

# Intel® Visual Fortran Composer XE 2011 for Windows\* Installation Guide and Release Notes

---

Document number: 321417-003US  
23 November 2011

## Table of Contents

1	Introduction .....	4
1.1	Change History .....	4
1.2	Product Contents .....	6
1.3	System Requirements.....	6
1.3.1	IA-64 Architecture (Intel® Itanium®) Development Not Supported.....	8
1.3.2	Support Deprecated for Microsoft Visual Studio 2005* .....	8
1.3.3	Support Deprecated for Microsoft Windows Vista* and Microsoft Windows Server 2003* .....	8
1.4	Documentation.....	8
1.4.1	Documentation on Creating Windows-based Applications Now on the Web .....	8
1.5	Samples.....	9
1.6	Japanese Language Support.....	9
1.7	Technical Support.....	9
2	Installation.....	9
2.1	Pre-Installation Steps.....	9
2.1.1	Install Prerequisite Software .....	9
2.1.2	Configure Visual Studio for 64-bit Applications.....	10
2.1.3	Installation on Microsoft Windows Vista* and Windows 7* .....	10
2.2	Installation .....	11
2.2.1	Cluster Installation .....	11
2.2.2	Activation of Purchase after Evaluation Using the Intel Activation Tool .....	11
2.2.3	Using a License Server.....	11
2.2.4	Installing the IMSL* Fortran Numerical Library*.....	11
2.2.5	Additional Steps to Install Documentation for Microsoft Visual Studio 2010 .....	12

2.2.6	Prompt for Administrator Permission with Microsoft Visual Studio 2005* .....	12
2.3	Changing, Updating and Removing the Product .....	12
2.4	Silent Install and Uninstall .....	13
2.5	Installation Folders.....	13
2.6	Installation Known Issues.....	14
2.6.1	Documentation Issue with Multiple Visual Studio Versions.....	14
3	Intel® Visual Fortran Compiler .....	15
3.1	Compatibility .....	15
3.1.1	Stack Alignment Change for REAL(16) and COMPLEX(16) Datatypes (12.0).....	15
3.2	New and Changed Compiler Features .....	16
3.2.1	Features from Fortran 2003 .....	16
3.2.2	Features from Fortran 2008 .....	16
3.2.3	Coarrays .....	17
3.2.4	New and Changed Directives (Update 6) .....	17
3.2.5	OpenMP Changes (Update 6).....	18
3.2.6	New QuickWin Library Routines (Update 6) .....	18
3.2.7	Other Changes .....	19
3.3	New and Changed Compiler Options.....	19
3.3.1	New and Changed in Composer XE 2011 Update 6 .....	19
3.3.2	New and Changed in Composer XE 2011.....	19
3.3.3	Additional Keywords for /QSOX option, default changed (Update 3) .....	20
3.4	Experimental Parallel Build Option (Update 8) .....	21
3.5	Source Editor Enhancements for Users of Visual Studio 2010 (Update 6) .....	21
3.5.1	Navigation Bar .....	21
3.5.2	Smart Indenting .....	21
3.5.3	Code Outlining .....	21
3.5.4	Block Delimiter Matching .....	22
3.5.5	Code Snippets .....	22
3.5.6	Go To Definition and Find All References .....	22
3.5.7	Find Symbol.....	22
3.5.8	Object Browser .....	23
3.5.9	Quick Info for Intrinsic Procedures .....	23
3.5.10	Task List Comments .....	23

3.6	Other Changes .....	23
3.6.1	Build Environment Command Script Change (12.0) .....	23
3.6.2	Static Security Analysis Feature (formerly Source Checker) Requires Intel® Inspector XE (12.0) .....	24
3.6.3	OpenMP* Legacy Libraries Removed .....	25
3.6.4	OpenMP* Libraries Default to Dynamic Linking.....	25
3.6.5	RANF Portability Function Is Now an Intrinsic.....	25
3.6.6	Support Deprecated for Intel® Parallel Debugger Extension.....	26
3.7	Known Issues .....	26
3.7.1	Fortran 2003 Features Not Yet Implemented .....	26
3.7.2	Coarray Issues.....	26
3.7.3	Command-Line Diagnostic Issue for Filenames with Japanese Characters .....	26
3.7.4	Allocatable Arrays and OpenMP* PRIVATE or FIRSTPRIVATE .....	26
3.8	Microsoft Visual Studio 2010* Notes .....	26
3.8.1	Configuring Microsoft Visual C++ to Reference Intel® Fortran Run-Time Libraries 26	
3.8.2	Adjusting Project Dependencies .....	28
3.9	Fortran 2003 and Fortran 2008 Feature Summary .....	28
4	Intel® Math Kernel Library .....	32
4.1	What's New in Intel® MKL 10.3 Update 8 .....	32
4.2	What's New in Intel® MKL 10.3 Update 7 .....	32
4.3	What's New in Intel® MKL 10.3 Update 6 .....	33
4.4	What's New in Intel® MKL 10.3 Update 5 .....	33
4.5	What's New in Intel® MKL 10.3 Update 4 .....	34
4.6	What's New in Intel® MKL 10.3 Update 3 .....	34
4.7	What's New in Intel® MKL 10.3 Update 2 .....	35
4.8	What's New in Intel® MKL 10.3 Update 1 .....	35
4.9	What's New in Intel® MKL 10.3.....	36
4.10	Known Issues .....	37
4.11	Notices.....	37
4.12	Attributions.....	38
5	Intel® Parallel Debugger Extension .....	38
5.1	New Features .....	38
5.2	Known Issues .....	38

5.3	Documentation.....	40
6	Disclaimer and Legal Information.....	40

## 1 Introduction

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

Intel® Visual Fortran Composer XE 2011 is the next release of the product formerly called Intel® Visual Fortran Compiler Professional Edition.

### 1.1 Change History

This section highlights important changes in product updates.

#### Update 8

- Intel® Visual Fortran Compiler updated to [12.1.2](#)
- Documentation provided for [QuickWin library routines GETLINEWIDTHQQ and SETLINEWIDTHQQ](#), added in Update 6
- Experimental [parallel build support added](#)
- Intel® Math Kernel Library updated to [10.3 Update 8](#)
- Corrections to reported problems

#### Update 7

- Intel® Visual Fortran Compiler updated to [12.1.1](#)
- Intel® Math Kernel Library updated to [10.3 Update 7](#)
  - Intel® MKL [Notices](#) updated to include more deprecations
- Microsoft .NET 4.0 Framework\* is now required for installation of Visual Studio 2010 Shell\*
- Corrections to reported problems

#### Update 6

- Full product includes integrated development environment based on Microsoft Visual Studio 2010\* Shell. Visual Studio 2008\* Shell, as installed by earlier versions of Intel® Visual Fortran, is still supported.
- The product installs into a [new top-level folder](#)
- Intel® Visual Fortran Compiler updated to [12.1.0](#)
  - The coarray feature is now supported for distributed memory environments on Windows
  - ALLOCATE with a polymorphic SOURCE= is now supported
  - [Many new editor features](#) have been added for users of Visual Studio 2010

- OpenMP\* 3.1 is now supported
- For this release, the core compiler documentation known as the User and Reference Guides has been reorganized and streamlined. Among the most noticeable changes are: a new Key Features section highlighting important Intel compiler functionality and the organization of the Compiler Option reference section into functional groups.
- Documentation on Creating and Building Windows-based applications has been [moved to Intel's web site](#)
- [Support for the Intel® Parallel Debugger Extension is deprecated](#)
- Intel® Math Kernel Library updated to [10.3 Update 6](#)
- Corrections to reported problems

#### Update 5

- [Support for Microsoft Visual Studio 2005\\* is deprecated](#)
- [Support for Microsoft Windows Vista\\* and Microsoft Windows Server 2003\\* is deprecated](#)
- Intel® Visual Fortran Compiler updated to [12.0.5](#)
- Intel® Math Kernel Library updated to [10.3 Update 5](#)
- Corrections to reported problems

#### Update 4

- Intel® Visual Fortran Compiler updated to [12.0.4](#)
- Intel® Math Kernel Library updated to [10.3 Update 4](#)
- Corrections to reported problems

#### Update 3

- Intel® Visual Fortran Compiler updated to [12.0.3](#)
  - The /QSOX option [now accepts optional keywords specifying information to include and the default behavior has changed](#)
  - Japanese localized documentation and diagnostic messages now available
- Intel® Math Kernel Library updated to [10.3 Update 3](#)
- Corrections to reported problems

#### Update 2

- Intel® Visual Fortran Compiler updated to [12.0.2](#)
- Intel® Math Kernel Library updated to [10.3 Update 2](#)
- The way that the [Static Security Analysis feature stores data files has changed](#)
- Corrections to reported problems

#### Update 1

- Intel® Visual Fortran Compiler updated to [12.0.1](#)

- Intel® Math Kernel Library updated to [10.3 Update 1](#)
- Corrections to reported problems

## Intel® Visual Fortran Composer XE 2011 Product Release

- Initial product release

### 1.2 Product Contents

*Intel® Visual Fortran Composer XE 2011 SP1 for Windows\** includes the following components:

- Intel® Visual Fortran Compiler XE 12.1.2 for building applications that run on IA-32 and Intel® 64 architecture systems
- Intel® Math Kernel Library 10.3 Update 8
- Integration into Microsoft\* development environments
- Intel® Parallel Debugger Extension 12.1.2 for Microsoft Visual Studio\*
- Microsoft Visual Studio 2010 Shell and Libraries (not included with Student or Evaluation licenses)
- Sample programs
- On-disk documentation

*Intel® Visual Fortran Composer XE 2011 SP1 with IMSL\* 6.0 for Windows\** includes the above plus the IMSL\* Fortran Numerical Library\* 6.0 from Rogue Wave Software\*

### 1.3 System Requirements

For an explanation of architecture names, see <http://intel.ly/mXlIjK>

- 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 Vista\*, Microsoft Windows 7\*, Microsoft Windows Server 2003\*, 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 2010\* or Visual Studio 2010\* Shell or Visual Studio 2008\* SP1 or Visual Studio 2008 Shell with Visual Studio 2008 SP1 update applied. Other versions of Visual Studio listed below are not supported on Windows Server 2008 or Windows HPC Server 2008.
- 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 2010\* with C++ and “X64 Compiler and Tools” components installed [\[1\]](#)

- Microsoft Visual Studio 2008\* Standard Edition or higher with C++ and “X64 Compiler and Tools” components installed [1]
- Microsoft Visual Studio 2005\* 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 12.0.)
- To use command-line tools only to build IA-32 architecture applications, one of:
  - Microsoft Visual C++ 2010\* Express Edition [2]
  - Microsoft Visual C++ 2008\* Express Edition
  - Microsoft Visual C++ 2005\* Express Edition and Microsoft Windows SDK for Windows 2008 and .NET Framework 3.5\*
- To use command-line tools only to build Intel® 64 architecture applications, one of:
  - Microsoft Windows Software Development Kit Update for Windows Vista\*
  - Microsoft Windows SDK for Windows 2008 and .NET Framework 3.5\*
- 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.
  - Installation requires prior installation of Microsoft .NET 4.0\* Framework. Installation on Windows XP and Windows Server 2003 requires prior installation of Microsoft Windows Imaging Component. See the [Installation section](#) of the Intel Visual Fortran Composer XE 2011 SP1 Release Notes for details.
- To read the on-disk documentation, Adobe Reader\* 7.0 or later

Notes:

1. Microsoft Visual Studio 2005 and 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 or Student 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 2011 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.
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 Win32 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.

### **1.3.1 IA-64 Architecture (Intel® Itanium®) Development Not Supported**

This product version does not support development on or for IA-64 architecture (Intel® Itanium®) systems. The version 11.1 compiler remains available for development of IA-64 architecture applications.

### **1.3.2 Support Deprecated for Microsoft Visual Studio 2005\***

In a future major release of the product, Intel® Visual Fortran Composer XE will remove support for Microsoft Visual Studio 2005\*. Intel recommends migrating to Visual Studio 2010 at your earliest convenience.

### **1.3.3 Support Deprecated for Microsoft Windows Vista\* and Microsoft Windows Server 2003\***

In a future major release of the product, Intel® Visual Fortran Composer XE will remove support for installation and use on Microsoft Windows Vista\* or Microsoft Windows Server 2003\*.

## **1.4 Documentation**

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

### **1.4.1 Documentation on Creating Windows-based Applications Now on the Web**

The chapter in the compiler documentation on “Using Intel® Visual Fortran to Create and Build Windows-based Applications” has been moved to the “Software Documentation from Intel” web site. You can download this documentation from <http://intel.ly/WinApp> (PDF).

## **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.

## 1.5 Samples

Samples for each product component can be found in the `Samples` folder as shown under [Installation Folders](#).

## 1.6 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 <http://intel.ly/pla2A5>

## 1.7 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 2011. 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 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 on Windows XP or Windows Server 2003, Microsoft Windows Imaging Component is required. If you do not already have this installed, you can download the installer:

- [Windows Imaging Component 32-bit](#)
- [Windows Imaging Component 64-bit](#)

If you are installing Intel® Visual Fortran Composer XE 2011 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 2005\* or 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 2005/2008 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 2005” (or 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.

### **2.1.3 Installation on Microsoft Windows Vista\* and Windows 7\***

Microsoft Visual Studio 2005\* users should install *Visual Studio 2005 Service Pack 1* (VS 2005 SP1) as well as the Visual Studio 2005 Service Pack 1 Update for Windows Vista, which is linked to from the VS 2005 SP1 page. After installing these updates, you must ensure that Visual Studio runs with Administrator permissions, otherwise you will be unable to use the Intel compiler. For more information, please see Microsoft's Visual Studio on Windows Vista page (<http://msdn2.microsoft.com/en-us/vstudio/aa948853.aspx>) and related documents. See also [Prompt for Administrator Permission with Microsoft Visual Studio 2005\\*](#).

## 2.2 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, 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.

### 2.2.1 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.

### 2.2.2 Activation of Purchase after Evaluation Using the Intel Activation Tool

Note for evaluation customers: Intel Activation Tool “ActivationTool.exe” is included in this product release and installed at [Common Files]\Intel\Parallel Studio XE 2011 SP1\Activation\.

If you installed the product using an Evaluation license or serial number or using the “Evaluate this product (no serial number required)” option during installation, and then purchased the product, you can activate your purchase using the Intel Activation Tool at Start > All Programs > Intel Parallel Studio XE 2011 > Product Activation. It will convert your evaluation software to a fully licensed product.

### 2.2.3 Using a License Server

If you have purchased a “floating” license, see <http://intel.ly/oPEdEe> for information on how to install using a license file or license server. 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.

### 2.2.4 Installing the IMSL\* Fortran Numerical Library\*

If you have purchased Intel Visual Fortran Composer XE 2011 with IMSL\* 6.0, the IMSL installation is separate from the compiler installation: either a separate download or a separate disc. You must install the compiler before installing IMSL.

### 2.2.5 Additional Steps to Install Documentation for Microsoft Visual Studio 2010

When installing Intel Visual Fortran Composer XE 2011 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 2011 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 2011 help documentation for Visual Studio 2010.

This step is only needed once. When you install Intel Visual Fortran Composer XE 2011 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”.

### 2.2.6 Prompt for Administrator Permission with Microsoft Visual Studio 2005\*

If you are installing on Microsoft Windows Vista\* or later versions of Microsoft Windows and are using Microsoft Visual Studio 2005, Windows may display a dialog similar to the following:



If this is displayed, it is important that you click the Continue button and leave the “Always show this message” box checked. If you select “Exit Visual Studio” instead, or do nothing (this message times out after two minutes), the compiler integration will not install completely.

For more information, see [Installation on Microsoft Windows Vista\\* and Windows 7\\*](#).

## 2.3 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 2011 for Windows\*

- Intel(R) Composer XE 2011 for Windows\*
- Intel(R) Parallel Studio XE 2011 SP1 for Windows\*

If you installed the IMSL\* libraries, an additional entry will be present:

- IMSL\* Fortran Library 6.0 for Intel(R) Visual Fortran Composer XE 2011

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

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

The entries for Visual Studio Shell and Files 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(R) Visual Fortran Compiler dialog. Compiler versions older than 11.1 are not available to be selected through Visual Studio 2005 or 2008 and compiler versions older than 12.0 are not available to be selected through Visual Studio 2010. 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.

## 2.4 Silent Install and Uninstall

For information on how to install and uninstall the compiler in an automated fashion, please see <http://intel.ly/qAwdvR>

## 2.5 Installation Folders

The installation folder arrangement is shown in the diagram below. Not all folders will be present in a given installation. Note that the top-level folder changed as of Update 6. The system environment variable `IFORT_COMPILER12` can be used to locate the most recently installed Intel Visual Fortran Composer XE 2011.

- C:\Program Files\Intel\Composer XE 2011 SP1
  - o bin
    - ia32
    - ia32\_intel64
    - intel64
  - o compiler
    - include
      - ia32

- intel64
  - lib
    - ia32
    - intel64
- debugger
- Documentation
- help
- mkl
  - benchmarks
  - bin
  - examples
  - include
  - interfaces
  - lib
  - tests
  - tools
- redistrib
- Samples

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 2011 SP1.nnn` where `nnn` is the update number.

## 2.6 Installation Known Issues

### 2.6.1 Documentation Issue with Multiple Visual Studio Versions

If you have both Microsoft Visual Studio\* 2005 and 2008 installed on your system and integrate Intel® Visual Fortran Composer XE 2011 into both versions, removing the integration from one of the versions will remove the integrated Intel® Visual Fortran Composer XE 2011 documentation from both.

To re-install the documentation:

1. Use the Control Panel to select the product.
  - For Windows XP\* users: Select **Control Panel > Add/Remove Programs**.
  - For Windows 7\* users: Select **Control Panel > Programs and Features**.
  - For Windows Vista\* users: Select **Control Panel > Programs**.
2. With the product selected, click the **Change/Remove** button and choose Modify mode.
3. In the **Select Components** dialog box, unselect “Integrated Documentation;” this will **remove** the documentation.
4. Repeat steps 1 and 2.
5. In the **Select Components** dialog box, select “Integrated Documentation” to **install** documentation again

### 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 12.0. 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 and were compiled with versions earlier than 11.0 must be recompiled. The compiler will notify you if this issue is encountered.

##### 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, the version 12 compiler aligns such items at 16 bytes and expects 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 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.

## 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 ([http://j3-fortran.org/doc/2003\\_Committee\\_Draft/04-007.pdf](http://j3-fortran.org/doc/2003_Committee_Draft/04-007.pdf)) and Fortran 2008 Standard (<http://j3-fortran.org/doc/standing/links/007.pdf>) if necessary.

### 3.2.1 Features from Fortran 2003

- FINAL subroutines
- GENERIC keyword for type-bound procedures
- A generic interface may have the same name as a derived type
- Bounds specification and bounds remapping list on a pointer assignment
- ALLOCATE with a polymorphic SOURCE (Update 6)

### 3.2.2 Features from Fortran 2008

- 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 IMAGE\_INDEX, LCOBOUND, NUM\_IMAGES, THIS\_IMAGE, UCOBOUND
  - **Note:** ATOMIC\_DEFINE and ATOMIC\_REF are not supported in this version
- 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

- (Update 6) 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 intrinsic function NULL, is considered not present
- (Update 6) 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.

### 3.2.3 Coarrays

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.

As of Update 6, support is added for coarray applications running using distributed memory across a Windows cluster. A license for Intel® Cluster Toolkit 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 <http://intel.ly/oZurZS>

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`.

#### 3.2.3.1 Coarray Known Issues

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

- Output (WRITE, PRINT, etc.) of an array slice of a coarray referencing another image. A whole array reference, or a single element works.
- Default initialization of a REAL(16) or COMPLEX(16) coarray

### 3.2.4 New and Changed Directives (Update 6)

The following compiler directives are new or changed in Intel® Composer XE 2011 Update 6 – please see the documentation for details:

- ATTRIBUTES VECTOR
- NOFUSION
- You can now specify a FIRSTPRIVATE clause in the PARALLEL directive
- You can now specify a FIRSTPRIVATE or LASTPRIVATE clause in the SIMD directive

### 3.2.5 OpenMP Changes (Update 6)

The following changes to OpenMP\* support are in Intel® Composer XE 2011 Update 6:

- OpenMP 3.1 is supported
- TASKYIELD directive
- New clauses have been added to the ATOMIC directive
- You can now specify FINAL and MERGEABLE clauses in the TASK directive

### 3.2.6 New QuickWin Library Routines (Update 6)

The following new QuickWin library routines were added in Intel® Visual Fortran Composer XE 2011 Update 6, but are not in the product documentation.

#### 3.2.6.1 GETLINEWIDTHQQ

**Graphics Function:** Gets the current line width or the line width set by the last call to SETLINEWIDTHQQ.

**Module:** USE IFQWIN

```
result = GETLINEWIDTHQQ()
```

#### Results

The result is of type INTEGER(4). It contains the current width of the line in pixels.

If there is no call to SETLINEWIDTHQQ in the program (or on that particular graphics window), it returns 1. (The default width of a line drawn by any graphics routine is 1 pixel.)

#### 3.2.6.2 SETLINEWIDTHQQ

**Graphics Subroutine:** Sets the width of a solid line drawn using any of the supported graphics functions.

**Module:** USE IFQWIN

```
CALL SETLINEWIDTHQQ (x)
```

x (Input) INTEGER(4). It can be any non-negative integer.

This subroutine sets the line width in pixels using the value that is passed as the argument.

SETLINEWIDTHQQ affects the drawing of straight lines using functions such as LINETO, POLYGON, LINETOAR, LINETOAREX, RECTANGLE, and it affects the drawing of curved lines using functions such as ARC, ELLIPSE, or PIE.

#### Note

The nWidth argument in the Windows\* API CreatePen() is the width used to draw the lines or borders of a closed shape. A cosmetic pen can only have a width of 1 pixel. If you specify a higher width, it is ignored. A geometric pen can have a width of 1 or more pixels, but the line can only be solid or null. This means that if you specify the style as PS\_DASH, PS\_DOT,

PS\_DASHDOT, or PS\_DASHDOTDOT, but set a width higher than 1 pixel, the line is drawn as PS\_SOLID.

### See Also

Windows\* API CreatePen in the Microsoft\* MSDN documentation.

### 3.2.7 Other Changes

- The ability to create a source listing file with identifier cross-reference has been added
- Guided auto-parallelism
- An option to use math library functions that are faster but return results with less precision or accuracy
- An option to use math library functions that return consistent results across different models and manufacturers of processors
- An option to simplify specifying that Fortran 2003 semantics should be assumed for all syntax where the compiler default is not consistent with Fortran 2003
- The ability to generate a build dependencies output file has been added
- In Visual Studio projects, the module output directory of a subproject is automatically added as an “additional INCLUDE directory” for its parent project

## 3.3 New and Changed Compiler Options

Please refer to the compiler documentation for details.

### 3.3.1 New and Changed in Composer XE 2011 Update 6

- /align [no]qcommons
- /arch:CORE-AVX-I
- /arch:CORE-AVX2
- /openmp
- /QaxCORE-AVX-I
- /QaxCORE-AVX2
- /Qfma[-]
- /Qopt-mem-layout-trans[:n]
- /QxCORE-AVX-I
- /QxCORE-AVX2
- /QxSSSE3\_ATOM

### 3.3.2 New and Changed in Composer XE 2011

- /assume:[no]fpe\_summary
- /assume:old\_ldout\_format
- /gen-dep
- /gen-depformat
- /list
- /list-line-len
- /list-page-len
- /Qcoarray ([see above](#))

- /Qcoarray-num-images
- /Qcov-dir
- /Qcov-file
- /Qcov-gen
- /Qdiag-sc-dir
- /Qguide
- /Qguide-data-trans
- /Qguide-file
- /Qguide-file-append
- /Qguide-opts
- /Qguide-par
- /Qguide-vec
- /Qimf-absolute-error
- /Qimf-accuracy-bits
- /Qimf-arch-consistency
- /Qimf-max-error
- /Qimf-precision
- /Qopt-args-in-regs
- /Qopt-matmul
- /Qpar-runtime-control
- /Qpatchable-addresses
- /Qprof-value-profiling
- /Qprofile-functions
- /Qprofile-loops
- /Qprofile-loops-report
- /Qsimd
- /Qsox=keyword
- /Qzero-initialized-in-bss
- /show
- /standard-semantics

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

### 3.3.3 Additional Keywords for /Qsox option, default changed (Update 3)

The /Qsox option, which adds information to the object file about compiler options used and procedure profiling information, has been enhanced to let the user request that the list of inlined functions be included and to let the user exclude information about procedure profiling.

The syntax for /Qsox is now:

```
/Qsox[-]
/Qsox=keyword[, keyword]
```

Where *keyword* is one of *inline* or *profile*. If `/Qsox` is specified with no keywords, only the command line options are included – this is a change from previous releases. To maintain the previous behavior, use `/Qsox=profile`. Multiple `/Qsox` options may be specified on the command line – if so, they are interpreted in left-to-right order.

The information is added to the object file as comment directives. These are ignored by Microsoft linkers beginning with Visual Studio 2005, therefore the information will not be present in the executable.

### 3.4 Experimental Parallel Build Option (Update 8)

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 > Command Line. In the Additional Options field, add `/MP` as the first (or only) option. If `/MP` is not the first option on the line, the parallel build will not be done.

This is an initial implementation – the method of requesting it and the order of files built is likely to change in future versions.

### 3.5 Source Editor Enhancements for Users of Visual Studio 2010 (Update 6)

Intel® Visual Fortran Composer XE 2011 Update 6 introduces new features for the Fortran source editor in Microsoft Visual Studio 2010. Users of earlier versions of Visual Studio will not see these changes. Some of the new features are disabled by default – they can be enabled or disabled through the new Visual Studio 2010 dialog Tools > Options > Text Editor > Fortran > Advanced. The property Disable Database in this dialog defaults to False – that is, the database is enabled. If this property is set to True, disabling the database, then some of the features below will also be disabled.

#### 3.5.1 Navigation Bar

The Navigation Bar is displayed above the source editor pane. It has two sections: the left side selects among modules defined in the current source file, if any, and the right side lists all procedures defined in the current source and allows you to jump to an individual procedure by clicking on the entry.

#### 3.5.2 Smart Indenting

The editor will automatically indent block constructs (IF, DO, etc.) and outdent the corresponding end statement. You can adjust this behavior using Tools > Options > Text Editor > Fortran > Tabs

#### 3.5.3 Code Outlining

Options: Enable Outlining and Outline Statement Blocks

Default: True for Enable Outlining, False for Outline Statement Blocks

If only Enable Outlining is set to True, you can collapse whole program units by clicking on the “-” symbol next to the PROGRAM, SUBROUTINE, FUNCTION, MODULE or BLOCK DATA statement in the editor. The contents of the program unit will be collapsed and replaced by a “...” symbol, with the button changing to a “+”. Click on the “+” to expand.

If Outline Statement Blocks is also set to True, you can also collapse block constructs such as IF, DO, etc.

#### **3.5.4 Block Delimiter Matching**

Option: Highlight Matching Tokens

Default: False

If Highlight Matching Tokens is set to True, clicking on the beginning or end keyword of a block construct (IF/THEN/ELSE/END IF, DO/END DO, SELECT/END SELECT, etc.) will highlight the matching keyword for that level.

This option will also highlight other references to identifiers when you click on an identifier.

#### **3.5.5 Code Snippets**

Right click anywhere in a source file and select Insert Snippet... You can then select from a list of prototype constructs such as DO WHILE, MODULE, etc. Double-click on the entry to select it. For those constructs that include a name, a temporary name will be inserted and selected. Replace the name by typing over it and the name on the corresponding END statement will be replaced to match.

Snippets also have a “shortname”. You can insert a snippet by typing its shortname and pressing the TAB key.

#### **3.5.6 Go To Definition and Find All References**

Options: Collect Object Browser Information, Enable Find All References, Enable Go To Definition, Scan System Includes

Default: False

If all of the options listed above are set to True, source files in your project and system include files will be scanned for declared identifiers. You will then be able to right click on an identifier and select either Go To Definition (F12 is defined by default as a shortcut for this) or Find All References. This feature is also sometimes referred to as “Source Browser”.

Go To Definition will reposition the cursor to the statement where the identifier was declared, opening the declaring source file if needed. For identifiers declared in system modules such as IFWINTY, the source for that module will be opened.

Find All References will open a list of all references to the selected identifier across the project. Double-click on one to go to find that reference.

#### **3.5.7 Find Symbol**

Options: Same as Go To Definition

Default: False

The Edit > Find and Replace > Find Symbol command (accessed by Alt-F12) brings up a dialog where you can search the project for a symbol or a substring of a symbol.

### 3.5.8 Object Browser

Options: Same as Go To Definition

Default: False

If the options are set to True, View > Object Browser opens the object browser that displays the procedures in your project in a hierarchical tree.

### 3.5.9 Quick Info for Intrinsic Procedures

Options: Enable Intrinsic Quick Info, Enable Intrinsic Parameter Info

Default: False

If these options are set to True, and you hover the mouse over the name of an intrinsic procedure, a short description of that intrinsic will appear in a tooltip. If you type the name of an intrinsic procedure and an open parenthesis, information about the procedure and its arguments will appear.

### 3.5.10 Task List Comments

Option: Enumerate Comment Tasks

Default: True

You can add specially formatted comments that will be collected for Visual Studio's Task List. A task comment begins with the "!" comment introducer (other comment introducers are not supported) followed by one of the "tokens" specified in Tools > Options > Environment > Task List. Whitespace may precede or follow the comment introducer. For example:

```
! TODO Make sure this gets filled in
```

is recognized as a task comment. Visual Studio predefines several tokens such as HACK, TODO and UNDONE – you can modify this list as desired. You can also specify that a token indicates a high, normal or low priority item.

To view the list of tasks for your project, use View > Other Windows > Task List and select "Comments" from the drop-down box. Double-click on any entry to go to that source line.

## 3.6 Other Changes

### 3.6.1 Build Environment Command Script Change (12.0)

The command window script used to establish the build environment allows the optional specification of the version of Microsoft Visual Studio to use. If you are not using the predefined Start menu shortcut to open a build environment window, use the following command to establish the proper environment:

```
"<install-dir>\bin\compilervars.bat" arch [vs]
```

Where *arch* is one of *ia32*, *ia32\_intel64* or *intel64* as appropriate for the target architecture you want to build for. *vs* is optional and can be one of *vs2010*, *vs2008* or *vs2005*. If *vs* is not specified, the version of Visual Studio specified at installation time for command-line integration is used by default.

**Note:** If the version of Visual Studio installed is Visual Studio 2008 Shell, you can specify *vs* as *vs2008shell* or omit it.

If you also have Intel® C++ Composer XE 2011 SP1 installed, this command will also establish the environment for using that compiler.

The script file names *iclvars.bat* and *ifortvars.bat* have been retained for compatibility with previous releases.

### 3.6.2 Static Security Analysis Feature (formerly Source Checker) Requires Intel® Inspector XE (12.0)

The “Source Checker” feature, from compiler version 11.1, has been enhanced and renamed “Static Security Analysis”. The compiler options to enable Static Security Analysis remain the same as in compiler version 11.1 (for example, */Qdiag-enable:sc*), but the results are now written to a file that is interpreted by Intel® Inspector XE rather than being included in compiler diagnostics output.

#### 3.6.2.1 Change in Static Security Analysis Behavior (Update 2)

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

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

The versions of *inspxe-runsc* included in Composer XE 2011 and Composer XE 2011 Update 1 generate all the SSA results in a single directory. In the multiple link case this violated the rule that all the SSA results for one and only one project must be created in the same directory. The updated version of *inspxe-runsc* respects this rule by generating results for each link step in a separate directory. The name of that directory is formed from the name of the file being linked. Thus if a build specification describes a project that builds two executables, *file1.exe* and *file2.exe*, then earlier versions of *inspxe-runsc* would create two results, one for *file1* and one for *file2*, say *r000sc* and *r001sc*, in the same directory. The new version of *inspxe-runsc* will also create two results, but the one for *file1* will be created in “My

Inspector XE results – file1\r000sc” and the one for file2 will be created in “My Inspector XE results – file2\r000sc”. The directories containing the results are both created in the same parent directory.

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

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

### 3.6.3 OpenMP\* Legacy Libraries Removed

The OpenMP “legacy” libraries were removed in version 12.0. Only the “compatibility” libraries are provided.

### 3.6.4 OpenMP\* Libraries Default to Dynamic Linking

As of version 11.0, OpenMP applications link to the dynamic OpenMP libraries by default. To specify static linking of the OpenMP libraries, specify `/Qopenmp-link:static` . The static libraries are deprecated and may be removed in a future major release.

### 3.6.5 RANF Portability Function Is Now an Intrinsic

The RANF function in the portability library is a non-standard random number generator. As of the version 12.0 compiler, RANF is an intrinsic function with a new, higher-performance implementation. If your program has added `USE IFPORT` to provide access to RANF, no changes will be seen and you will get the older version. If your program does not have `USE IFPORT`, or you add `INTRINSIC RANF`, you will get the new version that returns a different sequence, for a given seed, than the older version. The portability subroutine `SRAND` is still used to set the seed for RANF. Intel recommends use of the standard intrinsic `RANDOM_NUMBER`, but RANF is provided for compatibility with applications already using it.

### 3.6.6 Support Deprecated for Intel® Parallel Debugger Extension

In a future major release of the product, the Intel® Parallel Debugger Extension may be removed. This would remove the ability to use in the debugger:

- The OpenMP\* windows
- Data sharing detection (Intel® Inspector XE offers data sharing detection)
- The vector register window (the regular Visual Studio register window can show vector registers)

## 3.7 Known Issues

### 3.7.1 Fortran 2003 Features Not Yet Implemented

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

- User-defined derived type I/O
- Parameterized derived types
- Default initialization of CLASS objects
- The keyword MODULE may be omitted in MODULE PROCEDURE
- Transformational intrinsics, such as MERGE, in initialization expressions

### 3.7.2 Coarray Issues

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

### 3.7.3 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.

### 3.7.4 Allocatable Arrays and OpenMP\* PRIVATE or FIRSTPRIVATE

The compiler may fail to properly initialize allocatable arrays named in OpenMP PRIVATE or FIRSTPRIVATE clauses. This issue will be corrected in a future update. If you encounter problems with this combination of features, try adding the option `/switch:omp3_private`. This is a temporary workaround and should not be used on a permanent basis. The Intel issue ID for this problem is DPD200160978.

## 3.8 Microsoft Visual Studio 2010\* 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++.

### 3.8.1 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_COMPILER12)\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_COMPILER12)\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\\Local Settings\Application Data\Microsoft\MSBuild\v4.0
```

On Windows Vista and Windows 7, it is typically:

```
C:\Users\\Local Settings\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.

### 3.8.2 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. 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 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.

## 3.9 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
- 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
- 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

Fortran 2003 features not yet supported include:

- User-defined derived type I/O
- Parameterized derived types
- Empty structure constructors
- Use of certain transformational intrinsics in initialization expressions
- Omitting MODULE in MODULE PROCEDURE

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 IMAGE\_INDEX, LCOBOUND, NUM\_IMAGES, THIS\_IMAGE, UCOBOUND
  - **Note:** ATOMIC\_DEFINE and ATOMIC\_REF are not supported in this version
- 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 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). For information on bug fixes, see <http://intel.ly/neQlw2>

### 4.1 What's New in Intel® MKL 10.3 Update 8

- Data Fitting component: Added a set of new data fitting functions covering one-dimensional algorithms for vector spline construction, cell or bin search, and evaluation, differentiation, and integration of the spline interpolants. Includes support for:
  - Linear, quadratic, cubic, step-wise const, and user-defined splines
  - Cell search with configuration parameters for optimal performance
  - User-defined interpolation and extrapolation
  - Vector-valued functions
  - Column- and row-major storage formats
- Sparse BLAS: Improved compressed sparse row matrix-vector multiply (?CSRMV) performance for very sparse matrices on high core counts supporting Intel Advanced Vector Extensions (AVX)
- FFTs: Improved the performance of the 1D double precision FFTs on systems supporting Intel AVX
- Statistics functions: Improved the performance and scalability for computing the Variance-Covariance and Correlation matrices (FAST method) on Intel® Core processors
- Added a Microsoft Visual Studio\* project tool for building custom DLLs from static libraries
- Bug fixes

### 4.2 What's New in Intel® MKL 10.3 Update 7

- BLAS: Improved DSYRK/SSYRK threaded performance for small output matrices and large outer products (i.e., rectangular input matrices), on all recent Intel® Xeon® processors
- BLAS: Improved ?GEMM performance for small problems (<10) where beta =1 on all recent Intel Xeon processors
- BLAS: Improved DSCAL performance for small problems and for cases where INCX=1 on 32-bit programs running on Intel Xeon processors 5500, 5600, and 7500 series

- BLAS-like extensions: Improved threading and cache utilization of in-place transposition of square matrices
- PARDISO: Introduced an independent threading control for PARDISO; use `MKL_DOMAIN_PARDISO` with the `mkl_domain_set_num_threads()` function
- Poisson Library: Added support for 2D and 3D periodic boundary conditions
- Included the Link Line Advisor in the documentation directory
- Added a command line link tool for use with scripting tools such as `libtool`
- Added C header files with `stdcall` prototypes for functions in the following components: BLAS, Sparse BLAS, LAPACK, PARDISO/DSS, RCI Iterative Solvers, Vector Mathematical Functions, Vector Statistical Functions, and the support functions
- Changed the names of constants used to specify the domain in the `mkl_domain_set_num_threads()` function (e.g., `MKL_BLAS` has become `MKL_DOMAIN_BLAS`); the old names still exist with the exception of `MKL_PARDISO`
- Bug fixes

#### 4.3 What's New in Intel® MKL 10.3 Update 6

- Sparse BLAS: Added a new option to the `mkl_?csrbsr` converter function allowing detection and removal of zero elements when converting from the BSR format to the CSR format
- Changed DLL loading behavior on Windows\*: Intel® MKL DLLs can no longer be in separate directories on the `PATH`, but must all be in the same directory - either with the executable or in another directory specified in the `PATH` environment variable
- Bug fixes

#### 4.4 What's New in Intel® MKL 10.3 Update 5

- BLAS: Improved performance: `{S,C,Z}TRSM` for processors with Intel® Advanced Vector Extensions (Intel® AVX); `{S,D}GEM2VU` for processors with Intel AVX as well as the Intel® Core™ i7 processor and the Intel® Xeon® processor 5500 series
- BLAS: Improved scaling: `?TRMV` for large matrices on all architectures; `DGEMM` for odd numbers of threads on Intel® Xeon® processor 5400 series
- LAPACK: Included LAPACK 3.3.1 extensions and the respective LAPACKE interfaces
- LAPACK: Improved the performance of `?SYGST` and `?HEGST` used in generalized eigenvalue problems
- LAPACK: Improved the performance of the inverse of an LU factored matrix (`?GETRI`)
- PARDISO: Added transpose and conjugate transpose solve capability (`ATx=b` and `AHx=b`); facilitates compressed sparse column (CSC) format support
- PARDISO: Improve out-of-core PARDISO performance when the memory requirements slightly exceed available memory using `MKL_PARDISO_OOC_MAX_SWAP_SIZE` environment variable and in-core PARDISO
- Optimization Solvers: Added Inf and NaN checks in the RCI Trust-Region solvers
- FFTs: Improved the performance of 3D FFTs on small cubes from `2x2x2` to `10x10x10` for all supported precisions and types on all Intel® processors supporting Intel® SSE3 and later

- FFT examples: Re-designed example programs to cover common use cases for Intel MKL DFTI and FFTW
- VSL: Improved the performance of the single precision MT19937 and MT2203 basic random number generators on the Intel® Core™ i7-2600 processor on 64-bit operating systems
- VSL: Improved the performance of the integer version of the SOBOL quasi-random number generator on the Intel® Core™ i7-2600 processor and Intel® Xeon® processor 5400 series

#### 4.5 What's New in Intel® MKL 10.3 Update 4

- BLAS: Improved DTRMM performance on Intel® Xeon® processors 5400 and later
- BLAS: Improved DTRSM performance on all 64-bit enabled processors, especially processors with Intel® Advanced Vector Extensions (Intel® AVX)
- LAPACK: Incorporated bug fixes from the LAPACK 3.3.1 release
- OOC PARDISO: Improved the estimate of the amount of memory needed in out-of-core operation
- FFT: Improved 1D real FFT scaling through improved threading
- FFT: Updated C and Fortran FFT examples to use the new single dynamic library linking model
- VML: Improved performance of the single precision Enhanced Performance version of the real Hypot and complex Abs functions and of the complex Arg, Div, Mul, MulByConj functions for all accuracy modes on Intel® Xeon® processors 5600 and 7500 series, and the Intel® Core™ i7-2600 processor
- Service functions: Improvements and additions to the Intel MKL service functions the online release notes at <http://intel.ly/pkUQXI> for more information)
- Bug fixes

#### 4.6 What's New in Intel® MKL 10.3 Update 3

- BLAS: Improved multi-threaded performance of DSYRK, DTRSM, and DGEMM on Intel® Xeon® processor 5400 series running 32-bit Windows\*
- LAPACK: Implemented LAPACK 3.3 from netlib including Cosine-Sine decomposition, improved linear equations solvers for symmetric and Hermitian matrices and auxiliary functions
- PARDISO: 0-based permutation vectors are now allowed at input
- PARDISO: Documentation for the pardisoinit() routine
- PARDISO: Improved performance of serial PARDISO with multiple right-hand sides (RHS)
- PARDISO: Independent control for parallelism in the solve step for improved performance on small matrices—see description of iparm(25)
- PARDISO: Reduced backward substitution—allows partial solution computation for a full RHS—see description of iparm(31)
- FFT: Implemented Real FFT transforms for up to 3 to 7 dimensions

- FFT: Parallelized multi-dimensional complex transforms using split-complex data represented as two real arrays
- Cluster FFTs: Extended FORTRAN 90 interface to real-to-complex transforms and included new examples
- VML: Added new complex Pack/Unpack functions and real Gamma/LGamma functions
- VML: Improved performance on Intel® Xeon® processor 5600 series and processors supporting Intel® Advanced Vector Extensions (Intel® AVX) for the following: all functions when operating on short vectors (<100), all functions when operating on unaligned input vectors, the sPow2o3 function, and the enhanced performance (EP) version of complex Add and Sub.
- VSL: Functions for saving/restoring random number generator (RNG) streams to/from memory
- VSL: Added new UniformBits32 and UniformBits64 functions
- VSL: Extended the number of unique streams supported by MT2203 BRNG from 1024 to 6024
- Bug fixes

**Note:** The GMP Arithmetic Functions in Intel MKL will be removed in a future version of Intel MKL.

#### 4.7 What's New in Intel® MKL 10.3 Update 2

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

#### 4.8 What's New in Intel® MKL 10.3 Update 1

- PARDISO/DSS: Added true F90 overloaded API (see the Intel® MKL reference manual for more information)
- PARDISO: Improved the statistical reporting to be more reader friendly
- Sparse BLAS: Improved performance of ?BSRMM functions on the latest Intel® processors
- FFTs: Support for negative strides
- FFT examples: Added examples for split-complex FFTs in C and Fortran using both the DFTI and FFTW3 interfaces
- VML: Improved performance of real in-place Add/Sub/Mul/Sqr functions on systems supporting SSE2 and SSE3

- Poisson Library: Changed the default behavior of the Poisson library functions from sequential to threaded operation
- Bug fixes

## 4.9 What's New in Intel® MKL 10.3

- BLAS
  - New functions for computing 2 matrix-vector products at once: [D/S]GEM2VU, [Z/C]GEM2VC
  - New functions for computing mixed precision general matrix-vector products: [DZ/SC]GEMV
  - New function for computing the sum of two scaled vectors: \*AXPBY
  - Intel® AVX optimizations in key functions: SMP LINPACK, level 3 BLAS, DDOT, DAXPY
- LAPACK
  - New C interfaces for LAPACK supporting row-major ordering
  - Integrated Netlib LAPACK 3.2.2 including one new computational routine (\*GEQRF) and two new auxiliary routines (\*GEQR2P and \*LARFGP) and the earlier LAPACK 3.2.1 update
  - Intel® AVX optimizations in key functions: DGETRF, DPOTRF, DGEQRF
- PARDISO
  - Improved performance of factor and solve steps in multi-core environments
  - Introduced the ability to solve for sparse right-hand sides and perform partial solves—produces partial solution vector
  - Improved performance of the out-of-core (OOC) factorization step
  - Support for zero-based (C-style) array indexing
  - Zeros on the diagonal of the matrix are no longer required in sparse data structures for symmetric matrices
  - New ILP64 PARDISO interface allows the use of both LP64 and ILP64 versions when linked to the LP64 libraries
  - The memory required for storing files on the disk in OOC mode can now be estimated just after reordering
- Sparse BLAS
  - Format conversion functions now support all data types (single and double precision for real and complex data) and can return sorted or unsorted arrays
- FFTs
  - New MPI FFTW 3.3alpha1 wrappers cover new cluster functionality
  - Improved load-balancing of cluster FFTs provides improved performance
  - Intel AVX optimizations in all 1D/2D/3D FFTs
  - Improved performance of 2D and 3D mixed-radix FFTs for single and double precision data for all systems supporting the SSE4.2 instruction set
  - Support for split-complex data represented as two real arrays introduced for 2D/3D FFTs
  - Support for 1D complex-to-complex transforms of large prime lengths
  - Introduced Hybrid parallelism (MPI + OpenMP\*) on cluster 1D complex transforms and increased performance on vector lengths which are a multiple of the number of MPI processes
- VML

- A new function for computing  $(ax+b)/(cy+d)$  where a, b, c, and d are scalars, and x and y are real vectors: `v[s/d]LinearFrac()`
- Intel AVX optimizations for real functions
- A new mode for setting denormals to zero, overflow support for complex vectors, and for every VML function a new function with an additional parameter for setting the accuracy mode
- VSL
  - A set of new Summary Statistics functions was added covering basic statistics, covariance and correlation, pooled, group, partial, and robust covariance/correlation, quantiles and streaming quantiles, outliers detection algorithm, and missing values support
    - Performance optimized algorithms: MI algorithm for support of missing values, TBS algorithm for computation of robust covariance, BACON algorithm for detection of outliers, ZW algorithm for computation of quantiles (streaming data case), and 1PASS algorithm for computation of pooled covariance
  - Improved performance of SFMT19937 Basic Random Number Generator (BRNG)
  - Intel® AVX optimizations: MT19937 and MT2203 BRNGs
- Documentation: Product documentation is available in the Microsoft Help Viewer\* 1.x format that integrates with Microsoft Visual Studio\* 2010
- Added runtime dispatching dynamic libraries allowing link to a single interface library which loads dependent libraries dynamically at runtime depending on runtime CPU detection and/or library function calls
- A new directory structure has been established to simplify integration of Intel MKL with the Intel® Parallel Studio XE family of products and directories formerly designated as "em64t" are now designated by the "intel64" tag
- Intel® Itanium® architecture (IA-64) support is not included in this release. Intel® MKL 10.2 is the latest release for IA-64
- The sparse solver functionality has been fully integrated into the core Intel MKL libraries and the libraries with "solver" in the filename have been removed from the product

## 4.10 Known Issues

A full list of the known limitations of this release can be found in the Knowledge Base for the Intel® MKL at <http://intel.ly/ptEfAP>

## 4.11 Notices

The following change is planned for future versions of Intel MKL. Please contact [Technical Support](#) if you have concerns:

- Content in the libraries containing `solver` in the filenames will be moved to the core library in a future version of Intel MKL. These `solver` libraries will then be removed.
- The Intel MKL GNU Multiple Precision\* (GMP) function interfaces will be removed in a future library release.
- The timing function `mkl_set_cpu_frequency()` is deprecated. Please use `mkl_get_max_cpu_frequency()`, `mkl_get_clocks_frequency()`, and `mkl_get_cpu_frequency()` as described in the Intel® MKL Reference Manual.

## 4.12 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](http://www.intel.com/software/products/mkl)) in both the product documentation and website.

The original versions of the BLAS from which that part of Intel® MKL was derived can be obtained from <http://www.netlib.org/blas/index.html>.

The original versions of LAPACK from which that part of Intel® MKL was derived can be obtained from <http://www.netlib.org/lapack/index.html>. The authors of LAPACK are E. Anderson, Z. Bai, C. Bischof, S. Blackford, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, and D. Sorensen. Our FORTRAN 90/95 interfaces to LAPACK are similar to those in the LAPACK95 package at <http://www.netlib.org/lapack95/index.html>. All interfaces are provided for pure procedures.

The original versions of ScaLAPACK from which that part of Intel® MKL was derived can be obtained from <http://www.netlib.org/scalapack/index.html>. The authors of ScaLAPACK are L. S. Blackford, J. Choi, A. Cleary, E. D'Azevedo, J. Demmel, I. Dhillon, J. Dongarra, S. Hammarling, G. Henry, A. Petitet, K. Stanley, D. Walker, and R. C. Whaley.

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.

## 5 Intel® Parallel Debugger Extension

This section summarizes changes, new features and late-breaking news about this version of the Intel® Parallel Debugger Extension.

### 5.1 New Features

- The Data Sharing Event window now shows the state of the Data Sharing Detection in a local status bar.

### 5.2 Known Issues

- Coarray elements cannot be viewed.
- Fortran multi-dimension arrays are not displayed correctly and are not accepted in filter expressions.

- Fortran complex types are not displayed correctly.
- Filters cannot be set on Fortran arrays with custom array bounds.
- If you are using Microsoft Visual Studio 2005, there are six Intel-specific exceptions that must be enabled manually. Select `Debug > Exceptions`, expand the `Win32 Exceptions` tree, and enable items:

```

a1a01db0 Intel Parallel Debugger Extension Exception 0
a1a01db1 Intel Parallel Debugger Extension Exception 1
a1a01db2 Intel Parallel Debugger Extension Exception 2
a1a01db3 Intel Parallel Debugger Extension Exception 3
a1a01db4 Intel Parallel Debugger Extension Exception 4
a1a01db5 Intel Parallel Debugger Extension Exception 5

```

This needs to be done once per project.

- Disabling the Intel Debugging exceptions during a debug session may cause Visual Studio (up to Visual Studio 2008, SP1) to hang.
- Use of the Intel Parallel Debugger Extension requires that the OpenMP library be linked dynamically, which is the default. If you wish to use the Parallel Debugger Extension, do not use `/Qopenmp-link:static` to specify static linking of the OpenMP Library.
- Be sure to enable the parallel debug instrumentation (switch `/debug:parallel`) before you start parallel debugging:  

```
Project > [project name] Properties > Configuration Properties >
Fortran > Debugging > Enable Parallel Debug Checks: Yes
(/debug:parallel).
```

Otherwise, the debugger will not detect datasharing events nor break on re-entrant calls.
- If you are using Microsoft Visual Studio 2008 and debugging 64-bit applications, you must have Visual Studio 2008 Service Pack 1 installed.
  - You can debug 64-bit applications under Visual Studio 2005 and 2008 without Service Packs only if they are linked to the low memory area. If not linked to the low memory area, you will not see any events until the debuggee terminates. After termination, all events are displayed in the event window. In order to debug 64-bit applications properly, set the base address to `0x10000` in `Project > Properties > Linker > Advanced`.
- Function local or heap variables are displayed as "???" in the data sharing event window.
- The SSE Registers window does not work for 64-bit applications - the window shows "???"
- Filters on static local variables are not set correctly via context menu.

- Reentrant call detection stops in Disassembly view.
- The debugger extension windows remain empty when their placement is changed from "docked" to "floating". The workaround is to either keep them docked or to restart the debug session after the placement was changed.
- The debugger extension requires the application to be started from Visual Studio. It does not work when attaching to an existing process.
- Windows settings are restored to default (Hexadecimal) when the window is hidden or closed and reopened again.
- The Microsoft Visual Studio 2010 debugger may crash when debugging applications under the Experimental Hive if the Intel Parallel Debugger Extension is installed. There is currently no solution available. In case of a crash you need to uninstall the Parallel Debugger Extension. In Control Panel use Programs and Features (or Add or Remove Programs), select Intel Visual Fortran Composer XE 2011 <version> for Windows\* and then Change/Modify. In the Modify dialog, deselect Intel Parallel Debugger Extension.

### 5.3 Documentation

Intel Parallel Debugger Extension Documentation can be accessed via the Help menu of Microsoft Visual Studio or by pressing the function key F1 after activation of a Parallel Debugger Extension window. Help is also available by clicking the link "HTML version" inside `debugger-documentation.htm`.

## 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/](http://www.intel.com/products/processor_number/) for details.

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

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

Copyright © 2011 Intel Corporation. All Rights Reserved.