directory, and user action is required to adapt to this change. Those using scripts that invoke inspxe-runsc with the –r switch must update their scripts to reflect the new interpretation of the –r switch argument described earlier. Users must move their old result files into the new directory so that SSA results produced by earlier versions of inspxe-runsc share the same directory as results produced by the new version of inspxe-runsc. Users that had been using inspxe-runsc with a build specification with only one link step should move their old results into a directory of the form “My Inspector XE results – name”. If this is not done, then all the problems in the newly created result will appear to be “New”. Users that had been using inspxe-runsc with a build specification with multiple link steps have been having various issues with SSA that will be resolved by using the new utility. Such users are best advised to copy the most recent into their old results into each of the new “My Inspector XE results – name” directories. This offers the best chance that some old problem state information will be correctly applied to new results when they are created in the future.

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.

- -ansi-alias-check
- -auto-p32
- -cilk-serialize
- -diag-sc-dir
- -ffriend-injection
- -fzero-initialized-in-bss
- -fimf-absolute-error
- -fimf-accuracy-bits
- -fimf-arch-consistency
- -fimf-max-error
- -fimf-precision
- -fp-trap
- -fp-trap-all

- -fvar-tracking
- -fvar-tracking-assignments
- -guide
- -guide-data-trans
- -guide-file
- -guide-file-append
- -guide-opts
- -guide-par
- -guide-vec
- -intel-extensions
- -opt-args-in-regs
- -opt-matmul
- -prof-value-profiling
- -profile-functions
- -profile-loops
- -regcall
- -simd
- -Wremarks
- -Wsign-compare
- -Wstrict-aliasing

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

3.4 Other Changes

3.4.1 Establishing the Compiler Environment

The `compilervars.sh` script is used to establish the compiler environment. `compilervars.csh` is also provided.

The command takes the form:

`source <install-dir>/bin/compilervars.sh` *`argument`*

Where *`argument`* is either `ia32` or `intel64` as appropriate for the architecture you are building for. Establishing the compiler environment also establishes the environment for the Intel® Debugger, Intel® Performance Libraries and, if present, Intel® Fortran Compiler.

3.4.2 Instruction Set Default Changed to Require Intel® Streaming SIMD Extensions 2 (Intel® SSE2)

When compiling for the IA-32 architecture, `-msse2` (formerly `-xW`) is the default. Programs built with `-msse2` in effect require that they be run on a processor that supports the Intel® Streaming SIMD Extensions 2 (Intel® SSE2), such as the Intel® Pentium® 4 processor and some non-Intel processors. No run-time check is made to ensure compatibility – if the program is run on an unsupported processor, an invalid instruction fault may occur. Note that this may change

floating point results since the Intel® SSE instructions will be used instead of the x87 instructions and therefore computations will be done in the declared precision rather than sometimes a higher precision.

All Intel® 64 architecture processors support Intel® SSE2.

To specify the older default of generic IA-32, specify `-mia32`

3.4.3 OpenMP* Legacy Libraries Removed

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

3.5 Compatibility with Previous Versions

This section summarizes changes in the C++ compiler that may present compatibility issues when mixing code compiled with previous versions of Intel Parallel Composer or Intel C++ Compiler with code compiled with this version of Intel Parallel Composer.

3.6 Known Issues

3.6.1 `__GXX_EXPERIMENTAL_CXX0X__` Macro Not Supported

In the Gnu* version 4.3 or later environments, using the `-std=c++0x` or `-std=gnu++0x` option may lead to a diagnostic of the form:

```
This file requires compiler and library support for the upcoming ISO
C++ standard, C++0x. This support is currently experimental, and must
be enabled with the -std=c++0x or -std=gnu++0x compiler options.
```

The Intel compiler does not currently define the `__GXX_EXPERIMENTAL_CXX0X__` macro in any mode, since it does not yet support some C++0x features (such as variadic templates) enabled by the macro in the C++ standard library headers. This may lead to incompatibilities with g++ when using the C++ standard library in the `-std=c++0x` or `-std=gnu++0x` modes. One such example is that the `va_copy` macro may not be defined in `stdarg.h`. This can be worked around by adding the compiler flag `-Dva_copy=__builtin_va_copy`.

3.6.2 Intel® Cilk™ Plus Known Issues

1) Link error "`undefined reference to `__cilkrts_*'`"

   If you are using a version of `binutils` prior to 2.17, you may get linker errors like below when using Intel® Cilk™ Plus code:

   ```
   undefined reference to `__cilkrts_get_tls_worker'
   ```

   This is because the Intel Cilk™ Plus runtime library cannot be automatically linked unless you have `binutils` 2.17 or above. The work-around is to update your copy of `binutils` or manually link in the runtime library using `-lcilkrts` on your linker command line.

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();
```

3.6.3 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 (-ipo) is enabled

3.6.4 TR1 System Headers

If you are using the TR1 (C++ Library Technical Report 1) system headers on a system with g++ version 4.3 or later installed, the Intel C/C++ compiler will give errors when it tries to compile the <type_traits> header file. This is because the Intel C/C++ compiler does not yet support the C++0x feature called variadic templates. You will see these types of compilation errors:

```
../include/c++/4.3.0/tr1_impl/type_traits(170): error: expected an
identifier
   template<typename _Res, typename... _ArgTypes>

                                      ^

include/c++/4.3.0/tr1_impl/type_traits(171): error: expected a ")"
    struct __is_function_helper<_Res(_ArgTypes...)>
```

There is no workaround, other than not using these headers or using an older version of the g++ compiler.

3.6.5 Static Security Analysis Known Issues

3.6.5.1 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.

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® Debugger (IDB)

The following notes refer to the Graphical User Interface (GUI) available for the Intel® Debugger (IDB) when running on IA-32 and Intel® 64 architecture systems. In this version, the `idb` command invokes the GUI – to get the command-line interface, use `idbc`.

4.1 Setting up the Java* Runtime Environment

The Intel® IDB Debugger graphical environment is a Java application and requires a Java Runtime Environment (JRE) to execute. The debugger will run with a version 5.0 (also called 1.5) or 6.0 (1.6) JRE.

Install the JRE according to the JRE provider's instructions.

Finally you need to export the path to the JRE as follows:

```
export PATH=<path_to_JRE_bin_dir>:$PATH
```

4.2 Starting the Debugger

To start the debugger, first make sure that the compiler environment has been established as described at Establishing the Compiler Environment. Then use the command:

```
idb
```

or

```
idbc
```

as desired.

Once the GUI is started and you see the console window, you're ready to start the debugging session.

Note: Make sure that the executable you want to debug is built with debug info and is an executable file. Change permissions if required, e.g. `chmod +x <application_bin_file>`

4.3 Additional Documentation

Online help titled *Intel® Compilers / Intel® Debugger Online Help* is accessible from the debugger graphical user interface as `Help > Help Contents`.

Context-sensitive help is also available in several debugger dialogs where a `Help` button is displayed.

4.4 Debugger Features

4.4.1 Main Features of IDB

The debugger supports all features of the command line version of the Intel® IDB Debugger. Debugger functions can be called from within the debugger GUI or the GUI-command line. Please refer to the Known Limitations when using the graphical environment.

4.5 Known Issues

4.5.1 Signals Dialog Not Working

The Signals dialog accessible via the GUI dialog Debug / Signal Handling or the shortcut Ctrl+S is not working correctly. Please refer to the Intel® Debugger (IDB) Manual for use of the signals command line commands instead.

4.5.2 Resizing GUI

If the debugger GUI window is reduced in size, some windows may fully disappear. Enlarge the window and the hidden windows will appear again.

4.5.3 `$cdir, $cwd` Directories

`$cdir` is the compilation directory (if recorded). This is supported in that the directory is set; but `$cdir` is not itself supported as a symbol.

`$cwd` is the current working directory. Neither the semantics nor the symbol are supported.

The difference between `$cwd` and '.' is that `$cwd` tracks the current working directory as it changes during a debug session. '.' is immediately expanded to the current directory at the time an entry to the source path is added.

4.5.4 `info stack` Usage

The GDB mode debugger command `info stack` does not currently support negative frame counts the way GDB does, for the following command:

```
info stack [num]
```

A positive value of num prints the innermost num frames, a zero value prints all frames and a negative one prints the innermost –num frames in reverse order.

4.5.5 `$stepg0` Default Value Changed

The debugger variable $stepg0 changed default to a value of 0. With the value "0" the debugger will step over code without debug information if you do a "step" command. Set the debugger variable to 1 to be compatible with previous debugger versions as follows:

```
(idb) set $stepg0 = 1
```

4.5.6 SIGTRAP error on some Linux* Systems

On some Linux distributions (e.g. Red Hat Enterprise Linux Server release 5.1 (Tikanga)) a SIGTRAP error may occur when the debugger stops at a breakpoint and you continue debugging. As a workaround you may define the SIGTRAP signal as follows on command line:

```
(idb) handle SIGTRAP nopass noprint nostop
SIGTRAP is used by the debugger.
SIGTRAP       No        No      No              Trace/breakpoint trap
(idb)
```

Caveat: With this workaround all SIGTRAP signals to the debuggee are blocked.

4.5.7 idb GUI cannot be used to debug MPI processes

The idb GUI cannot be used to debug MPI processes. The command line interface (idbc) can be used for this purpose.

4.5.8 Thread Syncpoint Creation in GUI

While for plain code and data breakpoints the field “Location” is mandatory, thread syncpoints require both “Location” and “Thread Filter” to be specified. The latter specifies the threads to synchronize. Please note that for the other breakpoint types this field restricts the breakpoints created to the threads listed.

4.5.9 Data Breakpoint Dialog

The fields “Within Function” and “Length” are not used. The location to watch provides the watched length implicitly (the type of the effective expression is used). Also “Read” access is not working.

4.5.10 Stack Alignment for IA-32 Architecture

Due to changes in the default stack alignment for the IA-32 architecture, the usage of inferior calls (i.e. evaluation of expressions that cause execution of debuggee code) might fail. This can cause as well crashes of the debuggee and therefore a restart of the debug session. If you need to use this feature, make sure to compile your code with 4 byte stack alignment by proper usage of the `-falign-stack=<mode>` option.

4.5.11 GNOME Environment Issues

With GNOME 2.28, debugger menu icons may not being displayed by default. To get the menu icons back, you need to go to the “System->Preferences->Appearance, Interface” tab and enable, "Show icons in menus". If there is not “Interface” tab available, you can change this with the corresponding `GConf` keys in console as follows:

```
    gconftool-2 --type boolean --set
/desktop/gnome/interface/buttons_have_icons true
    gconftool-2 --type boolean --set
/desktop/gnome/interface/menus_have_icons true
```

4.5.12 Accessing Online-Help

On systems where the Online-Help is not accessible from the IDB Debugger GUI Help menu, you can access the web-based debugger documentation from http://software.intel.com/en-us/articles/intel-software-technical-documentation/

5 Eclipse Integration

The Intel C++ Compiler installs an Eclipse feature and associated plugins (the Intel C++ Eclipse Product Extension) which provide support for the Intel C++ compiler when added as an Eclipse product extension site to an existing instance of the Eclipse* Integrated Development Environment (IDE). With this feature, you will be able to use the Intel C++ compiler from within the Eclipse integrated development environment to develop your applications.

5.1 Supplied Integrations

The Intel feature provided in the directory

```
<install-dir>/eclipse_support/cdt6.0/eclipse
```

supports and requires Eclipse Platform version 3.5, Eclipse C/C++ Development Tools (CDT) version 6.0 or later and a functional Java Runtime Environment (JRE) (version 5.0 (also called 1.5) or 6.0 (1.6).

The Intel feature provided in the directory

```
<install-dir>/eclipse_support/cdt7.0/eclipse
```

supports and requires Eclipse Platform version 3.6, Eclipse C/C++ Development Tools (CDT) version 7.0 or later and a functional Java Runtime Environment (JRE) (version 5.0 (also called 1.5) or 6.0 (1.6).

5.1.1 Integration notes

If you already have the proper versions of Eclipse, CDT and a functional JRE installed and configured in your environment, then you can add the Intel C++ Eclipse Product Extension to your Eclipse Platform, as described in the section, below, entitled How to Install the Intel C++ Eclipse Product Extension in Your Eclipse Platform. Otherwise, you will first need to obtain and install Eclipse, CDT and a JRE, as described in the section, below, entitled How to Obtain and Install Eclipse, CDT and a JRE and then install the Intel C++ Eclipse Product Extension.

5.2 How to Install the Intel C++ Eclipse Product Extension in Your Eclipse Platform

To add the Intel C++ product extension to your existing Eclipse configuration, follow these steps, from within Eclipse.

Open the "Available Software" page by selecting: `Help > Install New Software...` Click on the "Add..." button. Select "Local...". A directory browser will open. Browse to select the `eclipse` directory in your Intel C++ compiler installation. For example, if you installed the compiler as root to the default directory, you would browse to `/opt/intel/composerxe-2011.xxx/eclipse_support/cdt6.0/eclipse`. (This assumes you are using CDT 6.0) Select “OK” to close the directory browser. Then select "OK" to close the “Add Site” dialog. Select the two boxes for the Intel C++ integration: there will be one box for “Intel® C++ Compiler Documentation” and a second box for “Intel® C++ Compiler XE 12.0 for Linux* OS”. Note: The Intel features will not be visible if you have Group items by category set – unset this option to view the Intel features. If you also installed the Intel® Debugger (idb) with its Eclipse product extension and would like to use idb from within Eclipse, repeat the above steps for the idb product extension site.

Click the “Next” button. An “Install” dialog will open which gives you a chance to review and confirm you want to install the checked items. Click “Next”. You will now be asked to accept the license agreement. Accept the license agreement and click “Finish”. Select “OK” on the “Security Warning” dialog that says you are installing software that contains unsigned content. The installation of the Intel support will proceed.

When asked to restart Eclipse, select “Yes”. When Eclipse restarts, you will be able to create and work with CDT projects that use the Intel C++ compiler. See the Intel C++ Compiler documentation for more information. You can find the Intel C++ documentation under `Help > Help Contents > Intel(R) C++ Compiler XE 12.0 User and Reference Guides`.

5.2.1 Integrating the Intel® Debugger into Eclipse

After completing the above steps, including restarting Eclipse, follow these steps to integrate the Intel® Debugger into Eclipse:

- Create a Debug launch configuration by selecting Run > Debug Configurations…
- In the dialog box that pops up, right click on `C/C++ Application` and select New.

- If you are using CDT 7.0, you will now see some tabs on the right. At the bottom-right you should see a label `Using GDB (DSF) Create Process Launcher - Select other…` Click this label – a new dialog will appear. Select `Standard Create Process Launcher` and click OK.
- Go to the Debugger tab and select the Intel® Debugger (idbc) from the combo box. Replace `idbc` with the full path to idbc.

5.3 How to Obtain and Install Eclipse, CDT and a JRE

Eclipse is a Java application and therefore requires a Java Runtime Environment (JRE) to execute. The choice of a JRE is dependent on your operating environment (machine architecture, operating system, etc.) and there are many JRE's available to choose from.

A package containing both Eclipse 3.6 and CDT 7.0 is available from:

http://www.eclipse.org/downloads/

Scroll down to find “Eclipse IDE for C/C++ Developers”. Choose either the Linux 32-bit or Linux 64-bit download as desired.

To download a package containing both Eclipse 3.5 and CDT 6.0, go to

http://wiki.eclipse.org/Older_Versions_Of_Eclipse

and select "Eclipse Galileo SR2 Packages (v 3.5.2). Scroll down to find “Eclipse IDE for C/C++ Developers”. Choose either the Linux 32-bit or Linux 64-bit download as desired.

5.3.1 Installing JRE, Eclipse and CDT

Once you have downloaded the appropriate files for Eclipse, CDT, and a JRE, you can install them as follows:

1. Install your chosen JRE according to the JRE provider's instructions.
2. Create a directory where you would like to install Eclipse and `cd` to this directory. This directory will be referred to as `<eclipse-install-dir>`
3. Copy the Eclipse package binary `.tgz` file to the `<eclipse-install-dir>` directory.
4. Expand the `.tgz` file.
5. Start `eclipse`

You are now ready to add the Intel C++ product extension to your Eclipse configuration as described in the section, *How to Install the Intel C++ Eclipse Product Extension in Your Eclipse Platform*. If you need help with launching Eclipse for the first time, please read the next section.

5.4 Launching Eclipse for Development with the Intel C++ Compiler

If you have not already set your `LANG` environment variable, you will need to do so. For example,

```
setenv LANG en_US
```

Setup Intel C++ compiler related environment variables by executing the `iccvars.csh` (or `.sh`) script prior to starting Eclipse:

`source <install-dir>/bin/iccvars.csh arch_arg` (where "arch_arg" is one of "ia32" or "intel64").

Since Eclipse requires a JRE to execute, you must ensure that an appropriate JRE is available to Eclipse prior to its invocation. You can set the `PATH` environment variable to the full path of the folder of the `java` file from the JRE installed on your system or reference the full path of the java executable from the JRE installed on your system in the `-vm` parameter of the Eclipse command, e.g.:

```
eclipse -vm /JRE folder/bin/java
```

Invoke the Eclipse executable directly from the directory where it has been installed. For example:

```
<eclipse-install-dir>/eclipse/eclipse
```

5.5 Installing on Fedora* Systems

If the Intel C++ Compiler for Linux is installed on an IA-32 or Intel® 64 architecture Fedora* system as a "local" installation, i.e. not installed as root, the installation may fail to properly execute the Eclipse graphical user interfaces to the compiler or debugger. The failure mechanism will typically be displayed as a `JVM Terminated` error. The error condition can also occur if the software is installed from the root account at the system level, but executed by less privileged user accounts.

The cause for this failure is that a more granular level of security has been implemented on Fedora, but this new security capability can adversely affect access to system resources, such as dynamic libraries. This new *SELinux* security capability may require adjustment by your system administrator in order for the compiler installation to work for regular users.

5.6 Selecting Compiler Versions

For Eclipse projects you can select among the installed versions of the Intel C++ Compiler. On IA-32 architecture systems, the supported Intel compiler versions are 9.1, 10.0, 10.1, 11.0, 11.1 and 12.0. On Intel® 64 architecture systems, only compiler versions 11.0, 11.1 and 12.0 are supported.

6 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; and the Intel IPP

Release Notes at http://software.intel.com/en-us/articles/intel-ipp-70-library-release-notes/.

- **Documentation, help, and samples**: see the documentation links on the IPP product page at: http://www.intel.com/software/products/ipp.

6.1 New and Changed Features

- 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.
- 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.
- 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.
- 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 and makefiles.
- Directories formerly designated as "em64t" are now designated by the "intel64" tag. This change may require that you update your build scripts and makefiles .
- 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.
- The domain-specific "emerged" and "merged" static library files have been combined for simpler reference (e.g., ippsemerged.lib + ippsmerged_t.lib ⇒ ipps_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 and makefiles.
- 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.
- 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.
- The SPIRAL generated functions (ippGEN domain) are now being distributed as a separate download. See instructions below for more information.

6.2 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/

6.3 Intel® IPP SPIRAL Domain (ippGEN) is a Separate Download

In order to decrease the size of the IPP library installation package, the SPIRAL domain (ippGEN) is now distributed as a separate library add-on. Go to the Intel® Software Development Products Registration Center to download the ippGEN component of the IPP library.

SPIRAL for IPP is a separate installation package that contains the binaries and header files needed to utilize the functions contained in the ippGEN domain. It is an *add-on* to the IPP library and, therefore, *requires that the core IPP library already be installed on your system*.

You must first install the IPP library product before installing the respective SPIRAL add-on library.

6.4 Intel® IPP Code Samples

The Intel® IPP code samples are organized into downloadable packages for Windows*, Linux* and Mac OS* 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.

7 Intel® Math Kernel Library

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

7.1 Changes in This Version

7.1.1 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 (*GEQRFP) 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) 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
9) The custom dynamic libraries builder now uses the runtime dispatching dynamic libraries on the Linux* and Mac OS* X operating systems
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

7.1.2 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/

7.1.3 Changes in Update 2

1) BLAS: Improved performance of transposition functions on the Intel® Xeon® processor 5600 series
2) BLAS: Added examples for transposition routines
3) FFT: Added Fortran examples showing how to reduce application footprint by linking only functions with the desired precision
4) FFT: Added check for stride consistency on in-place real transforms with CCE storage
5) FFT: Expanded threading to new cases for multi-dimensional transforms
6) VSL: Improved performance of Multivariate Gaussian random number generator for single- and double-precision on 4-core Intel® Xeon® processors 5500 series
7) VML: Improved performance of in-place operation of Add, Mul, and Sub functions on the Intel® Xeon® processor 5500 series

8) Bug fixes: http://software.intel.com/en-us/articles/intel-mkl-103-bug-fixes/

7.2 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.

8 Intel® Threading Building Blocks

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

9 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.