

Intel® Fortran Composer XE 2013 SP1 for Linux*

Installation Guide and Release Notes

Document number: 321415-005US

17 April 2014

Table of Contents

1	Introduction	4
1.1	Change History	5
1.1.1	Product Updates	5
1.1.2	Changes since Intel® Fortran Composer XE 2013.....	5
1.2	Product Contents	6
1.3	System Requirements.....	6
1.4	Documentation.....	7
1.5	Optimization Notice.....	7
1.6	Japanese Language Support.....	8
1.7	Technical Support.....	8
2	Installation.....	8
2.1	GUI installation	9
2.2	Online Installer.....	9
2.2.1	<code>http_proxy</code> is set, but sudo installation still fails to connect	9
2.3	Cluster Installation	9
2.4	Installation of Intel® Manycore Platform Software Stack (Intel® MPSS)	10
2.5	Intel® Software Manager	10
2.6	Silent Install	10
2.7	Using a License Server	10
2.8	Known Installation Issues.....	10
2.9	Installation Folders.....	11
2.10	Removal/Uninstall	13
3	Intel® Fortran Compiler.....	13

3.1	Compatibility	13
3.1.1	Stack Alignment Change for REAL(16) and COMPLEX(16) Datatypes.....	13
3.2	New and Changed Features	14
3.2.1	Features from Fortran 2003	14
3.2.2	Features from OpenMP*	14
3.2.3	Coarrays and Intel® Xeon Phi™ Coprocessors	15
3.2.4	New and Changed Directives.....	15
3.2.5	Other Features.....	15
3.2.6	Change in File Buffering Behavior (13.1)	15
3.3	New and Changed Compiler Options.....	16
3.3.1	New Option Affecting Fortran 2003 VALUE attribute	17
3.3.2	New -[a]xMIC-AVX512 Compiler Option (14.0.1)	17
3.3.3	New -f[no-]mpc_privatize Compiler Option (14.0.1)	17
3.3.4	New -opt-gather-scatter-unroll=n Compiler Option (14.0.1)	18
3.3.5	New -switch fe_debug_use_inherit Internal Command Line Switch (14.0.2)	18
3.4	Establishing the Compiler Environment.....	18
3.5	Known Issues	19
3.5.1	Coarray Issues.....	19
3.6	Coarrays	19
3.6.1	How to Debug a Coarray Application	19
3.6.2	Coarrays with Intel® Xeon Phi™ Coprocessors	21
3.6.3	Coarray Known Issues.....	23
3.7	Fortran 2003 and Fortran 2008 Feature Summary	23
4	GNU* GDB Debugger	27
4.1	Features	27
4.2	Pre-requisites.....	27
4.3	Using GNU* GDB.....	28
4.4	Documentation.....	28
4.5	Known Issues and Changes	28
4.5.1	Safely ending offload debug sessions.....	28
4.5.2	Intel® MIC Architecture-side debugger asserts on setting source directories	28
4.5.3	Offload debugging with Eclipse plug-in does not work with Intel® Manycore Platform Software Stack (Intel® MPSS) 3.2	29

5	Intel® Debugger (IDB)	29
5.1	Support Deprecated for Intel® Debugger	29
5.2	Using Intel® Debugger	29
5.3	Documentation	29
5.4	Debugger Features	30
5.4.1	Main Features of IDB	30
5.4.2	Inspector XE 2011 Update 6 Supports “break into debug” with IDB	30
5.5	Known Issues and Changes	30
5.5.1	Coarray elements cannot be viewed.	30
5.5.2	Using the Intel® Debugger with Intel® MPSS	30
5.5.3	Intel® Debugger does not work with Intel® MPSS 3.1	30
5.5.4	IDB might fail to setup command line argument for debuggee under Eclipse* IDE	30
5.5.5	Thread Data Sharing Filters may not work correctly	31
5.5.6	Core File Debugging	31
5.5.7	Debugger crash if \$HOME not set on calling shell	31
5.5.8	Command line parameter –idb and -dbx not supported	31
5.5.9	Watchpoints limitations	31
5.5.10	Position Independent Executable (PIE) Debugging not Supported	32
5.5.11	Command line parameter –parallel not supported	32
5.5.12	Signals Dialog Not Working	32
5.5.13	Resizing GUI	32
5.5.14	\$cdir, \$cwd Directories	32
5.5.15	info stack Usage	33
5.5.16	\$stepg0 Default Value Changed	33
5.5.17	SIGTRAP error on some Linux* Systems	33
5.5.18	idb GUI cannot be used to debug MPI processes	33
5.5.19	Thread Syncpoint Creation in GUI	33
5.5.20	Data Breakpoint Dialog	33
5.5.21	Stack Alignment for IA-32 Architecture	33
5.5.22	GNOME Environment Issues	34
5.5.23	Accessing Online-Help	34

6	Intel® Xeon Phi™ Coprocessors.....	34
6.1	Introduction.....	34
6.2	Documentation.....	34
6.3	Debugger.....	34
6.3.1	GNU* GDB	34
6.3.2	Intel® Debugger.....	34
6.4	Changes and Known Issues	34
6.4.1	Using offload code in shared libraries requires main program to be linked with – offload=mandatory or –offload=optional option	35
6.4.2	*MIC* tag added to compile-time diagnostics	35
6.4.3	Direct (native) mode requires transferring libiomp5.so to coprocessor	35
6.4.4	Tuning Memory Allocation Performance	35
6.4.5	Stepping “A” Hardware Requires –opt-streaming-stores never	36
6.4.6	Environment Variable for Controlling Offload Behavior	36
6.4.7	OFFLOAD_DEVICES	37
7	Intel® Math Kernel Library	37
7.1	What's New in Intel MKL 11.1 Update 3.....	37
7.2	What's New in Intel MKL 11.1 Update 2.....	38
7.3	What's new in Intel® MKL 11.1 Update 1.....	39
7.4	What's New in Intel® MKL 11.1	40
7.5	Notes	41
7.6	Known Issues	41
7.7	Attributions.....	42
8	Disclaimer and Legal Information.....	42

1 Introduction

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

Due to the nature of this comprehensive integrated software development tools solution, different Intel® Fortran Composer 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.

Intel® Fortran Composer XE 2013 SP1 for Linux* Installation Guide and Release Notes

1.1 Change History

This section highlights important changes 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 Product Updates

Update 3 – April 2014

- Intel® Fortran Compiler [updated to 14.0.3](#)
- Intel® Math Kernel Library [update to 11.1 Update 3](#)

Update 2 – February 2014

- Intel® Fortran Compiler [updated to 14.0.2](#)
 - Added [-switch fe debug use inherit internal command line switch](#)
- Intel® Math Kernel Library [update to 11.1 Update 2](#)

Update 1 – October 2013

- Intel® Fortran Compiler [updated to 14.0.1](#)
 - Added [-assume std_value](#)
 - Added [-\[a\]xMIC-AVX512 compiler option](#)
 - Added [-f\[no-\]mpc_privatize](#)
 - Added [-opt-gather-scatter-unroll=n compiler option](#)
- Intel® Math Kernel Library [update to 11.1 Update 1](#)
- GNU* Project Debugger (GDB) provides register support for Intel® Memory Protection Extensions (Intel® MPX) and Intel® Advanced Vector Extensions 512 (Intel® AVX-512)

1.1.2 Changes since Intel® Fortran Composer XE 2013

- Intel® Fortran Compiler [updated to version 14.0](#)
 - Support added for [running coarray applications on Intel® Xeon Phi™ coprocessors](#)
- Intel® Math Kernel Library [updated to version 11.1](#)
- [GNU* Project Debugger \(GDB\)](#)
- [Intel® Debugger support deprecated](#)
- A [Graphical User Interface form of the installer](#) is provided as an option
- An [Online form of the installer](#), where only required components are downloaded, is provided as an option
- Support for Ubuntu* 13.04 LTS, Fedora* 18, 19, and Debian* 7.0 added.
- Support for the following Linux distributions has been dropped:
 - Fedora 17*
 - Ubuntu 11.10*
 - Pardus 2011.2*
- Corrections to reported problems

1.2 Product Contents

*Intel® Fortran Composer XE 2013 SP1 for Linux** includes the following components:

- Intel® Fortran Compiler XE 14.0 for building applications that run on IA-32, Intel® 64 architecture systems and Intel® Xeon Phi™ coprocessors running the Linux* operating system
- GNU* Project Debugger (GDB*) 7.5
- Intel® Debugger 13.0
- Intel® Math Kernel Library 11.1
- On-disk documentation

1.3 System Requirements

For an explanation of architecture names, see [Intel® Architecture Platform Terminology](#)

- A PC based on an IA-32 or Intel® 64 architecture processor supporting the Intel® Streaming SIMD Extensions 2 (Intel® SSE2) instructions (Intel® Pentium® 4 processor or later, or compatible non-Intel processor)
 - Development of 64-bit applications, and those that offload work to Intel® Xeon Phi™ coprocessors, 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) to be installed from your Linux distribution.
- For the best experience, a multi-core or multi-processor system is recommended
- 1GB of RAM (2GB recommended)
- 2.5GB free disk space for all features
- For Intel® Xeon Phi™ coprocessor development/testing:
 - Intel® Manycore Platform Software Stack (Intel® MPSS)
- For development of IA-32 or Intel® 64 architecture applications, 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):
 - Debian* 6, 7
 - Fedora* 18, 19
 - Red Hat Enterprise Linux* 5 (deprecated), 6
 - SuSE LINUX Enterprise Server* 10 (deprecated), 11 SP2
 - Ubuntu* 12.04 LTS, 13.04
 - Intel® Cluster Ready
- Linux Developer tools component installed, including gcc, g++ and related tools. (this is the list of component versions tested by Intel; other versions may or may not work and are not recommended - please refer to [Technical Support](#) if you have questions)
 - gcc 4.1, 4.3-4.4, 4.6-4.8
 - binutils 2.17, 2.20-2.23

Intel® Fortran Composer XE 2013 SP1 for Linux* Installation Guide and Release Notes

- 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 GNU* GDB

- To use the provided GNU* GDB, Python* version 2.4, 2.6 or 2.7 is required.

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

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

Notes

- The Intel® compilers are tested with a number of different Linux distributions, with different versions of gcc. Some Linux distributions may contain header files different from those we have tested, which may cause problems. The version of glibc you use must be consistent with the version of gcc in use. For best results, use only the gcc versions as supplied with distributions listed above.
- The default for the Intel® compilers is to build IA-32 architecture applications that require a processor supporting the Intel® SSE2 instructions - for example, the Intel® Pentium® 4 processor. A compiler option is available to generate code that will run on any IA-32 architecture processor. However, Intel® MKL requires Intel® SSE2 as a minimum instruction set.
- 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.
- 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 Documentation

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

1.5 Optimization Notice

Optimization Notice

Intel's compilers 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 SSE2, SSE3, and 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. Certain optimizations not specific to Intel microarchitecture are reserved for Intel microprocessors. Please refer to the applicable

product User and Reference Guides for more information regarding the specific instruction sets covered by this notice.

Notice revision #20110804

1.6 Japanese Language Support

Intel® compilers optionally provide support for Japanese language users when the combined English-Japanese product is installed. 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 is not provided with every update of the product.

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 [Changing Language Setting to see English on a Japanese OS environment or Vice Versa on Linux*](#)

1.7 Technical Support

Register your license at the [Intel® Software Development Products Registration Center](#). Registration entitles you to free technical support, product updates and upgrades for the duration of the support term.

For information about how to find Technical Support, Product Updates, User Forums, FAQs, tips and tricks, and other support information, please visit:

<http://www.intel.com/software/products/support/>

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

2 Installation

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

If you received your product on DVD, mount the DVD, change the directory (`cd`) to the top-level directory of the mounted DVD and begin the installation using the command:

```
./install.sh
```

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

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

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

```
./install.sh
```

Follow the prompts to complete installation.

Note that there are several different downloadable files available, each providing different combinations of components. Please read the download web page carefully to determine which file is appropriate for you.

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

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

2.1 GUI installation

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

2.2 Online Installer

The default electronic installation package now consists of 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.

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"
```

2.3 Cluster Installation

If a license for Intel® Cluster Studio XE is present, and the installation detects that the installing system is a member of a cluster, you will have the option of installing on multiple nodes of the cluster.

To install on multiple nodes, follow these steps:

Intel® Fortran Composer XE 2013 SP1 for Linux* Installation Guide and Release Notes

1. Passwordless ssh must be configured among the nodes of the cluster
2. During install step 4, “Options”, select “Cluster installation”.
3. You will be prompted to provide the path to a `machines.LINUX` file with IP addresses, hostnames, or Fully Qualified Domain Names (FQDNs) of the cluster nodes, one per line. The first line is expected to be the current (master) node.
4. Once the `machines.LINUX` file is found, additional options will appear, including “Number of parallel installations” and “Check for shared installation directory”. Select the desired options.
5. Once all options are configured and the install is started, the installation will check connectivity to all the nodes; if successful, it will attempt the install on all indicated nodes.

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

The Intel® Manycore Platform System Software (Intel® MPSS) may be installed before or after installing the Intel® Fortran Composer XE 2013 SP1 for Linux* product.

Using the latest version of Intel® MPSS available is recommended.

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

2.5 Intel® Software Manager

The installation provides the 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 [Intel® Software Improvement Program](#).

2.6 Silent Install

For information on automated or “silent” install capability, please see [Intel® Compilers for Linux* Silent Installation Guide](#).

2.7 Using a License Server

If you have purchased a “floating” license, see Licensing: [Setting Up the Client for a Floating License](#). This article also provides a source for the Intel® License Manager for FLEXlm* product that can be installed on any of a wide variety of systems.

2.8 Known Installation Issues

- When uninstalling a more recent major version of Intel® Composer XE, the symbolic links will be removed even if other versions of Intel Composer XE exist on the system. For example, uninstalling Intel Composer XE 2011 will remove the symbolic links of Intel Composer XE 2013 SP1 at the same installed directory. As a workaround, refer to the Intel® Fortran Composer XE 2013 SP1 for Linux* Installation Guide and Release Notes

other Composer XE's directories explicitly using `link_install.sh` (for example `/opt/intel/link_install.sh -i -p /opt/intel/composer_xe_2013_sp1.<n>.<pkg>`).

- On some versions of Linux, auto-mounted devices do not have the "exec" permission and therefore running the installation script directly from the DVD will result in an error such as:

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

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

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

and then try the installation again.

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

2.9 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
- `mk1` – symbolic link to the directory for the latest installed version of Intel® Math Kernel Library
- `composerxe` – symbolic link to the `composer_xe_2013` directory
- `composer_xe_2013_sp1` – directory containing symbolic links to subdirectories for the latest installed Intel® Composer XE 2013 SP1 product release
- `composer_xe_2013_sp1.<n>.<pkg>` - physical directory containing files for a specific update version. `<n>` is the update number, and `<pkg>` is a package build identifier

Each `composer_xe_2013_sp1` directory contains the following directories that reference the latest installed Intel® Composer XE 2013 SP1 product:

- `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
- `mkl` – symbolic link to the `mkl` directory
- `debugger` – symbolic link to the `debugger` directory
- `man` – symbolic link to the directory containing man pages for the latest installed version
- `Documentation` – symbolic link to the `documentation` directory
- `Samples` – symbolic link to the `samples` directory
- `eclipse_support` – symbolic link to a directory created by the Intel Debugger component that is shared between Intel Fortran and Intel C++. Intel does not provide Eclipse support for Fortran.

Each `composer_xe_2013_sp1.<n>.<pkg>` directory contains the following directories that reference a specific update of the Intel® Composer XE 2013 SP1 compiler:

- `bin` – all executables
- `compiler` – shared libraries and include/header files
- `debugger` – debugger files
- `Documentation` – documentation files
- `eclipse_support` – directory created by the Intel Debugger component that is shared between Intel Fortran and Intel C++. Intel does not provide Eclipse support for Fortran.
- `man` – man pages
- `mkl` – Intel® Math Kernel Library libraries and header files
- `mpirt` – Intel® MPI Library run-time files used by Fortran coarray support
- `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 product update, no matter which version, the latest update of the Intel® Composer XE 2013 SP1 product, or a specific update. Most users will reference `<install-dir>/bin` for the `compilervars.sh` [`.csh`] script, which will always get the latest product installed. This layout should remain stable for future releases.

2.10 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 components installed.

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

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

3 Intel® Fortran Compiler

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

3.1 Compatibility

In general, object code and modules compiled with earlier versions of Intel Fortran Compiler for Linux* (8.0 and later) may be used in a build with version 14. Exceptions include:

- Sources that use the `CLASS` keyword to declare polymorphic variables and which were built with a compiler version earlier than 12.0 must be recompiled.
- Objects built with the multi-file interprocedural optimization (`-ipo`) option must be recompiled with the current version.
- Objects that use the `REAL(16)`, `REAL*16`, `COMPLEX(16)` or `COMPLEX*32` datatypes and which were compiled with versions earlier than 12.0 must be recompiled.
- Objects built for the Intel® 64 architecture with a compiler version earlier than 10.0 and that have module variables must be recompiled. If non-Fortran sources reference these variables, the external names may need to be changed to remove an incorrect leading underscore.
- Modules that specified an `ATTRIBUTES ALIGN` directive outside of a derived type and were compiled with versions earlier than 11.0 must be recompiled. The compiler will notify you if this issue is encountered.
- Modules that specified an `ATTRIBUTES ALIGN` directive inside a derived type declaration cannot be used by compilers older than 13.0.1.

3.1.1 Stack Alignment Change for `REAL(16)` and `COMPLEX(16)` Datatypes

In versions prior to 12.0, when a `REAL(16)` or `COMPLEX(16)` (`REAL*16` or `COMPLEX*32`) item was passed by value, the stack address was aligned at 4 bytes. For improved performance, the version 12 and later compilers align such items at 16 bytes and expects received arguments to be aligned on 16-byte boundaries. This change is also compatible with `gcc`.

This change primarily affects compiler-generated calls to library routines that do computations on REAL(16) values, including intrinsics. If you have code compiled with earlier versions and link it with the version 12 libraries, or have an application linked to the shared version of the Intel run-time libraries, it may give incorrect results.

In order to avoid errors, you must recompile all Fortran sources that use the REAL(16) and COMPLEX(16) datatypes if they were compiled by compiler versions earlier than 12.0.

3.2 New and Changed Features

3.2.1 Features from Fortran 2003

- User-Defined Derived Type I/O

3.2.2 Features from OpenMP*

The following directives, clauses and procedures, from [OpenMP 4.0](#), are supported by the compiler. Some of these features were supported in Intel® Fortran Composer XE 2013 Update 2 based on a preliminary specification, some syntax supported earlier (DECLARE TARGET MIRROR, DECLARE TARGET LINKABLE, MAPTO, MAPFROM, SCRATCH) is no longer supported, and some syntax has changed its meaning since the earlier specification.

For more information, see the compiler documentation or the link to the OpenMP Specification above.

SIMD Directives:

- OMP SIMD
- OMP DECLARE SIMD
- OMP DO SIMD
- OMP PARALLEL DO SIMD

Coprocessor Directives:

- OMP TARGET DATA
- OMP TARGET
- OMP TARGET UPDATE
- OMP DECLARE TARGET

Other Directives:

- OMP PARALLEL PROC_BIND
- OMP TASKGROUP

Clauses:

- MAP

Procedures:

Intel® Fortran Composer XE 2013 SP1 for Linux* Installation Guide and Release Notes

- OMP_GET_DEVICE_NUM
- OMP_GET_PROC_BIND
- OMP_SET_DEVICE_NUM

3.2.2.1 *KMP_PLACE_THREADS Environment Variable (13.1.0)*

This environment variable allows the user to simplify the specification of the number of cores and threads per core used by an OpenMP application, as an alternative to writing explicit affinity settings or a process affinity mask.

3.2.2.2 *KMP_DYNAMIC_MODE Environment Variable Support for “asat” Deprecated*

Support for “asat” (automatic self-allocating threads) by the environment variable KMP_DYNAMIC_MODE is now deprecated, and will be removed in a future release.

3.2.3 **Coarrays and Intel® Xeon Phi™ Coprocessors**

Support has been added to support development of applications using coarrays that run either natively on Intel® Xeon Phi™ coprocessors or run in a mixed mode with Intel® Xeon Phi coprocessors and an Intel® 64 architecture host system

For more information, see [Coarrays with Intel® Xeon Phi™ Coprocessors](#).

3.2.4 **New and Changed Directives**

The following compiler directives are new or changed in Intel® Composer XE 2013 SP1 – please see the documentation for details:

- [NO]FMA

3.2.5 **Other Features**

For information on these features, please see the compiler documentation.

- ESTABLISHQQ library routine to specify that a user routine is to be called when the Fortran run-time library is about to report a run-time error. This routine is declared in module IFPORT.
- A command line option `–[no-]wrap-margin`, and an environment variable `FORT_FMT_NO_WRAP_MARGIN`, that control whether or not list-directed output begins a new record when the previous record would extend past column 80.
- New predefined preprocessor symbols `__INTEL_COMPILER_UPDATE`, `__INTEL_OFFLOAD`, `__MIC__`
- New Environment variable `FOR_FORCE_STACK_TRACE`. When defined as 1, the compiler provides a traceback when any diagnostic message is issued at runtime. `FOR_FORCE_STACK_TRACE` overrides `FOR_DISABLE_STACK_TRACE`.

3.2.6 **Change in File Buffering Behavior (13.1)**

In product versions prior to Intel® Visual Fortran Composer XE 2013 (compiler version 13.0), the Fortran Runtime Library buffered all input when reading variable length, unformatted sequential file records. This default buffering was accomplished by the Fortran Runtime Library allocating an internal buffer large enough to hold any sized, variable length record in memory. Intel® Fortran Composer XE 2013 SP1 for Linux* Installation Guide and Release Notes

For extremely large records this could result in an excessive use of memory, and in the worst cases could result in available memory being exhausted. The user had no ability to change this default buffering behavior on such READs. There was always the ability to request or deny buffering of these records when writing them, but not when reading them.

This default buffering behavior was changed with the release of Intel® Visual Fortran Composer XE 2013. Beginning with this version, all such records are **not** buffered by default, but rather read directly from disk to the user program's variables. This change helped programs that needed to conserve memory, but could in fact result in a performance degradation when reading records that are made of many small components. Some users have reported this performance degradation.

The Intel® Visual Fortran Composer XE 2013 Update 2 (compiler version 13.1) release of the Fortran Runtime Library now provides a method for a user to choose whether or not to buffer these variable length, unformatted records. The default behavior remains as it was in 13.0; these records are not buffered by default. If you experience performance degradation when using 13.1 with this type of I/O, you can enable buffering of the input the same way that you choose whether to enable buffering of the output of these records – one of the following:

- specifying BUFFERED="YES" on the file's OPEN statement
- specifying the environment variable FORT_BUFFERED to be YES, TRUE or an integer value greater than 0
- specifying `-assume buffered_io` on the compiler command line

In the past, these mechanisms applied only when issuing a WRITE of variable length, unformatted, sequential files. They can now be used to request that the Fortran Runtime Library buffer all input records from such files, regardless of the size of the records in the file.

Using these mechanisms returns the READING of such records to the pre-13.0 behavior.

3.3 New and Changed Compiler Options

Please refer to the compiler documentation for details

- [-assume std_value](#) (14.0.1)
- [-\[a\]xMIC-AVX512](#) (14.0.1)
- [-f\[no-\]mpc_privatize](#) (14.0.1)
- -fimf-domain-exclusion
- -fma
- -fmerge-constants
- -foptimize-sibling-calls
- -mtune=<arch>
- -openmp-offload
- -openmp-simd
- -opt-assume-safe-padding
- -offload=<arg>

- [-opt-gather-scatter-unroll=n](#) (14.0.1)
- -opt-prefetch-distance
- -opt-streaming-cache-evict
- -opt-threads-per-core
- -static-libstdc++
- [-switch fe debug use inherit](#) (14.0.2)
- -vecabi
- -wrap-margin
- -xATOM_SSE4.2

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

3.3.1 New Option Affecting Fortran 2003 VALUE attribute

The Intel Fortran compiler's implementation of the Fortran 2003 VALUE attribute does not match the specification of the standard when used in a procedure that does not have the BIND(C) language binding specification. The compiler's default behavior is to treat the Fortran 2003 VALUE attribute the same as a DEC\$ ATTRIBUTES VALUE directive causing the argument to be passed and received "by value". The standard specifies that a redefinable copy of the argument is to be passed instead. This incorrect behavior also prevents the use of the OPTIONAL attribute with VALUE. Note that if the procedure does have the BIND(C) language binding specification, then the implementation matches the standard and arguments with the VALUE attribute are passed and received by value.

In the version 14 compiler, the standard-conforming implementation is available but is not enabled by default, as this could cause problems for existing applications that assumed the previous implementation. To get the standard behavior add the /assume:std_value (Windows) or -assume std_value (Linux and OS X) compiler option. This option is not documented. When using Visual Studio on Windows, this option can be added under Command Line > Additional Options. If /standard-semantics (Windows) or -standard-semantics (Linux and OS X) is in effect, this implies std_value.

A future major release of the Intel Fortran compiler may change the default behavior for VALUE to match the standard.

3.3.2 New [-\[a\]xMIC-AVX512](#) Compiler Option (14.0.1)

Optimizes for Intel(R) processors that support Intel(R) Advanced Vector Extensions 512 (Intel(R) AVX-512) instructions. May generate Intel(R) AVX-512 Foundation instructions, Intel(R) AVX-512 Conflict Detection instructions, Intel(R) AVX-512 Exponential and Reciprocal instructions, Intel(R) AVX-512 Prefetch instructions for Intel(R) processors, and the instructions enabled with CORE-AVX2.

3.3.3 New [-f\[no-\]mpc_privatize](#) Compiler Option (14.0.1)

This option enables privatization of all static data for the MultiProcessor Communications environment (MPC) unified parallel runtime. This will cause calls to extended thread local

storage resolution run-time routines that are not supported on standard Linux distributions. This option is only usable in conjunction with the MPC unified parallel runtime. The default is `-fno-mpc_privatize`. This option is only available in the Intel C++ and Fortran Composer XE 2013 SP1 for Linux targeting Intel® 64 architecture and Intel® MIC Architecture.

3.3.4 New `-opt-gather-scatter-unroll=n` Compiler Option (14.0.1)

This option lets you specify an alternative loop unroll sequence for gather and scatter loops on Intel® Many Integrated Core Architecture (Intel® MIC Architecture) and may improve performance of gather/scatter operations. This option only applies to Intel® MIC Architecture.

3.3.5 New `-switch fe_debug_use_inherit` Internal Command Line Switch (14.0.2)

Examining the parent fields of an extended derived type in the gdb debugger currently requires that you also list the parent name. Add the internal command line switch `-switch fe_debug_use_inherit` to your debug command line, and you will be able to use the abbreviated syntax to examine the parent field.

For example:

```
TYPE BASE
    integer Base_Counter
END TYPE BASE

TYPE, EXTENDS (BASE) :: Type2
END TYPE TYPE2

TYPE(Type2) :: Foo
```

It is legal Fortran to reference either `Foo%Base_Counter` or `Foo%base%base_counter`. Without the `fe_debug_use_inherit` switch, you cannot use the former form within gdb. Please note however, if you do set the `fe_debug_use_inherit` switch, you are unable to use the latter form within gdb.

This internal command line switch will not be supported in compiler version 15.0 as this feature will then be enabled by default.

3.4 Establishing the Compiler Environment

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

The command takes the form:

Intel® Fortran Composer XE 2013 SP1 for Linux* Installation Guide and Release Notes

```
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® C++ Compiler.

3.5 Known Issues

3.5.1 Coarray Issues

For a list of known issues with Fortran 2008 Coarray support, see [Coarray Known Issues](#).

3.6 Coarrays

No special procedure is necessary to run a program that uses coarrays in a shared-memory configuration; you simply run the executable file. The underlying parallelization implementation is Intel® MPI. Installation of the compiler automatically installs the necessary Intel® MPI run-time libraries to run on shared memory. The Intel® Cluster Toolkit product (optional) installs the necessary Intel® MPI run-time libraries to run on distributed memory. Use of coarray applications with any other MPI implementation, or with OpenMP*, is not supported.

By default, the number of images created is equal to the number of execution units on the current system. You can override that by specifying the option `-coarray-num-images <n>` on the `ifort` command that compiles the main program. You can also specify the number of images in an environment variable `FOR_COARRAY_NUM_IMAGES`.

3.6.1 How to Debug a Coarray Application

The following instructions describe how to debug a Coarray application.

1. Add a stall loop to your application before the area of code you wish to debug, e.g.:

```
LOGICAL VOLATILE :: WAIT_FOR_DEBUGGER
LOGICAL, VOLATILE :: TICK
:
DO WHILE(WAIT_FOR_DEBUGGER)
  TICK = .NOT. TICK
END DO
! Code you want to debug is here
!
```

The use of `VOLATILE` is required to ensure that the loop will not be removed by the compiler. If the problem is only found on one image, you can wrap the loop in `IF (THIS_IMAGE() .EQ. 4) THEN` or the like.

2. Compile and link with debug enabled (`-g`).
3. Create at least `N+1` terminal windows on the machine where the application will be running, where `N` is the number of images your application will have.
4. In a terminal window, start the application.
`linuxprompt> ./my_app`

5. In each of the other terminal windows, set your default directory to be the same as the location of the application executable. Use the `ps` command in one of the windows to find out which processes are running your application:

```
linuxprompt> ps -ef | grep 'whoami' | grep my_app
```

There will be several processes. The oldest is the one you started in step 4 – it has run the MPI launcher and is now waiting for the others to terminate. Do not debug it.

The others will look like this:

```
<your-user-name> 25653 25650 98 15:06 ?          00:00:49 my_app
<your-user-name> 25654 25651 97 15:06 ?          00:00:48 my_app
<your-user-name> 25655 25649 98 15:06 ?          00:00:49 my_app
```

The first number is the PID of the process (e.g., 25653 in the first line).

Call the PIDs of these N processes running "my_app" P1, P2, P3 and so on.

6. In each window other than the first, start your debugger and set it to stop processes when attached:

```
linuxprompt> idb -idb
(idb) set $stoponattach = 1
```

or

```
linuxprompt> gdb-ia
```

7. Attach to one of the processes (e.g. to P1 in window 1, to P2 in window 2, etc.)

```
(idb) attach <P1> my_app
```

or

```
(gdb) attach <P1>
```

8. Get execution out of the stall loop:

```
(idb) assign WAIT_FOR_DEBUGGER = .FALSE.
```

or

```
(gdb) set WAIT_FOR_DEBUGGER = .false.
```

9. You can now debug.

If you are using IDB, you can use the multiprocess capability of IDB to have only one debugger window instead of N. First, attach to each process and get out of the loop (steps 7 and 8).

```
(idb) attach <P1> my_app
(idb) assign WAIT_FOR_DEBUGGER = .FALSE.
(idb) attach <P2> my_app
(idb) assign WAIT_FOR_DEBUGGER = .FALSE.
(idb) attach <P3> my_app
(idb) assign WAIT_FOR_DEBUGGER = .FALSE.
```

Use the "process" command to switch debugging focus from one process to another:

```
(idb) process <Pn>
```

Processes not focused on will remain in the state they were left in: with breakpoints and watchpoints set but not running.

3.6.2 Coarrays with Intel® Xeon Phi™ Coprocessors

As of Intel® Fortran Composer XE 2013 SP1, support has been added for using the Fortran 2008 coarray feature with Intel® Xeon Phi™ coprocessors implementing the Intel® Many Integrated Core (Intel® MIC) architecture. You can choose among three execution models:

- A coarray application that has offload regions
- A coarray application that runs on both the coprocessor and the Intel® Xeon processor (heterogeneous)
- A coarray application that runs natively on the coprocessor

For all these modes, as with any Intel® MIC Architecture application, you must copy to the coprocessor all library shared objects referenced by the application. This will include Intel® MPI libraries as well as Intel® Fortran libraries such as libicaf.so.

For example:

```
sudo scp
/opt/intel/composer_xe_2013_sp1.NN/compiler/lib/mic/libicaf.so
mic0:/lib64/libicaf.so
```

```
sudo scp
/opt/intel/composer_xe_2013_sp1.NN/compiler/lib/mic/libintlc.so
mic0:/lib64/libintlc.so
```

```
sudo scp /opt/intel/composer_xe_2013_sp1.NN/mpirt/lib/mic/libmpi_mt.so
mic0:/lib64/libmpi_mt.so
```

```

sudo scp
/opt/intel/composer_xe_2013_sp1.NN/mpirt/bin/mic/mpiexec.hydra
mic0:/bin/mpiexec.hydra

sudo scp /opt/intel/composer_xe_2013_sp1.NN/mpirt/bin/mic/pmi_proxy
mic0:/bin/pmi_proxy

```

This needs to be done every time the coprocessor is rebooted.

3.6.2.1 Using coarrays with offload regions

Use of offload regions in a coarray application has the following restrictions:

- All accesses to coarrays within an offload region must be to the local copy of the coarray – coindexing is not allowed
- No use of SYNC ALL, SYNC MEMORY, SYNC IMAGES, or LOCK/UNLOCK is allowed in an offload region
- Coarrays must not be allocated or deallocated within an offload region

Please see the documentation on using offload regions for more general information.

3.6.2.2 Heterogeneous coarray application

You can run a coarray application where some of the images run on the Intel® 64 host system and some run on an Intel® Xeon Phi™ coprocessor. This is called “heterogeneous”.

First, you will need to build the application using the `–coarray=coprocessor` option as well as specifying a coarray configuration file. For example:

```

ifort -coarray=coprocessor \
-coarray-config-file=MixedPlatform.conf \
mycoarrayprog.f90 -o mycoarrayprog \

```

In this example, we have named the configuration file `MixedPlatform.conf`, but you can choose any name. This will create two executables, `mycoarrayprog` and `mycoarrayprogMIC`

The Intel® MIC Architecture native executable, `mycoarrayprogMIC`, must be copied to the coprocessor file system.

`MixedPlatform.conf` is an MPI configuration file, and is required to be able to run this heterogeneous configuration. An example configuration file is:

```

-n 4 -genv FOR_ICAF_STATUS=true -host myhostname mycoarrayprog : \
-n 4 -host mic0 /home/mydir/mycoarrayprogMIC

```

The `FOR_ICAF_STATUS=true` phrase is required; this is true whenever you have a configuration file. `myhostname` is the name of your host Intel® 64 architecture system. In this Intel® Fortran Composer XE 2013 SP1 for Linux* Installation Guide and Release Notes

example, `/home/mydir` is the path to the Intel® MIC Architecture executable on the coprocessor – change this as required.

This configuration file runs 4 images on the host, 4 images on the card. You can change the `-n` value as desired.

Before running your executable on the host system, define the environment variable `I_MPI_MIC` to be `ENABLE`. Then run the executable – it will start both on the host and on the coprocessor.

3.6.2.3 Native coarray application on the coprocessor

To build a coarray application that runs natively on the coprocessor only, build it with the `-coarray` and `-mmic` options. A configuration file is not required. For example:

```
ifort -coarray -mmic mycoarrayprog.f90 -o \
mycoarrayprog -L/opt/intel/composer_xe_2013_sp1/mpirt/lib/mic
```

This will create “mycoarrayprog” as an Intel® MIC Architecture native executable.

A convenient way to run the application is to use the `micnativeloadex` utility, for example:

```
/opt/intel/mic/coi/tools/micnativeloadex/release/micnativeloadex \
mycoarrayprog
```

This will run the application on the coprocessor, and will bring over any referenced shared images.

The tool includes a help system, found with

```
/opt/intel/mic/coi/tools/micnativeloadex/release/micnativeloadex -h
```

3.6.3 Coarray Known Issues

The following features are known not to work completely in this version:

- Accessing another image’s value of an `ALLOCATABLE` or `POINTER` component of a derived-type coarray. Some forms of this work, some do not.

3.7 Fortran 2003 and Fortran 2008 Feature Summary

The Intel Fortran Compiler supports many features that are new in Fortran 2003. Additional Fortran 2003 features will appear in future versions. Fortran 2003 features supported by the current compiler include:

- The Fortran character set has been extended to contain the 8-bit ASCII characters `~ \ [ ] ` ^ { } | # @`
- Names of length up to 63 characters
- Statements of up to 256 lines
- Square brackets `[ ]` are permitted to delimit array constructors instead of `( / )`

- Structure constructors with component names and default initialization
- Array constructors with type and character length specifications
- A named PARAMETER constant may be part of a complex constant
- Enumerators
- Allocatable components of derived types
- Allocatable scalar variables
- Deferred-length character entities
- PUBLIC types with PRIVATE components and PRIVATE types with PUBLIC components
- ERRMSG keyword for ALLOCATE and DEALLOCATE
- SOURCE= keyword for ALLOCATE
- Type extension
- CLASS declaration
- Polymorphic entities
- Inheritance association
- Deferred bindings and abstract types
- Type-bound procedures
- TYPE CONTAINS declaration
- ABSTRACT attribute
- DEFERRED attribute
- NON_OVERRIDABLE attribute
- GENERIC keyword for type-bound procedures
- User-defined Derived Type I/O
- FINAL subroutines
- ASYNCHRONOUS attribute and statement
- BIND(C) attribute and statement
- PROTECTED attribute and statement
- VALUE attribute and statement
- VOLATILE attribute and statement
- INTENT attribute for pointer objects
- Reallocation of allocatable variables on the left hand side of an assignment statement when the right hand side differs in shape or length (requires option `-assume realloc_lhs` if not deferred-length character)
- Bounds specification and bounds remapping on a pointer assignment
- ASSOCIATE construct
- SELECT TYPE construct
- In all I/O statements, the following numeric values can be of any kind: UNIT=, IOSTAT=
- NAMELIST I/O is permitted on an internal file
- Restrictions on entities in a NAMELIST group are relaxed
- Changes to how IEEE Infinity and NaN are represented in formatted input and output
- FLUSH statement

- WAIT statement
- ACCESS='STREAM' keyword for OPEN
- ASYNCHRONOUS keyword for OPEN and data transfer statements
- ID keyword for INQUIRE and data transfer statements
- POS keyword for data transfer statements
- PENDING keyword for INQUIRE
- The following OPEN numeric values can be of any kind: RECL=
- The following READ and WRITE numeric values can be of any kind: REC=, SIZE=
- The following INQUIRE numeric values can be of any kind: NEXTREC=, NUMBER=, RECL=, SIZE=
- Recursive I/O is allowed in the case where the new I/O being started is internal I/O that does not modify any internal file other than its own
- IEEE Infinities and NaNs are displayed by formatted output as specified by Fortran 2003
- BLANK, DECIMAL, DELIM, ENCODING, IOMSG, PAD, ROUND, SIGN, SIZE I/O keywords
- DC, DP, RD, RC, RN, RP, RU, RZ format edit descriptors
- In an I/O format, the comma after a P edit descriptor is optional when followed by a repeat specifier
- Rename of user-defined operators in USE
- INTRINSIC and NON_INTRINSIC keywords in USE
- IMPORT statement
- Allocatable dummy arguments
- Allocatable function results
- PROCEDURE declaration
- The keyword MODULE may be omitted from MODULE PROCEDURE in a generic interface block when referring to an external procedure
- Procedure pointers
- ABSTRACT INTERFACE
- PASS and NOPASS attributes
- The COUNT_RATE argument to the SYSTEM_CLOCK intrinsic may be a REAL of any kind
- Execution of a STOP statement displays a warning if an IEEE floating point exception is signaling
- MAXLOC or MINLOC of a zero-sized array returns zero if the option `-assume noold_maxminloc` is specified.
- Type inquiry intrinsic functions
- COMMAND_ARGUMENT_COUNT intrinsic
- EXTENDS_TYPE_OF and SAME_TYPE_AS intrinsic functions
- GET_COMMAND intrinsic
- GET_COMMAND_ARGUMENT intrinsic
- GET_ENVIRONMENT_VARIABLE intrinsic

- IS_IOSTAT_END intrinsic
- IS_IOSTAT_EOR intrinsic
- MAX/MIN/MAXVAL/MINVAL/MAXLOC/MINLOC intrinsics allow CHARACTER arguments
- MOVE_ALLOC intrinsic
- NEW_LINE intrinsic
- SELECTED_CHAR_KIND intrinsic
- The following intrinsics take an optional KIND= argument: ACHAR, COUNT, IACHAR, ICHAR, INDEX, LBOUND, LEN, LEN_TRIM, MAXLOC, MINLOC, SCAN, SHAPE, SIZE, UBOUND, VERIFY
- ISO_C_BINDING intrinsic module
- IEEE_EXCEPTIONS, IEEE_ARITHMETIC and IEEE_FEATURES intrinsic modules
- ISO_FORTRAN_ENV intrinsic module

The following is a partial list of Fortran 2003 features that are unimplemented or are known not to work in this release.

- Parameterized derived types
- Transformational intrinsics, such as MERGE and SPREAD, in initialization expressions

The Intel® Fortran Compiler also supports some features from the Fortran 2008 standard. Additional features will be supported in future releases. Fortran 2008 features supported by the current version include:

- Maximum array rank has been raised to 31 dimensions (Fortran 2008 specifies 15)
- Coarrays
- CODIMENSION attribute
- SYNC ALL statement
- SYNC IMAGES statement
- SYNC MEMORY statement
- CRITICAL and END CRITICAL statements
- LOCK and UNLOCK statements
- ERROR STOP statement
- ALLOCATE and DEALLOCATE may specify coarrays
- Intrinsic procedures ATOMIC_DEFINE, ATOMIC_REF, IMAGE_INDEX, LCOBOUND, NUM_IMAGES, THIS_IMAGE, UCOBOUND
- CONTIGUOUS attribute
- MOLD keyword in ALLOCATE
- DO CONCURRENT
- NEWUNIT keyword in OPEN
- G0 and G0.d format edit descriptor
- Unlimited format item repeat count specifier
- A CONTAINS section may be empty

- Intrinsic procedures BESSEL_J0, BESSEL_J1, BESSEL_JN, BESSEL_YN, BGE, BGT, BLE, BLT, DSHIFTL, DSHIFTR, ERF, ERFC, ERFC_SCALED, GAMMA, HYPOT, IALL, IANY, IPARITY, IS_CONTIGUOUS, LEADZ, LOG_GAMMA, MASKL, MASKR, MERGE_BITS, NORM2, PARITY, POPCNT, POPPAR, SHIFTA, SHIFTL, SHIFTR, STORAGE_SIZE, TRAILZ,
- Additions to intrinsic module ISO_FORTRAN_ENV: ATOMIC_INT_KIND, ATOMIC_LOGICAL_KIND, CHARACTER_KINDS, INTEGER_KINDS, INT8, INT16, INT32, INT64, LOCK_TYPE, LOGICAL_KINDS, REAL_KINDS, REAL32, REAL64, REAL128, STAT_LOCKED, STAT_LOCKED_OTHER_IMAGE, STAT_UNLOCKED
- An OPTIONAL dummy argument that does not have the ALLOCATABLE or POINTER attribute, and which corresponds to an actual argument that: has the ALLOCATABLE attribute and is not allocated, or has the POINTER attribute and is disassociated, or is a reference to the NULL() intrinsic function, is considered not present
- A dummy argument that is a procedure pointer may be associated with an actual argument that is a valid target for the dummy pointer, or is a reference to the intrinsic function NULL. If the actual argument is not a pointer, the dummy argument shall have the INTENT(IN) attribute.

4 GNU* GDB Debugger

This section summarizes the changes, new features, customizations and known issues related to the GNU* GDB provided with Intel® Composer XE 2013 SP1.

4.1 Features

GNU* GDB provided with Intel® Composer XE 2013 SP1 is based on GDB 7.5 with enhancements provided by Intel. This debugger is planned to [replace the Intel® Debugger in a future release](#). In addition to features found in GDB 7.5, 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.
- Improved Fortran support

4.2 Pre-requisites

In order to use the provided GNU* GDB Python* version 2.4, 2.6 or 2.7 is required.

4.3 Using GNU* GDB

GNU* GDB provided with Intel® Composer XE 2013 SP1 comes in different versions:

- IA-32/Intel® 64 debugger:
Debug applications natively on IA-32 or Intel® 64 systems.
- 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. This only works for native Intel® Xeon Phi coprocessor applications.
 - Use the Eclipse* IDE plugin. This works only for offload enabled Intel® Xeon Phi coprocessor applications. Native applications need to be debugged with the command line version.

Instructions on how to use GNU* GDB can be found in the [documentation](#)

4.4 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/debugger/gdb/mic/eclmigdb_config_guide.pdf

4.5 Known Issues and Changes

4.5.1 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.

4.5.2 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

Intel® Fortran Composer XE 2013 SP1 for Linux* Installation Guide and Release Notes

source directory settings. The debugger will prompt you to browse for files it cannot locate automatically.

4.5.3 Offload debugging with Eclipse plug-in does not work with Intel® Manycore Platform Software Stack (Intel® MPSS) 3.2

Offload debugging with the Eclipse plug-in from Intel® Composer XE 2013 SP1 does not work with Intel MPSS 3.2. A configuration file which is required for operation by the Intel Composer XE 2013 SP1 package has been removed from Intel MPSS 3.2. Previous Intel MPSS versions are not affected. A future Intel MPSS version will fix this problem.

5 Intel® Debugger (IDB)

Intel® Debugger (IDB) is available as host debugger for IA-32 and Intel® 64 applications, as well as for the Intel® Xeon Phi™ coprocessor.

5.1 Support Deprecated for Intel® Debugger

In a future major release of the product, the Intel® Debugger may be removed. This impacts all components and features described in this section.

New users should use the [GNU* GDB debugger components](#) instead.

5.2 Using Intel® Debugger

Intel® Debugger provided with Intel® Composer XE 2013 SP1 comes in different versions:

- IA-32/Intel® 64 debugger:
Debug applications natively on IA-32 or Intel® 64 systems.
- 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 (`idbserver_mic`) on the coprocessor.
There are two options:
 - Use the command line version of the debugger. This only works for native Intel® Xeon Phi coprocessor applications.
 - Use the Eclipse* IDE plugin. This works for both native and offload enabled Intel® Xeon Phi coprocessor applications.

Instructions on how to use Intel® Debugger can be found in the [documentation](#).

5.3 Documentation

Documentation for the Intel® Debugger can be found here:

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

5.4 Debugger Features

5.4.1 Main Features of IDB

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

5.4.2 Inspector XE 2011 Update 6 Supports “break into debug” with IDB

Inspector XE 2011 Update 6 now supports “break into debug” mode with the Composer XE 2011 Update 6 and later versions of IDB. Refer to the Inspector XE 2011 Release Notes for more information.

5.5 Known Issues and Changes

5.5.1 Coarray elements cannot be viewed.

The IDB Debugger cannot view coarray elements. Please refer to ['How to Debug a Coarray Application'](#) where a workaround is described.

5.5.2 Using the Intel® Debugger with Intel® MPSS

When using the Intel® Debugger for Intel® Many Integrated Core Architecture the following limitations apply:

- When debugging native coprocessor applications on the command line, the remote debug agent `idbserver_mic` is uploaded and started using scp/ssh. This implies that the user id used to start `idbc_mic` must also exist on the coprocessor. Unless passwordless authentication has been configured for this user id, scp and ssh will require a password being typed.
- When debugging heterogeneous applications on the command line, the offload process is started as root. Using `idbc_mic` with a different user id than root will cause the offload process to not be visible by the remote debug server `idbserver_mic`. The workaround is to launch the command line debugger `idbc_mic` as root. Alternatively the options `-mpm-launch=1 -mpm-cardid=<card-id>` can be added to the default launch options: `idbc_mic -mpm-launch=1 -mpm-cardid=<card-id> -tco -rconnect=tcpip:<cardip>:<port>`

5.5.3 Intel® Debugger does not work with Intel® MPSS 3.1

Intel® MPSS 3.1 is not compatible with the Intel® Debugger. Debugging with Intel® MPSS 3.1 is only possible with the provided [GNU* GDB version](#)

5.5.4 IDB might fail to setup command line argument for debuggee under Eclipse* IDE

The debugger might not set the command line argument for the debuggee correctly under Eclipse* IDE when loading an application using the `'file'` command in GDB mode. The debuggee may abort with the message:

```
*** abort -internal failure : get_command_argument failed
```

Intel® Fortran Composer XE 2013 SP1 for Linux* Installation Guide and Release Notes

In this case, add the executable to the command line argument of IDB.

5.5.4.1 Eclipse* IDE fails to display local variables

Local variables cannot be seen under the Eclipse* IDE environment while debugging an application.

Workaround:

Enter the local variable into the “Expressions” window to get its value.

5.5.5 Thread Data Sharing Filters may not work correctly

Setting Thread Data Sharing Filters may lead to unexpected behavior of the debugger. It may happen that threads will not continue after a data sharing detection and the debugger may exit with a SIG SEGV.

If you encounter issues related to Data Sharing Detection with filters enabled, disable all filters in the ‘Thread Data Sharing Filters’ window context menu.

5.5.6 Core File Debugging

To be able to debug core files you need to start the debugger (command line debugger `idbc` or GUI debugger `idb`) with command-line options as follows:

```
idb|idbc <executable> <corefile>
```

<or>

```
idb|idbc <executable> -core <corefile>
```

Once started with a core file, the debugger is not able to debug a live process e.g. attaching or creating a new process. Also, when debugging a live process a core file cannot be debugged.

5.5.7 Debugger crash if \$HOME not set on calling shell

The debugger will end with a “Segmentation fault” if no \$HOME environment variable is set on the shell the debugger is started from.

5.5.8 Command line parameter `-idb` and `-dbx` not supported

The debugger command line parameters `-idb` and `-dbx` are not supported in conjunction with the debugger GUI.

5.5.9 Watchpoints limitations

For IA-32 and Intel® 64 architecture systems there are the following limitations (if possible IDB will raise appropriate error messages to assist the user):

- Possible sizes of the watched memory areas are only 1, 2, 4 or 8 (Intel® 64 architecture only) bytes.
- The start address of the watched memory area has to be aligned with its size. For example it is not possible to watch 2 bytes starting with an odd address.

- There is only support for a maximum of 4 active/enabled watchpoints. Unused ones can be disabled to free resources and to enable/create other ones.
- Only the following access modes are supported:
 - Write: trigger on write accesses
 - Any: trigger on either write or read accesses
 - Changed: trigger on write accesses that actually changed the value
- Watched memory areas must not overlap each other.
- Watchpoints are not scope related but tied to a process. As long as a process exists the watchpoints are active/enabled. Only if the process is terminated (e.g. rerun) will the watchpoints will be disabled. They can be enabled again if the user wishes to do so.
- Using the debugger to access the watched memory area (e.g. assign a different value to a variable) bypasses the hardware detection. Hence watchpoints only trigger if the debuggee itself accessed the watched memory area.
- If the debuggee is running on a guest OS inside a virtual machine, stepping over an instruction or code line might continue the process without stopping. Watchpoints are only guaranteed to work when the debuggee runs on real hardware.

5.5.10 Position Independent Executable (PIE) Debugging not Supported

On some systems the compiler is tuned to produce Position Independent Executable (PIE) code. In those cases the flag `-fno-pie` has to be used both for compilation and linking, otherwise the application cannot be debugged.

5.5.11 Command line parameter `-parallel` not supported

The debugger command line parameter `-parallel` is not supported on the shell command prompt or on the Console Window of the Debugger GUI.

5.5.12 Signals Dialog Not Working

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

5.5.13 Resizing GUI

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

5.5.14 `$cdir`, `$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 is 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.

5.5.15 `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.

5.5.16 `$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
```

5.5.17 SIGTRAP error on some Linux* Systems

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

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

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

5.5.18 `idb` GUI cannot be used to debug MPI processes

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

5.5.19 Thread Syncpoint Creation in GUI

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

5.5.20 Data Breakpoint Dialog

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

5.5.21 Stack Alignment for IA-32 Architecture

Due to changes in the default stack alignment for the IA-32 architecture, the usage of inferior calls (i.e. evaluation of expressions that cause execution of debuggee code) might fail. This can cause as well crashes of the debuggee and therefore a restart of the debug session. If you need Intel® Fortran Composer XE 2013 SP1 for Linux* Installation Guide and Release Notes

to use this feature, make sure to compile your code with 4 byte stack alignment by proper usage of the `-falign-stack=<mode>` option.

5.5.22 GNOME Environment Issues

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

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

5.5.23 Accessing Online-Help

On systems where the Online-Help is not accessible from the IDB Debugger GUI Help menu, you can access the web-based debugger documentation from <http://intel.ly/o5DMp9>

6 Intel® Xeon Phi™ Coprocessors

This section summarizes changes, new features and late-breaking news about developing for Intel® Xeon Phi™ coprocessors using Intel® Composer XE 2013 for Linux*

6.1 Introduction

Intel® Fortran Composer XE 2013 supports development of applications that offload work to an Intel® MIC Architecture coprocessor (Intel® Xeon Phi™ product family). These sections of code run on the Intel® Xeon Phi™ coprocessor if it is available. Otherwise, they run on the host CPU. Development of applications that run natively on Intel® Xeon Phi™ coprocessors is also supported.

This document uses the terms *coprocessor* and *target* to refer to the target of an offload operation.

6.2 Documentation

For the latest documentation updates, please [see Intel® Composer XE 2013 Documentation Updates for Intel® MIC Architecture](#).

6.3 Debugger

6.3.1 GNU* GDB

See the section [GNU* GDB Debugger](#).

6.3.2 Intel® Debugger

See the section [Intel® Debugger \(IDB\)](#).

6.4 Changes and Known Issues

This section corrects or adds to the product documentation.

6.4.1 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.

6.4.2 *MIC* tag added to compile-time diagnostics

The compiler diagnostics infrastructure has been 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 offload directives are seen.

The new tag permits easier association with either the CPU or Target compilation.

6.4.3 Direct (native) mode requires transferring `libiomp5.so` to coprocessor

The Intel® Manycore Platform Software Stack (MPSS) does not include Intel® compiler libraries typically found under `/lib`.

When running applications in direct mode (i.e. on the coprocessor), users must first upload (via `scp`) a copy of any shared object libraries the application uses. For example, the OpenMP* library (`<install_dir>/compiler/lib/mic/libiomp5.so`) should be copied to the coprocessor (device names will be of the format `micN`, where the first coprocessor will be named `mic0`, the second `mic1`, and so on) before running the application.

Failure to make this library available will result in a run-time failure, such as:

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

Some applications may require uploading additional libraries.

6.4.4 Tuning Memory Allocation Performance

The following text replaces information on this topic in the product documentation.

For user-allocated data on the coprocessor, using large (2 MB) page allocations via `mmap`, instead of `malloc` or `ALLOCATE`, may improve application performance.

Not all applications benefit from using a larger page size. In general, the performance impact from a larger page size depends greatly on the data access pattern. If the application accesses multiple data structures that are allocated in different pages, having only limit TLB entries for 2 MB pages on the coprocessor can cause performance degradation.

The default page size is 4KB for `malloc` and `ALLOCATE`.

Intel® Fortran Composer XE 2013 SP1 for Linux* Installation Guide and Release Notes

6.4.5 Stepping “A” Hardware Requires `–opt-streaming-stores never`

If your Intel® Xeon Phi™ coprocessor is hardware stepping “A”, you must use the `–opt-streaming-stores never` option when compiling your application as otherwise the compiler may generate instructions not supported by the hardware. Stepping “B” and later hardware support the new instructions.

6.4.6 Environment Variable for Controlling Offload Behavior

Several additional environment variables are available for controlling offload behavior.

6.4.6.1 `MIC_USE_2MB_BUFFERS`

Sets the threshold for creating buffers with large pages. A buffer is created with the large pages hint if its size exceeds the threshold value.

Example:

```
// any variable allocated on a coprocessor that is equal to
// or greater than 100KB in size will be allocated in large pages.
setenv MIC_USE_2MB_BUFFERS 100k
```

6.4.6.2 `MIC_STACKSIZE`

Sets the size of the offload process stack for all Intel® Xeon Phi™ coprocessors used in the application. This is the overall stack size. Use `MIC_OMP_STACKSIZE` to modify the size of each OpenMP* thread.

Example:

```
setenv MIC_STACKSIZE 100M // Sets MIC stack to 100 MB
```

6.4.6.3 `MIC_ENV_PREFIX`

This is the general mechanism to pass environment variable values to each Intel® Xeon Phi™ coprocessor.

The value of `MIC_ENV_PREFIX` sets the value of the prefix which is used to recognize environment variable values intended for coprocessors. For example,

```
setenv MIC_ENV_PREFIX MYCARDS
```

will use “MYCARDS” as the string that indicates that an environment variable is intended for a specific coprocessor.

Environment variable values of the form

```
<mic-prefix>_<var>=<value>
```

will send `<var>=<value>` to each coprocessor.

Environment variable values of the form `<mic-prefix>_<card-number>_<var>=<value>`

will send `<var>=<value>` to the coprocessor numbered `<card-number>`.

Environment variable values of the form

`<mic-prefix>_ENV=<variable1=value1|variable2=value2>`

will send `<variable1>=<value1>` and `<variable2>=<value2>` to each coprocessor.

Environment variable values of the form

`<mic-prefix>_<card-number>_ENV=<variable1=value1|variable2=value2>`

will send `<variable1>=<value1>` and `<variable2>=<value2>` to the coprocessor numbered `<card-number>`.

Examples:

```
setenv MIC_ENV_PREFIX PHI      // Defines the prefix to be used
setenv PHI_ABCD abcd           // Sets ABCD=abcd on all coprocessors
setenv PHI_2_EFGH efgh        // Sets EFGH=efgh on coprocessor 2
setenv PHI_VAR X=x|Y=y        // Sets X=x and Y=y on all coprocessors
setenv PHI_4_VAR P=p|Q=q      // Sets P=p and Q=q on coprocessor 4
```

6.4.7 OFFLOAD_DEVICES

The environment variable `OFFLOAD_DEVICES` restricts the process to use only the coprocessors specified as the value of the variable. `<value>` is a comma separated list of physical device numbers in the range 0 to `(number_of_devices_in_the_system-1)`.

Devices available for offloading are numbered logically. That is `_Offload_number_of_devices()` returns the number of allowed devices and device indexes specified in the target specifier of an offload directive are in the range 0 to `(number_of_allowed_devices-1)`.

Example

```
setenv OFFLOAD_DEVICES "1,2"
```

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).

7.1 What's New in Intel MKL 11.1 Update 3

- BLAS:
 - Improved performance of Level 3 BLAS functions for 64-bit processors supporting Intel® Advanced Vector Extensions 2 (Intel® AVX2)
 - Improved performance of matrix generation in the heterogeneous Intel® Optimized MP LINPACK Benchmark for Clusters

- Optimized ?GEMM, ?TRSM, DTRMM for the Intel® Advanced Vector Extensions 512 (Intel® AVX-512) instruction set
- LAPACK:
 - Improved performance of ?(SY/HE)RDB
 - Improved performance of ?(SY/HE)(EV/EVR/EVD) when eigenvectors are not needed
 - Improved performance of ?SY(EV/EVD) when eigenvectors are needed
 - Improved performance of NaN checkers in LAPACKE interfaces
 - Improved performance of DGETRF for processors supporting Intel AVX2
 - Added Automatic Offload for ?SYRDB on Intel® Many Integrated Core Architecture (Intel® MIC Architecture), which speeds up DSY(EV/EVD) when eigenvectors are not needed
- Sparse BLAS:
 - Added example for diagonal format matrix
- Intel MKL PARDISO:
 - Improved memory estimation of out-of-core (OOC) portion size for reordering algorithm leading to improved factorization-solving step performance in OOC mode
 - Added pivot control support for nonsymmetric matrices and OOC mode
 - Added diagonal extraction support for nonsymmetric matrices and OOC mode
- Extended Eigensolver:
 - Improved output message
 - Improved examples
 - Added input and output iparm parameters in predefined interfaces for solving sparse problems
- FFT:
 - Optimized the whole range of FFTs for Intel® Advanced Vector Extensions 512 (Intel® AVX-512) instruction set
 - Improved performance of FFT for lengths which are not powers of two on Intel MIC Architecture

7.2 What's New in Intel MKL 11.1 Update 2

- Introduced support for Intel® Atom™ processors
- BLAS:
 - Improved performance of ?GEMM for m==1 or n==1 on all Intel architectures
 - Improved MP LINPACK performance for systems using Intel® Many Integrated Core Architecture (Intel® MIC Architecture)
 - 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 ?SYMM on Intel MIC Architecture
 - Improved {S/D}GEMM single thread performance on small matrices for 64-bit processors supporting Intel® Advanced Vector Extensions (Intel® AVX) and Intel® Advanced Vector Extensions 2 (Intel® AVX2)
 - Improved DGEMV performance for 64-bit processors supporting Intel AVX2
 - Improved threaded performance of {S,D,C,Z}GEMV for notrans:n>>m and trans:m>>n on all Intel architectures

- Improved DSYR2K performance for 64-bit processors supporting Intel AVX and Intel AVX2
 - Improved DTRMM performance on small matrices (A matrix size ≤ 10) for 64-bit processors supporting Intel AVX and Intel AVX2
 - Reduced stack usage for ZHEMM and ZSYRK
 - Added more detailed error messages for running Offload MP LINPACK scripts with unsupported configurations
- LAPACK:
 - Improved performance of (S/D)SYRDB and (D/S)SYEV for large dimensions and UPLO=L when eigenvectors are needed
 - Improved performance of ?GELQF, ?GELS and ?GELSS for underdetermined case (M
 - Improved performance of ?GEHRD, ?GEEV and ?GEES
 - Added Automatic Offload to Intel® Xeon Phi™ Coprocessor for DSYRDB UPLO=L
- Sparse BLAS:
 - Optimized SpMV kernels for Intel® Advanced Vector Extensions 512 (Intel® AVX-512) instruction set
 - Improved Sparse BLAS level 2 and 3 performance for systems supporting Intel® Streaming SIMD Extensions 4.2 (Intel® SSE4.2), Intel AVX and Intel AVX2 instruction sets
- PARDISO:
 - Improved memory estimation of out-of-core portion size for reordering algorithm leading to improved factorization-solving step performance in OOC mode
- VML:
 - Added v[d]s]Frac function computing fractional part for each vector element
- VSL RNG:
 - Improved performance of MRG32K3A, and MT2203 BRNGs on Intel Xeon Phi coprocessors
 - Improved performance of MT2203 BRNG on CPUs supporting Intel AVX and Intel AVX2 instruction sets
- VSL Summary Statistics:
 - Added support for computation of group/pooled (VSL_SS_GROUP_MEAN/VSL_SS_POOLED_MEAN) mean estimates

7.3 What's new in Intel® MKL 11.1 Update 1

- Introduced support for Intel® AVX-512 instructions set with limited set of optimizations
- Added support for Visual Studio 2013 in the examples
- BLAS:
 - Improved performance of DSDOT, and added support for multiple threads, on all 64-bit Intel processors supporting Intel® Advanced Vector Extensions (Intel® AVX) and Intel® Advanced Vector Extensions 2 (Intel® AVX2)
 - Improved handling of denormals on the diagonal in *TRSM
 - Improved SGEMM performance for small N and large M and K on Intel® Many Integrated Core Architecture (Intel® MIC Architecture)
 - Improved parallel performance of *HEMM on all Intel processors supporting Intel® SSE4.2 and later

- Improved parallel performance of 64-bit *SYRK/*HERK on all Intel processors supporting Intel® SSSE3 and later
 - Improved serial performance of 64-bit {D,S}SYRK on all Intel processors supporting Intel® SSE4.2 and later
 - Improved performance of DTRSM on Intel® MIC Architecture
 - Enhanced Intel® Optimized HPL Benchmark runmultiscript capabilities for Intel processors supporting Intel® AVX
 - Improved Intel® Optimized HPL Benchmark performance on Intel® MIC Architecture
- LAPACK
 - Decreased memory utilization for parallel LAPACK functions (OR/UN)M(QR/RQ/QL/LQ)
 - Decreased stack memory utilization in LAPACK functions
 - Improved performance of (S/D)SYRDB and (S/D)SYEV for large dimensions when eigenvalues are only needed
- ScaLAPACK
 - Updated PBLAS headers to mix default NETLIB and MKL complex datatypes
- DFT: Optimized complex-to-complex and real-to-complex transforms
- Transposition: Improved performance of mkl_?omatcopy routines on tall and skinny matrices
- DFTI interface and FFTW wrappers are now thread safe. Setting NUMBER_OF_USER_THREADS parameter when using MKL DFT from parallel regions became optional.

7.4 What's New in Intel® MKL 11.1

- Conditional Numerical Reproducibility : Introduced support for Conditional Numerical Reproducibility (CNR) mode on unaligned data
- Introduced MP LINPACK support for heterogeneous clusters - clusters whose nodes differ from each other, either by processor type or by having varying number of attached Intel® Xeon Phi™ coprocessors
- Improved performance of CNR=AUTO mode on recent AMD* systems
- BLAS:
 - Improved performance of [S/D]GEMV on all Intel processors supporting Intel® SSE4.2 and later
 - Optimized [D/Z]GEMM and double-precision Level 3 BLAS functions on Intel® Advanced Vector Extensions 2 (Intel® AVX2)
 - Optimized [Z/C]AXPY and [Z/C]DOT[U/C] on Intel® Advanced Vector Extensions (Intel® AVX) and Intel AVX2
 - Optimized sequential version of DTRMM on Intel MIC Architecture
 - Tuned DAXPY on Intel AVX2
- LAPACK:
 - Improved performance of (S/D)SYRDB and (S/D)SYEV for large dimensions when only eigenvalues are needed
 - Improved performance of xGESVD for small sizes like M,N<10
- VSL:
 - Added support and examples for mean absolute deviation

- Improved performance of Weibull Random Number Generator (RNG) for $\alpha=1$
 - Added support of raw and central statistical sums up to the 4th order, matrix of cross-products and median absolute deviation
 - Added a VSL example designed by S. Joe and F. Y. Kuo illustrating usage of Sobol QRNG with direction numbers which supports dimensions up to 21,201
 - Improved performance of SFMT19937 Basic Random Number Generator (BRNG) on Intel MIC Architecture
- DFT:
 - Improved performance of double precision complex-to-complex transforms on Intel MIC Architecture
 - Optimized complex-to-complex DFT on Intel AVX2
 - Optimized complex-to-complex 2D DFT on Intel® Xeon processor E5 v2 series (code named IvyTown)
 - Improved performance for workloads specific to GENE application on Intel Xeon E5-series (Intel AVX) and on Intel AVX2
 - Improved documentation data layout for DFTI compute functions
 - Introduced scaling in large real-to-complex FFTs
- Data Fitting:
 - Improved performance of `df?Interpolate1D` and `df?SearchCells1D` functions on Intel Xeon processors and Intel MIC Architecture
 - Improved performance of `df?construct1d` function for linear and Hermite/Bessel/Akima cubic types of splines on Intel MIC Architecture, Intel® Xeon® processor X5570 and Intel® Xeon® processor E5-2690
- Transposition
 - Improved performance of in-place transposition for square matrices
- Examples and tests for using Intel MKL are now packaged as an archive to shorten the installation time

7.5 Notes

- Intel MKL now provides a choice of components to install. Components necessary for PGI compiler, Compaq Visual Fortran Compiler, SP2DP interface, BLAS95 and LAPACK95 interfaces, Cluster support (ScaLAPACK and Cluster DFT) and Intel MIC Architecture support are not installed unless explicitly selected during installation
- Unaligned CNR is not available for MKL Cluster components (ScaLAPACK and Cluster DFT)
- Examples for using Intel MKL with BOOST/uBLAS and Java have been removed from the product distribution and placed in the following articles:
 - [How to use Intel® MKL with Java*](#)
 - [How to use BOOST* uBLAS with Intel® MKL](#)

7.6 Known Issues

A full list of the known limitations can be found in the [Intel® MKL Article List at Intel® Developer Zone](#)

7.7 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. 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.

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

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_number/

for details.

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

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

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

Copyright © 2014 Intel Corporation. All Rights Reserved.