

Intel® C++ Composer XE 2011 for Mac OS* X Installation Guide and Release Notes

Document number: 321413-003US
12 January 2011

Table of Contents

1	Introduction	3
1.1	Change History	3
1.2	Product Contents	3
1.3	System Requirements.....	4
1.4	Documentation.....	4
1.5	Technical Support.....	5
2	Installation.....	6
2.1	Activation of Purchase after Evaluation Using the Intel Activation Tool	6
2.2	Using a License Server	6
2.3	Installation Folders.....	7
2.4	Installing Intel® Integrated Performance Primitives Cryptography Libraries	8
2.5	Relocating Product After Install.....	8
2.6	Removal/Uninstall	8
3	Intel® C++ Compiler	9
3.1	New and Changed Features	9
3.1.1	Three intrinsics changed in update 2	9
3.2	New and Changed Compiler Options	10
3.3	Other Changes	11
3.3.1	Optimization Reports Disabled by Default.....	11
3.3.2	Environment Setup Script Changed	11
3.3.3	OpenMP* Legacy Libraries Removed	11
3.4	Known Issues	11
3.4.1	Runtime crash of 64bit executable running on a 32bit kernel when building with Intel C++ Compiler and XCode 3.2.2.....	11

3.4.2	__GXX_EXPERIMENTAL_CXX0X__ Macro Not Supported.....	12
4	Intel® Debugger (IDB)	12
4.1	Compilation Requirements.....	12
4.2	Known Issues	13
4.2.1	Dwarf vs. Stabs Debug Formats	13
4.2.2	Debug Info from Shared Libraries	13
4.2.3	Non-local Binary and Source File Access	13
4.2.4	Debugging applications that fork.....	13
4.2.5	Debugging applications that exec	13
4.2.6	Snapshots.....	13
4.2.7	Debugging optimized code.....	13
4.2.8	Watchpoints.....	13
4.2.9	Graphical User Interface (GUI)	14
4.2.10	MPP Debugging Restrictions	14
4.2.11	Function Breakpoints	14
4.2.12	Core File Debugging.....	14
4.2.13	Universal Binary Support	14
4.2.14	Debugger variable \$threadlevel	14
4.2.15	Open File Descriptors Limitation	14
4.2.16	\$cdir, \$cwd Directories	15
4.2.17	info stack Usage.....	15
4.2.18	\$stepg0 Default Value Changed.....	15
5	Intel® Integrated Performance Primitives.....	15
5.1	New and Changed Features	15
5.2	Intel® IPP Cryptography Libraries are Available as a Separate Download.....	16
5.3	Intel® IPP SPIRAL Domain (ippGEN) is a Separate Download.....	16
5.4	Intel® IPP Code Samples	17
6	Intel® Math Kernel Library	17
6.1	Changes in This Version.....	17
6.1.1	Changes in Initial Release	17
6.1.2	Changes in Update 1	19
6.1.3	Changes in Update 2	19
6.2	Attributions.....	19

7	Intel® Threading Building Blocks	20
8	Disclaimer and Legal Information	20

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 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.exe`” 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 Mac OS X* includes the following components:

- Intel® C++ Compiler XE 12.0 Update 2 for building applications that run on Intel-based Mac systems running the Mac OS* X operating system
- Intel® Debugger Update 2
- Intel® Integrated Performance Primitives 7.0 Update 2
- Intel® Math Kernel Library 10.3 Update 2
- Intel® Threading Building Blocks 3.0 Update 5
- Integration into the Xcode* development environment
- On-disk documentation

1.3 System Requirements

- An Intel®-based Apple* Mac* system
- 1GB RAM minimum, 2GB RAM recommended
- 3GB free disk space
- One of the following combinations of Mac OS* X, Xcode* and the Xcode SDK:
 - OS X 10.6.5 and Xcode 3.2.4 and SDK 10.6 or 10.5
 - OS X 10.5.8 and Xcode 3.1.4 and SDK 10.6 or 10.5
- gcc* 4

Note: Advanced optimization options or very large programs may require additional resources such as memory or disk space.

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 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

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 will be using Xcode*, please make sure that a supported version of Xcode is installed. If you install a new version of Xcode in the future, you must reinstall the Intel C++ Compiler afterwards.

You will need to have administrative or “sudo” privileges to install, change or uninstall the product.

If you received the compiler product on DVD, insert the DVD. Locate the disk image file (xxx.dmg) on the DVD and double-click on it. If you received the compiler product as a download, double-click the downloaded file.

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 Activation of Purchase after Evaluation Using the Intel Activation Tool

Note for evaluation customers a new tool Intel Activation Tool “ActivationTool” is included in this product release and installed at `/opt/intel/composerxe-2011.x.xxx/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/composerxe-2011.x.xxx/Activation/ActivationTool`. It will convert your evaluation software to a fully licensed product. To use the tool:

```
$ /opt/intel/composerxe-2011.x.xxx/Activation/ActivationTool  
[SN_Num_here]
```

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.

2.3 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. If Xcode integration is installed, a second copy of these files is present under `/Developer/opt/intel`.

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
- `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` – symbolic link to the `man` directory
- `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.

2.4 Installing Intel® Integrated Performance Primitives Cryptography Libraries

The Intel® Integrated Performance Primitives product provides an optional component containing libraries of cryptography routines. Installation and use of the cryptography libraries requires a separate license that is available at no charge from Intel once your license for Intel Integrated Performance Primitives has been registered. Export restrictions apply. For details, please see <http://software.intel.com/en-us/articles/intel-integrated-performance-primitives-cryptography-library/>

2.5 Relocating Product After Install

The Xcode integration is relocatable simply by dragging and dropping the Xcode directory tree to another location. If you wish to use `idb` from a command prompt using a relocated Xcode directory tree, please see <http://software.intel.com/en-us/articles/running-idb-from-command-line-after-relocating-xcode-environment/> for additional steps that are required. Note that `idb` is not available from within the Xcode IDE.

2.6 Removal/Uninstall

It is not possible to remove the compiler while leaving any of the performance library components installed.

1. Open Terminal and set default (`cd`) to any folder outside `<install-dir>`

2. Type the command:
`<install-dir>/compilerpro-12.0.<n>.<pkg>/uninstall_cproc.sh`
3. Follow the prompts

If you are not currently logged in as `root` you will be asked for the `root` password.

3 Intel® C++ Compiler

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

3.1 New and Changed Features

Please refer to the compiler documentation for details

- 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

3.1.1 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 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`
- `-ffriend-injection`
- `-fzero-initialized-in-bss`
- `-fimf-absolute-error`
- `-fimf-accuracy-bits`
- `-fimf-arch-consistency`
- `-fimf-max-error`
- `-fimf-precision`
- `-fms-dialect`
- `-fp-trap`
- `-fp-trap-all`
- `-fvar-tracking`
- `-fvar-tracking-assignments`
- `-opt-args-in-regs`
- `-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.

3.3 Other Changes

3.3.1 Optimization Reports Disabled by Default

As of version 11.1, the compiler no longer issues, by default, optimization report messages regarding vectorization, automatic parallelization and OpenMP threaded loops. If you wish to see these messages you must request them by specifying `-diag-enable vec`, `-diag-enable par` and/or `-diag-enable openmp`, or by using `-vec-report`, `-par-report` and/or `-openmp-report`.

Also, as of version 11.1, optimization report messages are sent to `stderr` and not `stdout`.

3.3.2 Environment Setup Script Changed

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

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.

3.3.3 OpenMP* Legacy Libraries Removed

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

3.4 Known Issues

3.4.1 Runtime crash of 64bit executable running on a 32bit kernel when building with Intel C++ Compiler and XCode 3.2.2

There is a known problem with Intel compiler and the linker from XCode 3.2.2 result in a runtime crash of 64bit executable running on a 32bit kernel.

One specific case is when your code contains switch statement with more than 5 cases. In such cases Intel C++ Compiler will generate symbols starting with “L” and those symbols are not resolved correctly by the linker from XCode 3.2.2.

There is a simple work-around for this issue. Please refer to the article <http://software.intel.com/en-us/articles/intel-compiler-and-xcode-322-linker-runtime-crash-with-switch-statement/> for more recent information.

3.4.2 `__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`.

4 Intel® Debugger (IDB)

4.1 Compilation Requirements

Starting with Xcode 2.3, the Dwarf debugging information is stored in the object (.o) files. These object files are accessed by the debugger to obtain information related to the application being debugged and thus must be available for symbolic debugging.

In cases where a program is compiled and linked in one command, such as:

```
icc -g -o hello.exe hello.c
```

the object files are generated by the compiler but deleted before the command completes. The binary file produced by this command will have no debugging information. To make such an application debuggable users have two options.

Users may build the application in two steps, explicitly producing a .o file:

```
icc -c -g -o hello.o hello.c
```

```
icc -g -o hello.exe hello.o
```

Alternatively, users may use the compiler switch `-save-temps` to prevent the compiler from deleting the .o files it generates:

```
icc -g -save-temps -o hello.exe hello.c
```

The debugger does not use the output of the “dsymutil” utility.

4.2 Known Issues

4.2.1 Dwarf vs. Stabs Debug Formats

The debugger only supports debugging of executables whose debug information is in Dwarf2 format, and does not support the Stabs debug format. Use the `-gdwarf-2` flag on the compile command to have gcc and g++ generate Dwarf output. The Intel compilers (icc and ifort) produce Dwarf2 debug format with the `-g` flag.

4.2.2 Debug Info from Shared Libraries

The debugger does not read debug information from shared libraries. Therefore you cannot set a breakpoint to symbols like `_exit` which are part of a system library.

4.2.3 Non-local Binary and Source File Access

The debugger cannot access binary files from a network-mounted file system (such as NFS). The error message will look like this:

```
Internal error: cannot create absolute path for: /home/me/hello
You cannot debug "/home/me/hello" because its type is "unknown".
```

The debugger cannot access source files from a network-mounted file system (such as NFS). The error message will look like this:

```
Source file not found or not readable, tried...
./hello.c
/auto/mount/site/foo/usr1/user_me/c_code/hello.c
(Cannot find source file hello.c)
```

The file-path specified will be correct.

The workaround in both cases is to copy the files to a local file system (i.e., one which is not mounted over the network).

4.2.4 Debugging applications that fork

Debugging the child process of an application that calls `fork` is not yet supported.

4.2.5 Debugging applications that exec

The `$catchexecs` control variable is not supported.

4.2.6 Snapshots

Snapshots are not yet supported as described in the manual.

4.2.7 Debugging optimized code

Debugging optimized code is not yet fully supported. The debugger may not be able to see some function names, parameters, variables, or the contents of the parameters and variables when code is compiled with optimizations turned on.

4.2.8 Watchpoints

Watchpoints that are created to detect write access don't trigger when a value identical to the original has been written. These restrictions are due to a limitation in the Mac OS* X operating system.

Because the SIGBUS signal rather than the SIGSEGV signal is used by the debugger to implement watchpoints, you cannot create a signal detector which will catch a SIGBUS signal.

4.2.9 Graphical User Interface (GUI)

This version of the debugger does not support the GUI

4.2.10 MPP Debugging Restrictions

MPP debugging is not supported as described in the manual.

4.2.11 Function Breakpoints

Debugger breakpoints set in functions (using the "stop in" command) may not halt user program execution at the first statement. This is due to insufficient information regarding the function prologue in the generated Dwarf debug information. As a work-around, use the "stop at" command to set a breakpoint on the desired statement.

The compiler generates a call to "`__dyld_func_lookup`" as part of the prologue for some functions. If you set a breakpoint on this function the debugger will stop there, but local variable values may not be valid. The work-around is to set a breakpoint on the first statement inside the function.

4.2.12 Core File Debugging

Debugging core files is not yet supported.

4.2.13 Universal Binary Support

Debugging of universal binaries is supported. The debugger supports debugging the IA-32 Dwarf sections of binaries on IA-32 and either the IA-32 or the Intel® 64 sections on Intel® 64.

4.2.14 Debugger variable `$threadlevel`

The manual's discussion of the debugger variable "`$threadlevel`" says "On Mac OS* X, the debugger supports POSIX threads, also known as pthreads." This sentence might be read as implying that other kinds of threads might be supported. This is not true; only POSIX threads are supported on Mac OS* X.

4.2.15 Open File Descriptors Limitation

Because the debugger opens the `.o` files of a debuggee to read debug information, you should raise the open file limit.

Mac OS* limits the number of open file descriptors to 256. You can increase this limit as follows:

```
ulimit -n 2000
```

Please use this command to increase the number of open descriptors before starting the debugger.

This is a workaround until the debugger can better share a limited number of open file descriptors over many files.

4.2.16 `$cdir`, `$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.

4.2.17 `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.

4.2.18 `$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
```

5 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://software.intel.com/en-us/intel-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://software.intel.com/en-us/intel-ipp>.

5.1 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 "_" 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.

5.2 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/>

5.3 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 before installing the respective SPIRAL add-on library.

5.4 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.

6 Intel® Math Kernel Library

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

6.1 Changes in This Version

6.1.1 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
- 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
- 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) 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

6.1.2 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/>

6.1.3 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/>

6.2 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.

7 Intel® Threading Building Blocks

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

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.