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

Document number: 321417-005US
30 January 2014

Table of Contents

1 Introduction

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

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 Product Updates

Update 2 – February 2014

- Intel® Visual Fortran Compiler updated to 14.0.2
  - Added /switch:fe_debug_use_inherit internal command line switch
- Intel® Math Kernel Library updated to 11.1 Update 2
- Microsoft* Visual Studio 2013* support
- KMP_DYNMIC_MODE Environment Variable Support for “asat” Deprecated

Update 1 – October 2013

- Intel® Visual Fortran Compiler updated to 14.0.1
  - Added /assume:std_value
  - Added /Q[a]xMIC-AVX512 compiler option
  - Added /Qopt-gather-scatter-unroll=n compiler option
- Intel® Math Kernel Library updated to 11.1 Update 1
- Support for Microsoft Windows 8.1* added

1.2 Changes since Intel® Fortran Composer XE 2013

- Intel® Fortran Compiler updated to version 14.0
  - Support added for developing applications that use Intel® Xeon Phi™ coprocessors
  - Intel® Debugger Extension for Intel® Many Integrated Core Architecture (Intel® MIC Architecture)
- Intel® Math Kernel Library updated to version 11.1
- An online version of the installer, where only required components are downloaded, is provided as an option
- Corrections to reported problems

1.3 Product Contents

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

- Intel® Visual Fortran Compiler XE 14.0 for building applications that run on IA-32 and Intel® 64 architecture systems
- Intel® Math Kernel Library 11.1
- Intel® Debugger Extension for Intel® Many Integrated Core Architecture (Intel® MIC Architecture)
- Fortran Expression Evaluator (FEE) for debugging Fortran applications with Microsoft Visual Studio*
- Integration into Microsoft* development environments
- Microsoft Visual Studio 2010* Shell and Libraries (not included with Evaluation licenses)

- Sample programs
- On-disk documentation

1.4 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
    - For the best experience, a multi-core or multi-processor system is recommended
- 1GB RAM (2GB recommended)
- 2GB free disk space required for all product features and all architectures
- Microsoft Windows XP SP3*, Microsoft Windows 7*, Microsoft Windows 8*, Microsoft Windows 8.1*, Microsoft Windows Server 2012*, Microsoft Windows Server 2008* or Microsoft Windows HPC Server 2008* (embedded editions not supported)
    - Microsoft Windows Server 2008 or Windows HPC Server 2008 requires Microsoft Visual Studio 2013* or Visual Studio 2012* or Visual Studio 2010* or Visual Studio 2010* Shell or Visual Studio 2008* SP1 or Visual Studio 2008* Shell with Visual Studio 2008 SP1 update applied.
    - On Microsoft Windows 8, the product installs into the “Desktop” environment. Development of “New Windows 8* UI” applications is not supported.
- To use the Microsoft Visual Studio development environment or command-line tools to build IA-32 or Intel® 64 architecture applications, one of:
    - Microsoft Visual Studio 2013* Professional Edition or higher, with C++ component installed
    - Microsoft Visual Studio 2012* Professional Edition or higher, with C++ component installed
    - Microsoft Visual Studio 2010* Professional Edition or higher, with C++ component installed
    - Microsoft Visual Studio 2008* Standard Edition or higher, with C++ and “X64 Compiler and Tools” components installed [1]
    - Intel® Visual Fortran development environment based on Microsoft Visual Studio 2010 Shell (included with some license types of Intel® Fortran Compiler) [2]
    - Intel® Visual Fortran development environment based on Microsoft Visual Studio 2008 Shell (included with compiler versions 11.0, 11.1 and Intel® Visual Fortran Composer XE 2011 through Update 5.)
- To use command-line tools only to build IA-32 architecture applications, one of:
    - Microsoft Visual Studio Express 2013 for Windows Desktop*
    - Microsoft Visual Studio Express 2012 for Windows Desktop*
    - Microsoft Visual C++ 2010* Express Edition [3]
    - Microsoft Visual C++ 2008* Express Edition
- To use command-line tools only to build Intel® 64 architecture applications, one of:
    - Microsoft Visual Studio Express 2013 for Windows Desktop*
    - Microsoft Visual Studio Express 2012 for Windows Desktop*

    - Microsoft Windows Software Development Kit for Windows 8.1*
    - Microsoft Windows Software Development Kit for Windows 8*
- Installation of the included Microsoft Visual Studio 2010 Shell has the following limitations:
    - Windows XP 64-bit is not supported. Microsoft Visual Studio 2008 Shell from earlier versions of Intel® Visual Fortran can be used on Windows XP 64-bit.
- Installation on Windows XP requires prior installation of Microsoft .NET 4.0* Framework. See the Installation section of the Intel® Visual Fortran Composer XE 2013 SP1 Release Notes for details.
- To read the on-disk documentation, Adobe Reader* 7.0 or later

Notes:

1. Microsoft Visual Studio 2008 Standard Edition installs the "x64 Compiler and Tools" component by default – the Professional and higher editions require a "Custom" install to select this. Microsoft Visual Studio 2010 installs this component by default in all editions.
2. Intel® Visual Fortran development environment based on Microsoft Visual Studio 2010* Shell is included with Academic and Commercial licenses for Intel® Visual Fortran Composer XE. It is not included with Evaluation licenses. This development environment provides everything necessary to edit, build and debug Fortran applications. Some features of the full Visual Studio product are not included, such as:
    - Resource Editor (see ResEdit* (http://www.resedit.net/), a third-party tool, for a substitute)
    - Automated conversion of Compaq* Visual Fortran projects
    - Microsoft language tools such as Visual C++* or Visual Basic*
3. Microsoft Visual C++ 2010 Express Edition will coexist with the Intel® Visual Fortran Composer XE 2013 installation of Microsoft Visual Studio 2010 Shell. Note that the C++ and Fortran development environments will be separate.
4. The default for Intel® Visual Fortran is to build IA-32 architecture applications that require a processor supporting the Intel® SSE2 instructions. A compiler option is available to generate code that will run on any IA-32 architecture processor. Note, however, that applications calling Intel® MKL require a processor supporting the Intel® SSE2 instructions.
5. Applications can be run on the same Windows versions as specified above for development. Applications may also run on non-embedded 32-bit versions of Microsoft Windows earlier than Windows XP, though Intel does not test these for compatibility. Your application may depend on a Windows API routine not present in older versions of Windows. You are responsible for testing application compatibility. You may need to copy certain run-time DLLs onto the target system to run your application.

1.4.1 Windows XP* Support Deprecated

A future major version of Intel® Visual Fortran Composer XE will remove support for installing on Microsoft Windows XP.

1.4.2 Support for Microsoft Visual Studio 2008* Deprecated

In a future major release of Intel® Visual Fortran Composer, support for use with Microsoft Visual Studio 2008 will be removed. Intel recommends that customers migrate to Microsoft Visual Studio 2013* or higher at their earliest convenience.

1.5 Documentation

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

1.5.1 Documentation on Creating Windows-based Applications Now on the Web

The chapter in the compiler documentation on using QuickWin, dialogs and the Windows API has been moved to the "Software Documentation from Intel" web site: See Using Intel® Visual Fortran to Create and Build Windows-based Applications (PDF)

1.5.2 Delay on First Access of Intel® Composer XE Help in Visual Studio 2008*

The first time you access the Intel-installed help documentation in Microsoft Visual Studio 2008, a substantial delay may occur. Prior to displaying the help, Visual Studio must merge the new help into the collection and re-index the collection. Depending on whether you have Visual Studio help content already present and the size of the installed help, the delay in viewing help the first time may be several minutes or more.

1.5.3 Documentation Viewing Issue with Microsoft Internet Explorer* 10 and Windows Server* 2012

If on Windows Server 2012 you find that you cannot display help or documentation from within Internet Explorer 10, modifying a security setting for Microsoft Internet Explorer usually corrects the problem. From Tools > Internet Options > Security, add "about:internet" to the list of trusted sites. Optionally, you can remove "about:internet" from the list of trusted sites after you finish viewing the documentation.

1.6 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

1.7 Samples

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

1.8 Japanese Language Support

Intel® compilers provide support for Japanese language users when the combined Japanese-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 is not provided in all product updates.

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 Windows.

1.9 Technical Support

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

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

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

2 Installation

2.1 Pre-Installation Steps

2.1.1 Install Prerequisite Software

If you will be installing the included Microsoft Visual Studio 2010 Shell, additional Microsoft software may be required to be installed before beginning the installation of Intel® Visual Fortran Composer XE 2013 SP1. Note that installation of Microsoft Visual Studio 2010 Shell is not supported on Windows XP 64-bit.

Microsoft .NET 4.0 Framework* is required. If you do not already have this installed, you can download the installer:

- .NET 4.0 Framework 32-bit and 64-bit

If you are installing on Windows 8.1*, Windows 8*, Windows 7* or Windows Server 2008, the installation of the Shell will attempt to download and install .NET Framework 4.0 automatically if it is not already present. If this fails, the Intel® Visual Fortran Composer install will fail with a message that may not indicate the exact problem. If you find that the installation of the Shell fails, please download .NET 4.0 Framework from the above link and try again.

If you are installing Intel® Visual Fortran Composer XE 2013 SP1 from DVD or from the full product downloadable that includes Visual Studio 2010 Shell, it will try to install Visual Studio 2010 Shell if you do not already have Visual Studio 2010 installed. If you do not want Visual Studio 2010 Shell to install, you can choose a Custom install and deselect it, or choose the "_novsshell.exe" downloadable installer.

2.1.2 Configure Visual Studio for 64-bit Applications

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

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

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

This step is not required when using Microsoft Visual Studio 2010, Visual Studio 2012 or Visual Studio 2013.

2.1 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 Windows* 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.

2.2 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.

2.3 Installation

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

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

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

You do not need to uninstall previous versions or updates before installing a newer version – the new version will coexist with the older versions. If you want to remove older versions, you may do so before or after installing the newer one.

Register your serial number at the Intel® Software Development Products Registration Center for access to product updates and previous versions.

2.3.1 Reboot After Install Recommended

Installation of Intel® Visual Fortran Composer XE adds to the system PATH environment variable the folders containing the compiler run-time DLLs (but not those of Intel® Math Kernel Library). On some systems, if the length of the PATH value is between 2048 and 4096 characters, command line operations may fail until the system is rebooted. Intel recommends a reboot after the first install of Intel® Visual Fortran Composer XE.

2.3.2 Cluster Installation

If Microsoft Compute Cluster Pack* is present, and the installation detects that the installing system is a member of a cluster, the product will be installed on all visible nodes of the cluster when a “Full” installation is requested. If a “Custom” installation is requested, you will be given the option to install on the current node only.

2.3.3 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.

2.3.4 Additional Steps to Install Documentation for Microsoft Visual Studio 2010

When installing Intel® Visual Fortran Composer XE 2013 SP1 on a system with Microsoft Visual Studio 2010 for the first time, you will be asked to initialize the “Local Store” for documentation for Visual Studio 2010 if it was not done before. The "Help Library Manager" will register the Intel® Visual Fortran Composer XE 2013 SP1 help documentation within Visual Studio 2010. Please follow the instructions of the "Help Library Manager" installation wizard to install the Intel® Visual Fortran Composer XE 2013 SP1 help documentation for Visual Studio 2010.

This step is only needed once. When you install Intel® Visual Fortran Composer XE 2013 SP1 updates in the future, you will not be required to re-register the documentation through the "Help Library Manager".

For the more information, see http://msdn.microsoft.com/en-us/library/dd264831.aspx or search on microsoft.com for "Help Library Manager".

2.4 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.

2.5 Changing, Updating and Removing the Product

Use the Windows Control Panel "Add or Remove Products" or "Programs and Features" applet to change which product components are installed or to remove the product. Depending on which product you installed, the entry will be one of the following:

- Intel(R) Visual Fortran Composer XE 2013 SP1 for Windows*
- Intel(R) Composer XE 2013 SP1 for Windows*
- Intel(R) Parallel Studio XE 2013 SP1 for Windows*

If you also installed Microsoft Visual Studio 2010 Shell as part of the compiler install, the following additional entries may be present:

- Microsoft Visual Studio 2010 Shell (Integrated) - ENU
- Microsoft Visual Studio 2010 Files for Intel Visual Fortran
- Microsoft Visual Studio 2010 Remote Debugger – ENU

The entries for Visual Studio Shell, Files and Remote Debugger should not be removed unless you want to completely remove the product.

When installing an updated version of the product, you do not need to remove the older version first. The first time you install an update, you will have the choice to replace the older version or to keep both the older and newer versions on the system. This choice is remembered for future updates. In Microsoft Visual Studio you can select which specific compiler version to use through the Tools > Options > Intel Composer XE > Visual Fortran Compiler dialog. Compiler versions older than 12.0 (Intel® Visual Fortran Composer XE 2011) are not available to be selected through Visual Studio. All installed versions can be used from the command line.

If you remove a newer version of the product you may have to reinstall the integrations into Microsoft Visual Studio from the older version.

2.6 Silent Install and Uninstall

For information on how to install and uninstall the compiler in an automated fashion, please see Intel® Compilers for Windows* Silent Installation Guide.

2.7 Installation Folders

The installation folder arrangement is shown in the diagram below. Not all folders will be present in a given installation. The system environment variable `IFORT_COMPILER14` can be used to locate the most recently installed Intel® Visual Fortran Composer XE 2013 SP1.

- `C:\Program Files\Intel\Composer XE 2013 SP1`
  - `bin`
    - `ia32`
    - `ia32_intel64`
    - `intel64`
    - `intel64_mic`
    - `sourcechecker`
  - `compiler`
    - `include`
      - `ia32`
      - `intel64`
      - `mic`
    - `lib`
      - `ia32`
      - `intel64`
      - `mic`
  - `debugger`
  - `Documentation`
  - `Help`
  - `mkl`
    - `benchmarks`
    - `bin`
    - `examples`
    - `include`
    - `interfaces`
    - `lib`
    - `tests`
    - `tools`
  - `redist`
  - `Samples`
  - `setup_x_xxx`

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

- `ia32`: Files used to build applications that run on IA-32

- `intel64`: Files used to build applications that run on Intel® 64
- `ia32_intel64`: Compilers that run on IA-32 to build applications that run on Intel®64

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

By default, updates of a given version will replace the existing directory contents. When the first update is installed, the user is given the option of having the new update installed alongside the previous installation, keeping both on the system. If this is done, the top-level folder name for the older update is changed to `Composer XE 2013 SP1.`*`nnn`* where *`nnn`* is the update number.

3 Intel® Visual Fortran Compiler

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

3.1 Compatibility

In general, object code and modules compiled with earlier versions of Intel Fortran Compiler (8.0 and later) may be used in a build with version 14.0. 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 (`/Qipo`) option must be recompiled.
- 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 declaration 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 (12.0)

In previous releases, 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, compiler versions 12.0 and later align such items at 16 bytes and expect received arguments to be aligned on 16-byte boundaries.

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 13 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.

3.1.2 Static Form of the Intel® OpenMP* Library is No Longer Provided

The static form of the Intel® OpenMP* library, libiomp5mt.lib, is no longer provided, and the `/Qopenmp-link:static` command line option is no longer supported. Please replace all references to libiomp5mt.lib with libiomp5md.lib, the DLL import library. This change also implies that applications using OpenMP will need to have the Intel® compiler redistributables installed if deployed on a system where an Intel® compiler is not also present. See Redistributable Libraries for Intel® Visual Fortran Composer XE for more information.

3.1.3 Fortran Expression Evaluator

Fortran Expression Evaluator (FEE) is a plug-in for Microsoft Visual Studio* that is installed with Intel® Visual Fortran Compiler. It extends the standard debugger in Microsoft Visual Studio* IDE by handling Fortran expressions. There is no other change in usability.

3.2 New and Changed Compiler Features

Some language features may not yet be described in the compiler documentation. Please refer to the Fortran 2003 Standard (PDF)) and Fortran 2008 Standard (PDF) if necessary.

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 keywords supported earlier (DECLARE TARGET MIRROR, DECLARE TARGET LINKABLE, MAPTO, MAPFROM, SCRATCH) are 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:

- 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_DYNMIC_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 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

3.2.4 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.

3.2.5 Coarrays (13.0)

No special procedure is necessary to run a program that uses coarrays in a shared-memory environment; 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.

A license for Intel® Cluster Studio must be present in order to use the `/coarray:distributed` option. For details on how to run a distributed coarray application on Windows, please see Building and Running a Distributed Coarray Application on Windows.

Use of coarray applications with any MPI implementation other than Intel® MPI, or with OpenMP*, is not supported at this time.

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 `/Qcoarray-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`.

3.2.6 ATTRIBUTES ALIGN for component of derived type (13.0.1)

As of compiler version 13.0.1, the ATTRIBUTES ALIGN directive may be specified for an ALLOCATABLE or POINTER component of a derived type. The directive must be placed within the derived type declaration, and if it is an extended type, the directive must not name a component in a parent type.

If this is specified, the compiler will apply the indicated alignment when the component is allocated, either through an explicit ALLOCATE or, for ALLOCATABLE components, through implicit allocation according to Fortran language rules.

A module containing an ATTRIBUTES ALIGN directive for a derived type component cannot be used with a compiler earlier than version 13.0.1.

3.2.7 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. 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.

3.3.1 New and Changed in Composer XE 2013 SP1

- /assume:std_value (14.0.1)
- /Q[a]xMIC-AVX512 (14.0.1)
- /Qfma
- /Qimf-domain-exclusion
- /Qmic
- /Qoffload
- /Qoffload-attribute-target
- /Qoffload-option
- /Qopenmp-offload
- /Qopenmp-simd
- /Qopt-assume-safe-padding
- /Qopt-gather-scatter-unroll=n (14.0.1)
- /Qopt-prefetch-distance
- /Qopt-streaming-cache-evict
- /Qopt-threads-per-core
- /Qvecabi
- /QxATOM_SSE4.2
- /switch:fe_debug_use_inherit (14.0.2)

- /wrap-margin

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

3.3.1.1 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.

3.3.1.2 New /Q[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.

3.3.1.3 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.

3.3.1.4 New /switch:fe_debug_use_inherit Internal Command Line Switch (14.0.2)

Examining the parent fields of an extended derived type in the Microsoft Visual Studio* 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 the Microsoft Visual Studio debugger. Please note however, if you do set the fe_debug_use_inherit switch, you are unable to use the latter form within the debugger.

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 Visual Studio Integration Changes

3.4.1 DLL Libraries Default in New Projects

New Fortran projects, created after Intel® Visual Fortran Composer XE 2013 SP1 has been installed, have the project properties set so that the DLL form of the run-time libraries is used. This is consistent with Microsoft Visual C++, but is a change from previous versions of Intel® Visual Fortran. If you wish to use the static libraries, you can change the project property Fortran > Libraries > Use Runtime Library. Note that the OpenMP* library, libiomp5md.dll, is provided in DLL form only and will be used no matter which setting you select, should your application use OpenMP.

3.4.2 Parallel Build Option (13.1)

An enhancement to the Visual Studio build environment has been added which allows for parallel builds of sources without unresolved dependencies on multicore or multiprocessor systems. This can reduce the total time needed to build larger projects.

To enable this, open the project property page Fortran > General and set the property “Multi-processor compilation” to Yes.

3.5 Known Issues

3.5.1 Command-Line Diagnostic Issue for Filenames with Japanese Characters

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

3.5.2 Debugging might fail when only Microsoft Visual Studio 2012 is installed

On Microsoft Windows* systems with only Microsoft Visual Studio 2012* installed debugging of Fortran applications might fail. Some symptoms might be failing watches (expression evaluations) or conditional breakpoints.

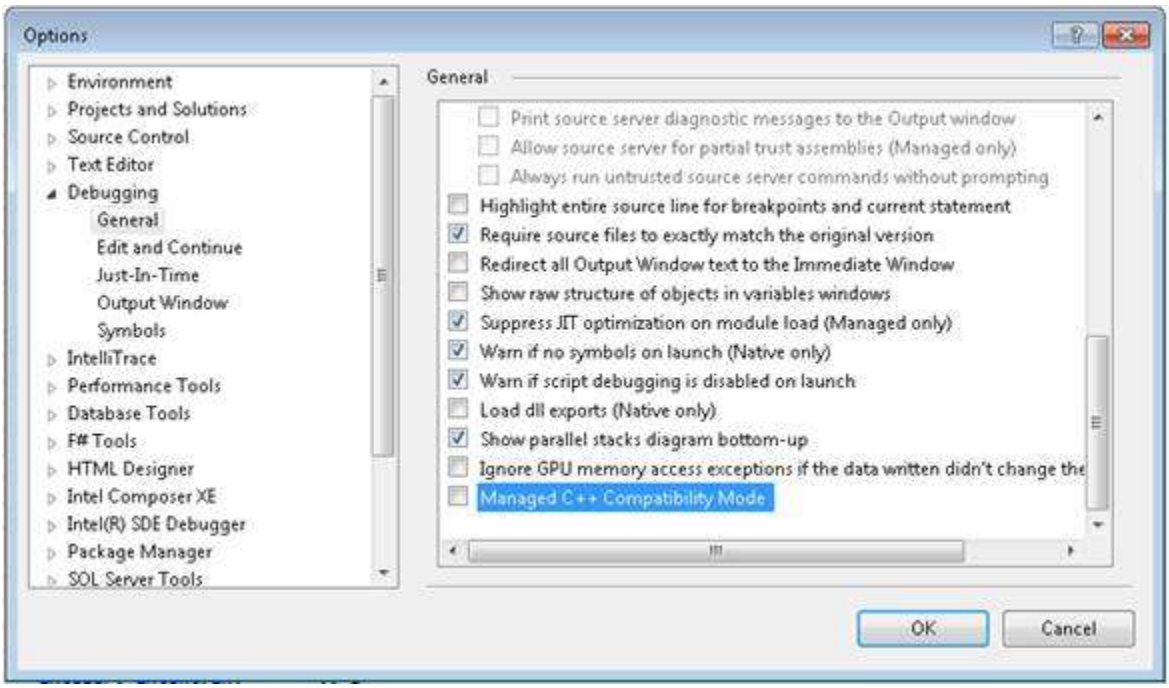
Intel(R) Visual Fortran Composer XE 2013 (SP1) provides a debugger extension called Fortran Expression Evaluator (FEE) to enable debugging of Fortran applications. For some FEE functionality the Microsoft Visual Studio 2010* libraries are required.

One solution is to install Microsoft Visual Studio 2010* in addition to Microsoft Visual Studio 2012*. An alternative is to install the Microsoft Visual C++ 2010 SP1 Redistributable Package (x86) found here: http://www.microsoft.com/en-us/download/details.aspx?id=8328

3.5.3 Debugging mixed language programs with Fortran does not work

To enable debugging mixed language programs with Fortran Expression Evaluator (FEE), unset the following configuration:

Menu Tools ->Options, under section Debugging->General, clear the Managed C++ Compatibility Mode check box



3.6 Microsoft Visual Studio 2010 and 2012 Notes

Microsoft Visual Studio 2010 brings several changes that primarily affect building of mixed-language applications where the main program is in C or C++. These changes were carried forward into Visual Studio 2012.

3.6.1 Configuring Microsoft Visual C++ to Reference Intel® Fortran Run-Time Libraries

In previous releases, one used the Tools > Options > Projects and Solutions > VC++ Directories dialog to make the Intel Fortran LIB folder available to C/C++ projects. In Visual Studio 2010, the method of doing this is very different.

1. In Visual Studio, with a solution open that contains a C++ project, select View > Property Manager. If you do not see Property Manager under the View menu, you will find it under View > Additional Windows. The Property Manager window will appear. Note that this is not Properties Window or Properties Pages.
2. Click on the triangles or + signs to expand the property tree under the Debug|Win32 configuration
3. Double click on Microsoft.Cpp.Win32.user
4. Select VC++ Directories
5. Click in the field to the right of "Library Directories"
6. Click the triangle that appears to the right and select <Edit...>
7. Click the New Line button or press Ctrl-Insert
8. In the new field that appears, type:

   ```
   $(IFORT_COMPILER14)\compiler\lib\ia32
   ```
9. Click OK, OK
10. In the Visual Studio toolbar, select File > Save All

If you will be building Intel® 64 (x64) configurations:

1. Back in the Property Manager, expand the Debug|x64 configuration
2. Double click on Microsoft.Cpp.x64.user
3. Select VC++ Directories
4. Click in the field to the right of "Library Directories"
5. Click the triangle that appears to the right and select <Edit...>
6. Click the New Line button or press Ctrl-Insert
7. In the new field that appears, type:

   ```
   $(IFORT_COMPILER14)\compiler\lib\intel64
   ```
8. Click OK, OK
9. In the Visual Studio toolbar, select File > Save All

Click on the Solution Explorer tab, or press Ctrl-Alt-L, to make it visible again.

If you do not see the Microsoft.Cpp.x64.user property page listed for the x64 configuration, right click on Debug|x64 and select Add Existing property Sheet. Browse to the location which contains the MsBuild 4.0 property pages. On Windows XP, this is typically:

```
C:\Documents and Settings\<username>\Local Settings\Application Data\Microsoft\MSBuild\v4.0
```

On Windows 7 and Windows 8, it is typically:

```
C:\Users\<username>\AppData\Local\Microsoft\MSBuild\v4.0
```

You may need to enable viewing of hidden files and folders to see these paths.

Select Microsoft.Cpp.x64.user.props and click Open. Now follow the steps above.

3.6.2 Adjusting Project Dependencies

If you are converting a project from an earlier version of Visual Studio and had established Project Dependencies, these are converted to References by Visual Studio 2010/2012. A Fortran project that is referenced by a C/C++ project will prevent the C/C++ project from building, with an MSB4075 error. To solve this:

1. Right click on the C/C++ project and select References.
2. If any Fortran project is shown as a reference, click Remove Reference. Repeat this for all Fortran projects shown as a reference. Click OK.
3. Repeat the above steps for any other C/C++ project

Now you have to reestablish project dependencies.

1. Right click on the C/C++ project and select Project Dependencies.
2. Check the box for each project that is a dependent of this project.
3. Click OK.
4. Repeat the above steps for any other C/C++ project that has dependencies.

Unlike earlier versions of Visual Studio, Visual Studio 2010/2012 does not automatically link in the output library of dependent projects, so you will need to add those libraries explicitly to the parent project under Linker > Additional Dependencies. You can use the Visual Studio macros $(ConfigurationName) and $(PlatformName) as required to qualify the path. For example:

```
..\FLIB\$(ConfigurationName)\FLIB.lib
```

Where $(ConfigurationName) will expand to Release or Debug, as appropriate. Similarly, $(PlatformName) will expand to Win32 or x64 as appropriate.

3.6.3 Showing Documentation Issue with Visual Studio 2012 and Windows Server 2012

If on Windows Server 2012* you find that you cannot display help or documentation from within Visual Studio 2012, correcting a security setting for Microsoft Internet Explorer* usually corrects the problem. From Tools > Internet Options > Security, change the settings for Internet Zone to allow “MIME Sniffing” and “Active Scripting”.

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
- FINAL subroutines
- User-defined derived type I/O
- 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
- Default initialization of polymorphic 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 Developing Applications that use Intel® Xeon Phi™ Coprocessors

This section summarizes changes, new features and late-breaking news about developing for Intel® Xeon Phi™ coprocessors using Intel® Visual Fortran Composer XE 2013 SP1 for Windows*

4.1 Introduction

Intel® Visual Fortran Composer XE 2013 SP1 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.

4.2 Documentation

For the latest documentation updates, please see Intel® Composer XE 2013 Documentation Updates for Intel® MIC Architecture.

4.3 Changes and Known Issues

This section corrects or adds to the product documentation.

4.3.1 Fortran code built for Intel® MIC Architecture on Windows with initial 14.0 compiler release must be recompiled if relinked with 14.0 Update 1 libraries

A fix has been introduced in version 14.0 Update 1 of the Intel® Visual Fortran Compiler which will allow users to build for one Intel® MIC Architecture platform (Linux* or Windows*) and deploy on the other platform. As a result of this fix, all Fortran code built with the initial release of the 14.0 compiler for Intel® MIC Architecture on Windows must be recompiled if the objects are either relinked with the Update 1 (or later) libraries or are using dynamic libraries in order to avoid segmentation faults while performing IO.

4.3.2 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.

4.3.3 *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.

4.3.4 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 `<common_files>\Intel\Shared Libraries\redist\lib\mic` where `<common_files>` is usually `C:\Program Files (x86)\Common Files`.

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 (`<common_files>\Intel\Shared Libraries\redist\intel64_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.

4.4 Intel® Debugger Extension for Intel® Many Integrated Core Architecture (Intel® MIC Architecture)

This section summarizes new features and changes, usage and known issues related to the Intel® Debugger Extension. This debugger extension only supports code targeting Intel® Many Integrated Core Architecture (Intel® MIC Architecture).

4.4.1 Features

- Support for both native coprocessor applications and host applications with offload extensions
- Debug multiple coprocessor cards at the same time (with offload extension)

4.4.2 Using the Intel® Debugger Extension

The Intel® Debugger Extension is a plug-in for Microsoft Visual Studio* IDE. It transparently enables debugging of projects defined by that IDE. Applications for Intel® Xeon Phi™ can be either loaded and executed or attached to.

Instructions on how to use Intel® Debugger Extension can be found in the documentation

4.4.3 Documentation

The full documentation for the Intel® Debugger Extension can be found here:

```
<install-dir>\Documentation\en_US\debugger\mic\ ↵
              vsmigdb_config_guide.pdf
```

4.4.4 Known Issues

- Offload debugging is only supported in Microsoft Visual Studio 2012* and Microsoft Visual Studio 2013*.
- Data breakpoints are not yet supported within offload sections.
- Disassembly window cannot be scrolled outside of 1024 bytes from the starting address within an offload section.
- Handling of exceptions from the Intel® MIC Architecture application is not supported.
- Changing breakpoints while the application is running does not work. The changes will appear to be in effect but they are not applied.
- Starting an Intel® MIC Architecture native application is not supported. You can attach to a currently running application, though.
- Under certain conditions, the Thread Window does not show all threads running on the coprocessor.
- The Thread Window in Microsoft Visual Studio* offers context menu actions to Freeze, Thaw and Rename threads. These context menu actions are not functional when the thread is on a coprocessor.

- Setting a breakpoint right before an offload section sets a breakpoint at the first statement of the offload section. This only is true if there is no statement for the host between set breakpoint and offload section. This is normal Microsoft Visual Studio* breakpoint behavior but might become more visible with interweaved code from host and coprocessor. The superfluous breakpoint for the offload section can be manually disabled (or removed) if desired.
- Under certain conditions, debugging into offloaded code can stop the execution. The source position seems to be the offload directive but the execution is actually inside a library routine. You should be able to continue execution from that point.
- Only Intel® 64 applications containing offload sections can be debugged with the Intel® Debugger Extension for Intel® Many Integrated Core Architecture.
- Stepping out of an offload section does not step back into the host code. It rather continues execution without stopping (unless another event occurs). This is intended behavior.
- The functionality “Set Next Statement” is not working within an offload section.
- If breakpoints have been set for an offload section in a project already, starting the debugger might show bound breakpoints without addresses. Those do not have an impact on functionality.
- For offload sections, setting breakpoints by address or within the Disassembly window won’t work.
- For offload sections, using breakpoints with the following conditions of hit counts do not work: “break when the hit count is equal to” and “break when the hit count is a multiple of”.
- The following options in the Disassembly window do not work within offload sections: “Show Line Numbers”, “Show Symbol Names” and “Show Source Code”
- Evaluating variables declared outside the offload section shows wrong values.
- When starting a multi-threaded application that uses a high amount of threads, updates due to events (stepping, hitting a breakpoint, adding new thread, …) might be slow. This only happens until the first 200 threads are available. A current workaround is to close the Thread window until the first 200 threads are created. With more than 200 threads and the Thread window open this performance problem does not occur.

5 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).

5.1 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

5.2 What's New in Intel® MKL 11.1 Update 1

- Introduced support for Intel® AVX-512 instructions set with limited set of optimizations
- 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.

5.3 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
- Intel MKL now supports compiler assisted offload and Automatic offload programming model on Intel Xeon Phi™ coprocessors based on the Intel® Many Integrated Core Architecture (Intel® MIC Architecture) on Windows* OS*
- 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
- Link Tool and Link Line advisor: Added support for Intel MIC Architecture on Windows* OS

5.4 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

5.5 Known Issues

A full list of the known limitations can be found in the Intel® MKL Article List at Intel® Developer Zone

5.6 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/)

6 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.

The Visual Intel® Fortran Compiler and Intel® Math Kernel Library are provided under Intel Corporation’s End User License Agreement (EULA).

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