Intel® Parallel Studio XE 2015 Composer Edition for C++ Linux* Installation Guide and Release Notes

24 July 2014

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. For the most current update to these release notes, see the release notes posted at the Intel® Software Development Products Registration Center where you downloaded this product.

Due to the nature of this comprehensive integrated software development tools solution, different Intel® Parallel Studio XE components may be covered by different licenses. Please see the licenses included in the distribution as well as the Disclaimer and Legal Information section of these release notes for details.

1.1 Change History

This section highlights important from the previous product version and changes in product updates. For information on what is new in each component, please read the individual component release notes.

1.1.1 Changes since Intel® Composer XE 2013 SP1 (New in Intel® Parallel Studio XE 2015 Composer Edition)

- Compiler offload to Intel® Graphics Technology is supported
- Support for native code generation for Intel® Graphics Technology
- -ansi-alias is now enabled by default (may change runtime behavior without warning)
- New Optimization Report interface, structure, and options (strongly recommended to read for users of existing options -opt-report, -vec-report, -openmp-report, and -par-report)
- Full C++11 language support (see details for potential limitations)
- Additional OpenMP* 4.0 features
- Intel® Cilk™ Plus changes
- Create custom install packages with the online installer
- Enforce same code to be executed regardless of data alignment with –no-opt-dynamic-align
- Enable threadsafe profile generation with PGO

- Control diagnostic strictness of Pointer Checker for problems with pointers to structure fields
- aligned_ new header
- GNU-compatible function multiversioning for CPU dispatching
- Improved debugging of lambda functions
- Debug information now in DWARF Version 3 format by default
- Extended offload syntax to allow copying of non-contiguous memory
- New pragma directives to control inlining behavior per function
- New INTEL_PROF_DYN_PREFIX environment variable to add custom prefix to PGO .dyn filenames
- _bittest and _bittestandcomplement intrinsics supported in 15.0 compiler
- gcc 4.9 supported
- binutils 2.19 not supported, binutils 2.24 supported
- Red Hat Enterprise Linux* 7 now supported
- Ubuntu* 14.04 LTS now supported
- SUSE LINUX Enterprise Server* 10 not supported
- Debian 6* is deprecated
- Python* no longer a requirement to use GNU* Project Debugger
- Intel® Debugger (IDB) removed
- Static analysis is deprecated
- Compiler options starting with –o are deprecated
- Intel® C++ Compiler 15.0.0
- Intel® Math Kernel Library 11.2
- Intel® Integrated Performance Primitives 8.2
- Intel® Threading Building Blocks 4.3
- GNU* Project Debugger (GDB*) 7.7 with improved Intel® Cilk™ Plus support

1.2 Product Contents

Intel® Parallel Studio XE 2015 Composer Edition for C++ Linux initial release* includes the following components:

- Intel® C++ Compiler 15.0.0 for building applications that run on IA-32, Intel® 64 architecture, Intel® Xeon Phi™ coprocessors, or Intel® Graphics Technology running the Linux* operating system
- GNU* Project Debugger (GDB*) 7.7
- Intel® Math Kernel Library 11.2
- Intel® Integrated Performance Primitives 8.2
- Intel® Threading Building Blocks 4.3
- Integration into the Eclipse* development environment
- On-disk documentation

1.3 Intel® Debugger (IDB) is removed from this release

The Intel Debugger (IDB) has been removed from this release. A debugger based on the GNU* Project Debugger (GDB*) is now provided for debugging.

1.4 System Requirements

For an explanation of architecture names, see http://intel.ly/q9JVjE

- 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 of 64-bit applications or applications targeting Intel® MIC Architecture is supported on a 64-bit version of the OS only. Development of 32-bit applications is supported on either 32-bit or 64-bit versions of the OS Development for a 32-bit on a 64-bit host may require optional library components (ia32-libs, lib32gcc1, lib32stdc++6, libc6-dev-i386, gcc-multilib, g++-multilib) to be installed from your Linux distribution.
- For Intel® MIC Architecture development/testing:
    - Intel® Xeon Phi™ coprocessor
    - Intel® Manycore Platform Software Stack (Intel® MPSS)
- For offload to or native support for Intel® Graphics Technology development/testing
    - Offload is supported from 64-bit applications only
    - SUSE LINUX Enterprise Server* 11 SP3
        - Kernel 3.0.76-11
    - Ubuntu* 12.04 LTS
        - Kernel 3.2.0-41 for 3rd Generation Intel® Core™ Processors
        - Kernel 3.8.0-23 for 4th Generation Intel® Core™ Processors
    - The following processor models are supported:
        - Intel® Xeon® Processor E3-1285 v3 and E3-1285L v3 (Intel® C226 Chipset) with Intel® HD Graphics P4700
        - 4th Generation Intel® Core™ Processors with Intel® Iris™ Pro Graphics, Intel® Iris™ Graphics or Intel® HD Graphics 4200+ Series
        - 3rd Generation Intel® Core™ Processors with Intel® HD Graphics 4000/2500

          Please note: Intel® Xeon® processors are only supported with the chipsets listed. Intel® Xeon® configurations with other chipsets are not supported. Previous generations of Intel® Core™ processors are not supported. Intel® Celeron and Intel® Atom™ processors are also not compatible.
    - The latest 64-bit graphics driver with support for Intel® Graphics Technology (available from the Intel® Software Development Products Registration Center at http://registrationcenter.intel.com). You should have access to the Intel® HD Graphics Drivers for Linux* download area as part of your Intel® Parallel Studio XE registration. If you do not see this area, please contact support.

- For the best experience, a multi-core or multi-processor system is recommended
- 2GB of RAM (4GB recommended)
- 7.5GB 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):
    - Fedora* 20
    - Red Hat Enterprise Linux* 5, 6, 7
    - SUSE LINUX Enterprise Server* 11
    - Ubuntu* 12.04 LTS (64-bit only), 13.10, 14.04 LTS
    - Debian* 6.0, 7.0
    - Intel® Cluster Ready
- Linux Developer tools component installed, including gcc, g++ and related tools
    - gcc versions 4.1-4.9 supported
    - binutils versions 2.17-2.24 supported
- 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 integration into the Eclipse development environment*

- Eclipse Platform version 4.3 with:
    - Eclipse C/C++ Development Tools (CDT) 8.2 or later
    - Java* Runtime Environment (JRE) 6.0 (also called 1.6†) or later
- Eclipse Platform version 4.2 with:
    - Eclipse C/C++ Development Tools (CDT) 8.1 or later
    - Java* Runtime Environment (JRE) 6.0 (also called 1.6†) or later
- Eclipse Platform version 3.8 with:
    - Eclipse C/C++ Development Tools (CDT) 8.1 or later
    - Java* Runtime Environment (JRE) 6.0 (also called 1.6†) or later

† There is a known issue with JRE 6.0 through update 10 that causes a crash on Intel® 64 architecture. It is recommended to use the latest update for your JRE. See http://www.eclipse.org/eclipse/development/readme_eclipse_3.7.html section 3.1.3 for details.

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.

1.4.1 SuSE Enterprise Linux 10* is Not Supported

Support has been removed for installation and use on these operating system versions. Intel recommends migrating to a newer version of these operating systems.

1.4.2 Red Hat Enterprise Linux 5* and Debian 6* are Deprecated

Support for Red Hat Enterprise Linux 5* and Debian 6* is deprecated and will be removed in a future release.

1.5 Documentation

Product documentation can be found in the `Documentation` folder as shown under Installation Folders.

1.6 Samples

Samples for each product component can be found in the `Samples` folder as shown under Installation Folders.

1.7 Japanese Language Support

Intel compilers provide support for Japanese language users (when the combined Japenese-English installation is used). 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.

Japanese language support will be available in an update on or after the release of Intel® Parallel Studio XE Composer Edition 2015.

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://intel.ly/qhINDv

1.8 Technical Support

If you did not register your compiler during installation, please do so at the Intel® Software Development Products Registration Center at http://registrationcenter.intel.com. 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.

To begin installation, first unpack the installation tarball 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.

Please do not run the install script as a background process (i.e. running "./install.sh &"). This is not supported.

2.1 GUI installation now available

If on a Linux* system with GUI support, the installation will now provide a GUI-based installation. If a GUI is not supported (for example if running from an ssh terminal), a command-line installation will be provided.

2.2 Online Installation now available

The electronic installation package for Intel® Parallel Studio XE now offers as an alternative a smaller installation package that dynamically downloads and then installs packages selected to be installed. This requires a working internet connection and potentially a proxy setting if you are behind an internet proxy. Full packages are provided alongside where you download this online install package if a working internet connection is not available. The online installer may be downloaded and saved as an executable file which can then be launched from the command line.

2.2.1 `http_proxy` is set, but sudo installation still fails to connect

Most sudo profiles are set to not inherit certain settings like `http_proxy` from the original user. Make sure your /etc/sudoers file contains a line like the following to allow your proxy settings to propagate:

```
Defaults   env_keep += "http_proxy"
```

2.2.2 Storing Online Installer Download Content

The online installer stores the downloaded content in the form-factor of the standard install package which can then be copied and reused offline on other systems. The default download location is /tmp/<UID>. This location may be changed with the online installer command line option "--download-dir [FOLDER]". The online installer also supports a download only mode which allows the user to create a package without installation. This mode is enabled with the "--download-only" command line option.

2.3 Intel® Software Manager

The installation now provides an Intel® Software Manager to provide a simplified delivery mechanism for product updates and provide current license status and news on all installed Intel® software products.

You can also volunteer to provide Intel anonymous usage information about these products to help guide future product design. This option, the Intel® Software Improvement Program, is not

enabled by default – you can opt-in during installation or at a later time, and may opt-out at any time. For more information please see http://intel.ly/SoftwareImprovementProgram.

2.4 Installation of Intel® Manycore Platform Software Stack (Intel® MPSS)

The Intel® Manycore Platform Software Stack (Intel® MPSS) may be installed before or after installing the Intel® Parallel Studio XE for Linux* product.

Using the latest version of Intel® MPSS available is recommended. It is available from the Intel® Software Development Products Registration Center at http://registrationcenter.intel.com as part of your Intel® Parallel Studio XE for Linux* registration.

Refer to the Intel® MPSS documentation for the necessary steps to install the user space and kernel drivers.

2.5 Cluster Installation

To install a product on multiple nodes of a cluster on Linux*, the following steps should be taken:

1) Run the installation on a system where Intel® Parallel Studio XE Cluster Edition is installed. Also, passwordless ssh must be configured between machines in a cluster.
2) On step "`4 Options`" there will be a "`Cluster installation`" option. The default value is "`Current node`".
3) To install on a cluster, the user must select this option and then provide a `machines.LINUX` file with IP-addresses, hostnames, FQDNs, and other information for the cluster nodes (one node per line). The first line is expected to describe the current (master) node.
4) Once the `machines.LINUX` file is provided, additional options will appear: `Number of parallel installations`, `Check for shared installation directory`.
5) When all options are configured and installation has begun, the installation will check connectivity with all nodes (a prerequisite) and only then will it install the product on these nodes.

2.6 Silent Install

For information on automated or “silent” install capability, please see http://intel.ly/ngVHY8.

2.6.1 Support of Non-Interactive Custom Installation

Intel® Parallel Studio XE 2015 Composer Edition supports the saving of user install choices during an ‘interactive’ install in a configuration file that can then be used for silent installs. This configuration file is created when the following option is used from the command line install:

- `--duplicate config_file_name (or -d config_file_name)`: it specifies the configuration file name. If full path file name is specified, the “`--download-dir`” is ignored and the installable package will be created under the directory where configuration file is.
- `--download-dir dir_name`: optional, it specifies where the configuration file will be created. If this option is omitted, the installation package and the configuration file will be created under the default download directory:
  - `/tmp/<UID>/<package_id>`

For example: `l_ccompxe_online_2015.0.0XX.sh –duplicate icc15_install_config.ini --download-dir "/temp/custom_pkg_ic15"`
The configuration file and installable package will be created under "`"/temp/custom_pkg_ic15`".

2.7 Using a License Server

If you have purchased a "floating" license, see http://intel.ly/pjGfwC 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.

2.8 Eclipse* Integration Installation

Please refer to the section below on Eclipse Integration

2.9 Known Installation Issues

- When uninstalling a more recent major version of Intel® Parallel Studio XE Composer Edition or Intel® Composer XE, the symbolic links will be removed even if earlier versions of Intel Parallel Studio XE Composer Edition or Intel® Composer XE exist on the system. For example, uninstalling Intel® Parallel Studio XE 2015 Composer Edition will remove the links if this is the only 2015 update installed on the system, regardless if Intel® Composer XE 2013 SP1 is also installed. As a workaround, refer to the older product's directories explicitly (for example, `composer_xe_2013_sp1.<n>.<pkg>`)
- 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 system running Ubuntu or Debian, please visit the Intel® Software Evaluation Center (http://intel.ly/nJS8y8) to obtain an evaluation serial number.

2.10 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.

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
- `ism` – contains files for Intel® Software Manager
- `composerxe` – symbolic link to the `composer_xe_2015` directory
- `composer_xe_2015` – directory containing symbolic links to subdirectories for the latest installed Intel® Parallel Studio XE 2015 compiler release
- `composer_xe_2015.<n>.<pkg>` - physical directory containing files for a specific compiler version. `<n>` is the update number, and `<pkg>` is a package build identifier.

Each `composer_xe_2015` directory contains the following directories that reference the latest installed Intel® Parallel Studio XE 2015 Composer Edition:

- `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 `composer_xe_2015.<n>.<pkg>` directory contains the following directories that reference a specific update of the Intel® Parallel Studio XE 2015 Composer Edition:

- `bin` – all executables
- `pkg_bin` – symbolic link to `bin` directory
- `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

- `uninstall` – Files for uninstallation

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® Parallel Studio XE 2015 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.

2.11 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>/composer_xe_2015.<n>.<pkg>/uninstall.sh` for a command-line uninstall or `<install-dir>/composer_xe_2015.<n>.<pkg>/uninstall-GUI.sh` for a GUI uninstall.
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 15.0 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

This product now contains Intel® C++ Compiler 15.0. The following features are new or significantly enhanced in this version. For more information on these features, please refer to the documentation.

- Support for native code generation for Intel® Graphics Technology
- Support for offload to Intel® Graphics Technology
- -ansi-alias is now enabled by default (may change runtime behavior without warning)
- New Optimization Report interface, structure, and options
- Full C++11 language support (includes these feature new to 15.0) (-std=c++11)
    - Value categories (N3055)
    - alignas and alignof (N2341)
    - decltype extensions (N3049, N3276)
    - Inheriting constructors (N2540)
    - User-defined literals (N2765)
    - thread_local (N2659)
    - Note that language features available can depend on gcc* version installed. The version of gcc used for Intel® Xeon Phi™ coprocessor development may vary from the version used for host development which could affect C++11 language feature support.
- Additional OpenMP* 4.0 features
- Intel® Cilk™ Plus changes in Intel® C++ Compiler 15.0
- GNU-compatible function multiversioning for CPU dispatching
- aligned_new header
- Improved debugging of lambda functions
- Debug information now in DWARF Version 3 format by default
- Extended offload syntax to allow copying of non-contiguous memory
- GNU C standard include files provided with Intel compiler
- New pragma directives to control inlining behavior per function
- New INTEL_PROF_DYN_PREFIX environment variable to add custom prefix to PGO .dyn filenames
- Static Analysis is deprecated
- _bittest and _bittestandcomplement intrinsics supported in Intel® C++ Compiler 15.0

3.2.1 Support for native code generation for Intel® Graphics Technology

By default, the compiler will generate virtual ISA code for the kernels to be offloaded to Intel® Graphics Technology. The vISA is target independent and will run on processors that have the Intel graphics processor integrated on the platform and that have the proper Intel® HD Graphics driver installed. The Intel HD Graphics driver contains the offload runtime support and a Jitter (just-in-time compiler) that will translate the virtual ISA to the native ISA at runtime for the platform on which the application runs and do the offload to the processor graphics. The Jitter gets the current processor graphics information at runtime. The new feature allows generation

of native ISA at link time by using the new option /Qgpu-arch:<arch> for Windows and –mgpu-arch=<arch> for Linux. The option is described in detail in the User's Guide.

3.2.2 Static Analysis is deprecated

Support for Static Analysis is deprecated and will be removed in a future release. If you have concerns or feedback, please comment.

3.2.3 Support for offload to Intel® Graphics Technology

Support is provided via either a synchronous (with `#pragma offload target(gfx)` and a `cilk_for` parallel loop) or asynchronous (with `#pragma offload target(gfx_kernel)` and APIs provided in the provided `gfx_rt.h` header) offload implementation. Offload is supported from 64-bit applications only. Please see the Intel Compiler User's Guide under `Key Features->Intel® Graphics Technology` for information. Known limitations are documented in the release notes.

3.2.4 _bittest and _bittestandcomplement intrinsics supported in Intel® C++ Compiler 15.0

Support is now provided for the _bittest and _bittestandcomplement intrinsics that are currently supported for Windows*.

3.2.5 Intel-specific version of x86intrin.h provided in Intel® C++ Compiler 15.0 to avoid compilation errors

Compilations including the x86intrin.h provided by gcc* 4.9 may include unguarded references to intrinsics the Intel compiler does not support. To resolve this, the Intel® C++ Compiler is providing a version of x86intrin.h that will compile cleanly.

3.2.6 New Optimization Report interface, report structure, and options in Intel® C++ Compiler 15.0

The four kinds of optimization reports (-opt-report, -vec-report, -openmp-report, and -par-report) have been consolidated under one -opt-report interface in this version of Intel® C++ Compiler. This consolidated optimization report has been rewritten to improve the presentation, content, and precision of the information provided so that users better understand what optimizations were performed by the compiler and how they may be tuned to yield the best performance.

The output of this report no longer defaults to stderr due to issues with parallel builds. Instead, by default an output file (extension .optrpt) containing the report for each corresponding object file is generated to the target directory of the compilation process (i.e. the same directory where object files would be generated). -opt-report-file (for example: -opt-report-file:stderr) can be used to change this behavior.

The -vec-report, -openmp-report, and -par-report options have been deprecated, but they remain and map to corresponding values of the -opt-report option. However, the report information and formatting, and the default to reporting to a file, will follow the new opt-report model.

It is strongly recommended that you read the documentation for full details. See the Intel Compiler User's Guide under `Compiler Reference->Compiler Option Categories and Descriptions->Optimization Report Options`.

3.2.7 Updated Support for Upcoming OpenMP* features added in the Intel® C++ Compiler 15.0

Intel Compiler 15.0 adds the following OpenMP* 4.0 features:

- `cancel` and `cancellation point` directives
- `depend` clause for `task` directives

OpenMP* 4.0 user defined reductions are not supported.

3.2.8 Intel® Cilk™ Plus changes in Intel® C++ Compiler 15.0

Please note the following new features for Intel® Cilk™ Plus in Intel C++ Compiler 15.0:

- Ability to implement explicit vector programming with keywords as an alternative to #pragma simd syntax. The keywords are _Simd, _Safelen, and _Reduction. See the compiler User's Guide for more details.
- __intel_simd_lane() intrinsic to represent the "lane id" within a SIMD vector function (__declspec(vector))
- New __attribute__((vector_variant(…))) to define a vector-specific overload of a scalar function.
- Intel Cilk Plus documentation can now be generated using Doxygen*. See the compiler/include/cilk/ReadMe.html for more details.

3.2.9 GNU-compatible function multiversioning for CPU dispatching

A new interface for CPU dispatching has been added for improved compatibility with gcc. See http://gcc.gnu.org/onlinedocs/gcc/Function-Multiversioning.html for more information.

3.2.10 aligned_new header provides way to correctly dynamically allocate data with class types with alignment specifications

C++11 allows the programmer to specify increased data alignment for class types, but there is no standard mechanism for actually allocating data with that alignment based on those types. aligned_new provides overloads of new and delete that will properly align such data.

3.2.11 GNU C standard include files provided with Intel compiler

The files limits.h, stddef.h, stdbool.h, stdarg.h, stdint.h, and iso646.h are now provided as part of our compiler package, and compilations using icc/icpc will include these provided headers. To disable use of these includes, use the compiler option –no-gcc-include-dir.

3.2.12 New pragma directives to control inlining behavior per function

Intel® C++ Compiler 15.0 has added two new pragma directives, inline-max-per-routine and inline-max-total-size, to control the inlining behavior per function. See the C++ Compiler User's Guide under Compiler Reference->Pragmas->Intel-specific Pragma Reference for more information.

3.2.13 Static Analysis Feature (formerly "Static Security Analysis" or "Source Checker") Requires Intel® Inspector XE

The "Source Checker" feature, from compiler version 11.1, has been enhanced and renamed "Static Analysis". The compiler options to enable Static 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.

3.3 New and Changed Compiler Options

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

Please see Compiler Options section of the documentation for detailed information.

3.3.1 New and Changed in Intel C++ Compiler 15.0

- -mgpu-arch=<arch>
- -q[no-]opt-multi-version-aggressive
- -qopt-ra-region-strategy[=keyword]
- -qopt-malloc-options={0|1|2|3|4}
- -qopt-calloc
- -q[no-]opt-jump-tables=<arg>
- -qopt-block-factor=<n>
- -qopt-streaming-stores=<keyword>
- -qopt-subscript-in-range
- -q[no-]opt-matmul
- -q[no-]opt-mem-layout-trans[=<level>]
- -q[no-]opt-prefecth[=n]
- -qopt-prefetch-distance=n1[,n2]
- -qopt-threads-per-core=n
- -qopt-streaming-cache-evict=n
- -qopt-gather-scatter-unroll=n
- -qno-opt-gather-scatter-unroll
- -q[no-]opt-dynamic-align
- -qopenmp
- -qopenmp-stubs
- -qopenmp-lib=<ver>
- -qopenmp-link=<library>
- -qopenmp-task=<arg>
- -qopenmp-threadprivate=<ver>
- -q[no-]openmp-simd
- -q[no-]openmp-offload
- -qopt-assume-safe-padding
- -q[no-]offload=<arg>

- -qoffload-attribute-target=<name>
- -qoffload-option,<target>,<tool>,"option list"
  - "jit" added as tool for offload to Intel® Graphics Technology
- -f[no-]fat-lto-objects
- -prof-gen=threadsafe
- -qopt-report[=n]
- -qopt-report-file=[stdout | stderr | <file>]
- -qopt-report-per-object
- -qopt-report-phase=<phase>[,<phase>,…]
- -qopt-report-routine=<name>[,<name>,…]
- -qopt-report-filter=<string>
- -qopt-report-format=[text|vs]
- -qopt-report-names=[mangled|unmangled]
- -qopt-report-help
- -q[no-]opt-report-embed
- -[no-]check-pointers-narrowing
- -no-gcc-include-dir
- -std=gnu++11
- -debug [no]emit-column
- -debug [no]macros
- -fast and –Ofast now include –fp-model fast=2
- -f[no-]eliminate-unused-debug-types
- -I-

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

3.3.2 Compiler options starting with –o are deprecated

All compiler options starting with –o are deprecated. These will be replaced by new options preceded with –q. For example, -opt-report should now be –qopt-report. This is to improve compatibility with third-party tools that expect –o<text> to always refer to output filenames.

3.3.3 –ansi-alias is enabled by default

To align more closely with gcc, -ansi-alias is enabled by default at –O2 and above. This option allows the compiler to assume the code follows ANSI aliasing rules when generating code. If your code does not follow these rules, incorrect code may be generated. To disable this, use –no-ansi-alias.

3.3.4 Use -I- to control if search path is used for include files with angle brackets

Use of –I- effectively splits the command line include paths specified. Any directories specified with -I options before -I- are searched only for headers included with the "`#include "file"`" directive. They are not searched for headers included with angle brackets as in "`#include <file>`". If additional directories are specified with -I options after the -I- in the command line, those directories are searched for all #include directives. In addition, -I- inhibits the use of the

current file directory as the first search directory for includes with quotes as in "`#include "file"`".

3.3.5 Enforce same code to be executed regardless of data alignment with –no-opt-dynamic-align

By default, the compiler may generate multiple code paths to execute depending on the alignment of data in order to improve performance which may affect the consistency of floating-point calculations. To disable this behavior, use –no-opt-dynamic-align

3.3.6 Enable threadsafe profile generation with PGO

To enable the safe generation of profile information in multithreaded applications, use the -prof-gen=threadsafe option.

3.3.7 Control diagnostic strictness of Pointer Checker for problems with pointers to structure fields

The –no-check-pointers-narrowing option can be used to disable Pointer Checker diagnostics for problems with pointers to structure fields.

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. If you wish to build for Intel® Many Integrated Core Architecture, use `intel64`. Establishing the compiler environment also establishes the environment for the provided GNU* GDB (`gdb-ia and gdb-mic`), 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 "asm" keyword no longer accepted with –std=c99 in Intel® C++ Compiler 15.0 for Linux*

When using `-std=c99`, the `asm` keyword will not be recognized starting with the Intel C++ Compiler 15.0. This matches current gcc behavior. Use `-std=gnu99` to recognize the asm keyword while maintaining C99 compilation.

3.5 Known Issues

3.5.1.1 Pointer Checker requires a dynamic runtime library

When using the `-check-pointers` option, the runtime library libchkp.so must be linked in. When using options like `-static` or `-static-intel` with `-check-pointers`, be aware that this dynamic library will be linked in regardless of your settings. See the article at http://intel.ly/1jV0eWD for more information.

3.5.2 Known Issues with Intel® Many Integrated Core Architecture (Intel® MIC Architecture)

3.5.2.1 Using offload code in shared libraries requires main program to be linked with –offload=mandatory or –offload=optional option

There is initialization required for offload that can only be done in the main program. For offload code in shared libraries, this means that the main program must also be linked for offload so that the initialization happens. This will happen automatically if the main code or code statically linked with the main program contains offload constructs. If that is not the case, you will need to link the main program with the `-offload=mandatory` or `-offload=optional` compiler options.

3.5.2.2 Missing symbols not detected at link time

In the offload compilation model, the binaries targeting the Intel® MIC Architecture are generated as dynamic libraries (.so). Dynamic libraries do not need all referenced variables or routines to be resolved during linking as these can be resolved during load time. This behavior could mask some missing variable or routine in the application resulting in a failure during load time. In order to identify and resolve all missing symbols at link time, use the following command line option to list the unresolved variables.

```
-offload-option,mic,compiler,"-z defs"
```

*3.5.2.3 *MIC* tag added to compile-time diagnostics*

The compiler diagnostics infrastructure is modified to add an additional offload *MIC* tag to the output message to allow differentiation from the Target (Intel® MIC Architecture) and the host CPU compilations. The additional tag appears only in the Target compilation diagnostics issued when compiling with offload extensions for Intel® MIC Architecture.

In the examples below the sample source programs trigger identical diagnostics during both the host CPU and Target Intel® MIC Architecture compilations; however, some programs will

generate different diagnostics during these two compilations. The new tag permits easier association with either the CPU or Target compilation.

```
$ icc -c sample.c
sample.c(1): warning #1079: *MIC* return type of function "main" must
be "int"
  void main()
       ^

sample.c(5): warning #120: *MIC* return value type does not match the
function type
      return 0;
             ^

sample.c(1): warning #1079: return type of function "main" must be
"int"
  void main()
       ^

sample.c(5): warning #120: return value type does not match the
function type
      return 0;
```

3.5.2.4 Runtime Type Information (RTTI) not supported

Runtime Type Information (RTTI) is not supported under the Virtual-Shared memory programming method; specifically, use of `dynamic_cast<>` and `typeid()` is not supported.

3.5.2.5 Direct (native) mode requires transferring runtime libraries like libiomp5.so to coprocessor

The Intel® Manycore Platform Software Stack (Intel® MPSS) no longer includes Intel compiler libraries under /lib, for example the OpenMP* library, libiomp5.so.

When running OpenMP* applications in direct mode (i.e. on the coprocessor card), users must first upload (via scp) a copy of the Intel® MIC Architecture OpenMP* library (`<install_dir>/compiler/lib/mic/libiomp5.so`) to the card (device names will be of the format `micN`, where the first card will be named `mic0`, the second `mic1`, and so on) before running their application.

Failure to make this library available will result in a run-time failure like:

```
/libexec/ld-elf.so.1: Shared object "libiomp5.so" not found, required
by "sample"
```

This can also apply to other compiler runtimes like libimf.so. The required libraries will depend on the application and how it's built.

3.5.2.6 Calling exit() from an offload region

When calling exit() from within an offload region, the application terminates with an error diagnostic "`offload error: process on the device 0 unexpectedly exited with code 0`"

3.5.3 Known issues for offload to Intel® Graphics Technology

3.5.3.1 Host-side execution of offload code is not parallelized

The compiler will generate both a target and host version of the parallel loop under `#pragma offload`. The host version is executed when the offload cannot be performed (usually when the target system does not have a unit with Intel® Graphics Technology enabled).The parallel loop must be specified using the parallel syntax of `cilk_for` or an Array Notation statement, which has parallel semantics for offload. The target version of the loop will be parallelized for target execution, but there is a current limitation where the host-side back-up version of the parallel loop will not be parallelized. Please be aware this can affect the performance of the back-up code execution significantly when offload execution does not happen in the case of `cilk_for` use. Array notation does not currently generate parallel code on the host, so performance should not differ here in that case. This is a known issue that may be resolved in a future product release.

3.5.3.2 If multiple processes running with non-root privilege try to offload there may be sporadic fails.

You may see sporadic fails if multiple processes (with non-root privilege) try to offload. Only the first process that opens `/dev/dri/card0` can pass DRM authentication. Only the first process to open `/dev/dri/card0` has master privilege. "Root" or "master" privilege is needed to pass the DRM authentication. That is why all processes pass through when running with root privilege, but only one of them passes with non-root privilege. This is a known restriction for Linux*.

Possible workarounds

- Execute each process serially.
- Execute as root

3.5.3.3 Other known limitations with offload to Intel® Graphics Technology

- In the offloaded code, the following are not allowed:
  - Exception handling
  - RTTI
  - longjmp/setjmp
  - VLA
  - Variable parameter lists
  - Virtual functions, function pointers, or other indirect calls or jumps
  - Shared virtual memory
  - Data structures containing pointers, such as arrays or structs
  - Globals with pointer or reference type

    - OpenMP*
    - cilk_spawn or cilk_sync
    - Intel® Cilk™ Plus reducers
    - ANSI C runtime library calls (with the exception of SVML, math.h, and mathimf.h calls and a few others)
- 64-bit float and integer operations are inefficient

3.5.4 Intel® Cilk™ Plus Known Issues

- Static linkage of the runtime is not supported

  Static versions of the Intel® Cilk™ Plus library are not provided by design. Using `-static-intel` to link static libraries will generate an expected warning that the dynamic version of the of Intel® Cilk™ Plus library, libcilkrts.so, is linked.

```
$ icc -static-intel sample.c

icc: warning #10237: -lcilkrts linked in dynamically, static
library not available
```

  Alternatively, you can build the open source version of Intel Cilk Plus with a static runtime. See http://cilk.com for information on this implementation of Intel Cilk Plus.

3.5.5 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

3.5.6 Static Analysis Known Issues

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

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

Static 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 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 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 analysis result that was affected by this issue and you are able to open that result in the Intel® 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 GNU* GDB Debugger

This section summarizes the changes, new features, customizations and known issues related to the GNU* GDB provided with Intel® Parallel Studio XE 2015.

4.1 Features

GNU* GDB provided with Intel® Parallel Studio XE 2015 is based on GDB 7.7 with enhancements provided by Intel. This debugger replaces the Intel® Debugger from previous releases. In addition to features found in GDB 7.7, there are several other new features:

- Support for Intel® Many Integrated Core Architecture (Intel® MIC Architecture)
- Support for Intel® Transactional Synchronization Extensions (Intel® TSX)
- Register support for Intel® Memory Protection Extensions (Intel® MPX) and Intel® Advanced Vector Extensions 512 (Intel® AVX-512)
- Data Race Detection (*pdbx*):
  Detect and locate data races for applications threaded using POSIX* thread (pthread) or OpenMP* models
- Branch Trace Store (*btrace*):
  Record branches taken in the execution flow to backtrack easily after events like crashes, signals, exceptions, etc.
- Pointer Checker:
  Assist in finding pointer issues if compiled with Intel® C++ Compiler and having Pointer Checker feature enabled (see Intel® C++ Compiler documentation for more information)
- Improved Intel® Cilk™ Plus Support
  Serialized execution of Intel® Cilk™ Plus parallel applications can be turned on and off during a debug session using the following command:
  `(gdb) set cilk-serialization [on|off]`

4.2 Using GNU* GDB

GNU* GDB provided with Intel® Parallel Studio XE 2015 comes in different versions:

- IA-32/Intel® 64 debugger:
  Debug applications natively on IA-32 or Intel® 64 systems with `gdb-ia` on the command line.
  A standard Eclipse* IDE can be used for this as well if a graphical user interface is desired.
- Intel® Xeon Phi™ coprocessor debugger:
  Debug applications remotely on Intel® Xeon Phi™ coprocessor systems. The debugger will run on a host system and a debug agent (`gdbserver`) on the coprocessor.
  There are two options:
  - Use the command line version of the debugger with `gdb-mic`. This only works for native Intel® Xeon Phi coprocessor applications.
    A standard Eclipse* IDE can be used for this as well if a graphical user interface is desired.
  - Use an Eclipse* IDE plugin shipped with Intel® Parallel Studio XE 2015. This works only for offload enabled Intel® Xeon Phi coprocessor applications.

Instructions on how to use GNU* GDB can be found in the Documentation section.

4.3 Documentation

The documentation for the provided GNU* GDB can be found here:

```
<install-dir>/Documentation/[en_US|ja_JP]/debugger/gdb/gdb.pdf
<install-dir>/Documentation/[en_US|ja_JP]/debugger/ ↵
                                        gdb/gdb_quickstart_lin.pdf
```

4.4 Known Issues and Changes

4.4.1 Problem using Eclipse* IDE with GDB*

If the GNU* GDB version that is provided by Intel is used within an Eclipse* IDE, you need to setup the environment before starting Eclipse* as described in the section on Establishing the Compiler Environment.

4.4.2 Safely ending offload debug sessions

To avoid issues like orphan processes or stale debugger windows when ending offload applications, manually end the debugging session before the application is reaching its exit code. The following procedure is recommended for terminating a debug session.

- Manually stop a debug session before the application reaches the exit-code.
- When stopped, press the red stop button in the toolbar in the Intel® MIC Architecture-side debugger first. This will end the offloaded part of the application.
- Next, do the same in the CPU-side debugger.

- The link between the two debuggers will be kept alive. The Intel® MIC Architecture-side debugger will stay connected to the debug agent and the application will remain loaded in the CPU-side debugger, including all breakpoints that have been set.
- At this point, both debugger windows can safely be closed.

4.4.3 Intel® MIC Architecture-side debugger asserts on setting source directories

Setting source directories in the GNU* GDB might lead to an assertion.

Resolution:
The assertion should not affect debugger operation. To avoid the assertion anyway, don't use source directory settings. The debugger will prompt you to browse for files it cannot locate automatically.

4.4.4 Accessing _Cilk_shared variables in the debugger

Writing to a shared variable in an offloaded section from within the CPU-side debugger **before** the CPU-side debuggee has accessed that variable may result in loss of the written value/might display a wrong value or cause the application to crash.

Consider the following code snippet:

```
_Cilk_shared bool is_active;
_Cilk_shared my_target_func() {
//Accessing "is_active" from the debugger *could* lead to unexpected
//results e.g. a lost write or outdated data is read.
is_active = true;
//Accessing "is_active" (read or write) from the debugger at this
//point is considered safe e.g. correct value is displayed.
}
```

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 following directory:

```
<install-dir>/eclipse_support/cdt8.0/eclipse
```

supports and requires Eclipse Platform versions 4.3, 4.2, or 3.8; Eclipse C/C++ Development Tools (CDT) version 8.1 or later; and a functional Java Runtime Environment (JRE) version 6.0 (also called 1.6) update 11 or later.

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.

If your installation of Eclipse already has an earlier Intel® C++ Compiler integration installed, installing the updated integration will not work. You will need to install a fresh version of Eclipse into which you can install the latest Intel C++ Compiler integration. For this same reason, using the Eclipse update mechanism to install a newer Intel C++ Compiler integration will not work.

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 cdt8.0 `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/composer_xe_2015.<n>.<xxx>/eclipse_support/cdt8.0/eclipse`. 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 15.0 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.

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 15.0 User and Reference Guides`.

5.2.1 Integrating the GNU* Project Debugger into Eclipse

See the section GNU* GDB Debugger.

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 4.3 and CDT 8.2 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. Refer to the links on the right for older versions.

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

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 integration-supported Intel compiler versions are 13.0, 14.0, and 15.0. On Intel® 64 architecture systems, only compiler versions 13.0, 14.0, and 15.0 are supported by the integration.

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).

The latest information on Intel® IPP 8.2 can be found in the product release notes under `<install dir>/composer_xe_2015.x.xxx/Documentation/<locale>/ipp/ReleaseNotes.htm`.

For detailed information about Intel 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://intel.ly/OG5IF7; and the Intel IPP Release Notes at http://intel.ly/1uj984p.
- **Documentation, help, and samples**: see the documentation links on the IPP product page at: http://intel.ly/OG5IF7.

6.1 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://intel.ly/ndrGnR

7 Intel® Math Kernel Library

This section summarizes changes, new features and late-breaking news about this version of the Intel® Math Kernel Library (Intel MKL). All the bug fixes can be found here: http://intel.ly/RGiGV9

7.1 Changes in This Version

7.1.1 What's New in Intel MKL 11.2

- Intel MKL now provides optimizations for all Intel® Atom™ processors that support Intel® Streaming SIMD Extensions 4.1 (Intel® SSE4.1) and Intel® Streaming SIMD Extensions 4.2 (Intel® SSE4.2) instruction sets
- Introduced support for Intel® Advanced Vector Extensions 512 (Intel® AVX-512) instruction set with limited optimizations in BLAS, DFT and VML
- Introduced Verbose support for BLAS and LAPACK domains, which enables users to capture the input parameters to Intel MKL function calls
- Introduced support for Intel® MPI Library 5.0
- Introduced the Intel Math Kernel Library Cookbook (http://software.intel.com/en-us/mkl_cookbook) , a new document that describes how to use Intel MKL routines to solve certain complex problems
- Introduced the MKL_DIRECT_CALL or MKL_DIRECT_CALL_SEQ compilation feature that provides ?GEMM small matrix performance improvements for all processors (see the Intel® Math Kernel Library User's Guide for more details)
- Added the ability to link a Single Dynamic Library (mkl_rt) on Intel® Many Integrated Core Architecture (Intel® MIC Architecture)
- Added a customizable error handler. See the Intel Math Kernel Library Reference Manual description of mkl_set_exit_handler() for further details
- Extended the Intel® Xeon Phi™ coprocessor Automatic Offload feature with a resource sharing mechanism.See the Intel Math Kernel Library Reference Manual for the description of mkl_mic_set_resource_limit() function and the MKL_MIC_RESOURCE_LIMIT environment variable for further details
- Parallel Direct Sparse Solver for Clusters:
    - Introduced Parallel Direct Sparse Solver for Clusters, a distributed memory version of Intel MKL PARDISO direct sparse solver
    - Improved performance of the matrix gather step for distributed matrices
    - Enabled reuse of reordering information on multiple factorization steps
    - Added distributed CSR format, support of distributed matrices, RHS, and distributed solutions
    - Added support of solving of systems with multiple right hand sides
    - Added cluster support of factorization and solving steps
    - Added support for pure MPI mode and support for single OpenMP* thread in hybrid configurations
- BLAS:
    - Improved threaded performance of ?GEMM for all 64-bit architectures supporting Intel® Advanced Vector Extensions 2 (Intel® AVX2)
    - Optimized ?GEMM, ?TRSM, DTRMM for the Intel AVX-512 instruction set
    - Improved performance of ?GEMM for outer product [large m, large n, small k] and tall skinny matrices [large m, medium n, small k] on Intel MIC Architecture
    - Improved performance of ?TRSM and ?SYMM in Automatic Offload mode on Intel MIC Architecture

    - Improved performance of Level 3 BLAS functions for 64-bit processors supporting Intel AVX2
    - Improved ?GEMM performance on small matrices for all processors when MKL_DIRECT_CALL or MKL_DIRECT_CALL_SEQ is defined during compilation (see the Intel® Math Kernel Library User's Guide for more details)
    - Improved performance of DGER and DGEMM for the beta=1, k=1 case for 64-bit processors supporting Intel SSE4.2, Intel® Advanced Vector Extensions (Intel® AVX), and Intel AVX2 instruction sets
    - Optimized (D/Z)AXPY for the Intel AVX-512 instruction set
    - Optimized ?COPY for Intel AVX2 and AVX512 instruction sets
    - Optimized DGEMV for Intel AVX-512 instruction set
    - Improved performance of SSYR2K for 64-bit processors supporting Intel AVX and Intel AVX2
    - Improved threaded performance of ?AXPBY for all Intel processors
    - Improved DTRMM performance for the side=R, uplo={U,L}, transa=N, diag={N,U} cases for Intel AVX-512
- LINPACK:
    - Improved performance of matrix generation in the heterogeneous Intel® Optimized MP LINPACK Benchmark for Clusters
    - Intel MIC Architecture offload option of the Intel Optimized MP LINPACK Benchmark for Clusters package now supports Intel AVX2 hosts
    - Improved performance of the Intel Optimized MP LINPACK for Clusters package for 64-bit processors supporting Intel AVX2
- LAPACK:
    - Improved performance of ?(SY/HE)RDB
    - Improved performance of ?(SY/HE)EV when eigenvectors are needed
    - Improved performance of ?(SY/HE)(EV/EVR/EVD) when eigenvectors are not needed
    - Improved performance of ?GELQF,?GELS and ?GELSS for underdetermined case (M less than N)
    - Improved performance of ?GEHRD,?GEEV and ?GEES
    - Improved performance of NaN checkers in LAPACKE interfaces
    - Improved performance of ?GELSX, ?GGSVP
    - Improved performance of ?(SY/HE)(EV/EVR/EVD) when eigenvectors are not needed
    - Improved performance of ?GETRF
    - Improved performance of (S/D)GE(SVD/SDD) when M>=N and singular vectors are not needed
    - Improved performance of ?POTRF UPLO=U in Automatic Offload mode on Intel MIC Architecture
    - Added Automatic Offload for ?SYRDB on Intel MIC Architecture, which speeds up ?SY(EV/EVD/EVR) when eigenvectors are not needed
- PBLAS and ScaLAPACK:

    - Enabled Automatic Offload in P?GEMM routines for large distribution blocking factors
- Sparse BLAS:
    - Optimized SpMV kernels for Intel AVX-512 instruction set
    - Added release example for diagonal format use in Sparse BLAS
    - Improved Sparse BLAS level 2 and 3 performance for systems supporting Intel SSE4.2, Intel AVX and Intel AVX2 instruction sets
- Intel MKL PARDISO:
    - Added the ability to store Intel MKL PARDISO handle to the disk for future use at any solver stage
    - Added pivot control support for unsymmetric matrices and out-of-core mode
    - Added diagonal extraction support for unsymmetric matrices and out-of-core mode
    - Added example demonstrating use of Intel MKL PARDISO as iterative solver for non-linear systems
    - Added capability to free memory taken by original matrix after factorization stage if iterative refinement is disabled
    - Improved memory estimation of out-of-core (OOC) portion size for reordering algorithm leading to improved factorization-solve performance in OOC mode
    - Improved message output from Intel MKL PARDISO
    - Added support of zero pivot during factorization for structurally symmetric cases
- Poisson library:
    - Added example demonstrating use of the Intel MKL Poisson library as a preconditioner for linear systems solves
- Extended Eigensolver:
    - Improved message output
    - Improved examples
    - Added input and output iparm parameters in predefined interfaces for solving sparse problems
- FFT:
    - Optimized FFTs for the Intel AVX-512 instruction set
    - Improved performance for non-power-of-2 length on Intel® MIC Architecture
- VML: Added v[d|s]Frac function computing fractional part for each vector element
- VSL RNG:
    - Added support of ntrial=0 in Binomial Random Number Generator
    - Improved performance of MRG32K3A and MT2203 BRNGs on Intel MIC Architecture
    - Improved performance of MT2203 BRNG on CPUs supporting Intel AVX and Intel AVX2 instruction sets
- VSL Summary Statistics:
    - Added support for group/pooled mean estimates (VSL_SS_GROUP_MEAN/VSL_SS_POOLED_MEAN)
- Data Fitting: Fixed incorrect behavior of the natural cubic spline construction function when number of breakpoints is 2 or 3

- Introduced an Intel MKL mode that ignores all settings specified by Intel MKL environment variables
    - User can set up the mode by calling mkl_set_env_mode() routine which directs Intel MKL to ignore all environment settings specific to Intel MKL so that all Intel MKL related environment variables such as MKL_NUM_THREADS, MKL_DYNAMIC, MKL_MIC_ENABLE and others are ignored; users can instead set needed parameters via Intel MKL service routines such as mkl_set_num_threads() and mkl_mic_enable()

Note: API symbols, order of arguments and link line have changed since Intel MKL 11.2 Beta Update 2. See the Intel® Math Kernel Library User's Guide for more details.

Note: Important deprecations are listed in Intel® Math Kernel Library (Intel® MKL) 11.2 Deprecations

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 (http://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.

The Intel® MKL Extended Eigensolver functionality is based on the Feast Eigenvalue Solver 2.0 http://www.ecs.umass.edu/~polizzi/feast/

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. 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 found in `<installdir>/composer_xe_2015.x.xxx/Documentation/<locale>/tbb`.

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/

The Intel® C++ Compiler, Intel® Debugger, Intel® Integrated Performance Primitives, Intel® Math Kernel Library, and Intel® Threading Building Blocks are provided under Intel’s End User License Agreement (EULA).

The GNU* Project Debugger, GDB is provided under the General GNU Public License GPL V3.

Please consult the licenses included in the distribution for details.