Intel® C++ Compiler Professional Edition 11.1 for Windows* Installation Guide and Release Notes

Document number: 321414-002US
27 April 2009

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.

1.1 Product Contents

*Intel® C++ Compiler Professional Edition 11.1 for Windows** includes the following components:

- Intel® C++ Compilers for building applications that run on IA-32, Intel® 64 or IA-64 architecture systems running the Windows* operating system
- Intel® Assembler for IA-64 Architecture applications
- Intel® Integrated Performance Primitives
- Intel® Math Kernel Library
- Intel® Threading Building Blocks
- Intel® Parallel Debugger Extension
- Integration into Microsoft* development environments
- Sample programs

- On-disk documentation

1.2 System Requirements

For an explanation of architecture names, see http://software.intel.com/en-us/articles/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), or based on an IA-64 architecture (Intel® Itanium®) processor
  - For the best experience, a multi-core or multi-processor system is recommended
- 1GB RAM (2GB recommended)
- 4GB free disk space for all product features and all architectures
- Microsoft Windows XP*, Microsoft Windows Vista*, 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 2008* SP1. 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 2008* Standard Edition (or higher edition) with C++ and "X64 Compiler and Tools" components installed [1]
  - Microsoft Visual Studio 2005* Standard Edition (or higher edition) with C++ and "X64 Compiler and Tools" components installed [1]
- To use the Microsoft Visual Studio development environment or command-line tools to build IA-32 architecture applications, one of:
  - Microsoft Visual Studio .NET 2003* with C++ component installed [2]
  - Microsoft Visual C++ .NET 2003* [2]
- To use the Microsoft Visual Studio development environment or command-line tools to build IA-64 architecture applications, one of:
  - Microsoft Visual Studio 2008* Team System Edition with C++ and "Itanium Compiler and Tools" components installed [3] plus Microsoft Windows SDK for Windows 2008 and .NET Framework 3.5*
  - Microsoft Visual Studio 2005* Team System Edition with C++ and "Itanium Compiler and Tools" components installed [3]
- To use command-line tools only to build IA-32 architecture applications, one of:
  - 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*
- To use command-line tools only to build IA-64 architecture applications:

    - Microsoft Windows SDK for Windows 2008 and .NET Framework 3.5*
- 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.
2. Microsoft Visual Studio .NET 2003 is not supported on Microsoft Windows Vista. Support for Microsoft Visual Studio .NET 2003 will be removed in a future version of the product.
3. Microsoft Visual Studio is not supported for installation on IA-64 architecture systems
4. Development on an IA-64 architecture system supports building IA-64 architecture applications only.
5. The default for Intel® Parallel Composer 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.
6. 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 Documentation

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

1.4 Samples

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

1.5 Technical Support

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

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

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

2 Installation

2.1 Pre-installation Steps

2.1.1 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 or IA-64 architectures) 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, 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.

To use Microsoft Visual Studio 2005/2008 Team System Edition to build applications to run on IA-64 architecture systems, follow the above steps and ensure that the box “Itanium Compiler and Tools” is checked.

2.1.2 Installation on Microsoft Windows Vista*

On Microsoft Windows Vista*, Microsoft Visual Studio.NET 2003* is not supported. 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.

When installing on Microsoft Windows Vista and with Microsoft Visual Studio 2005, you may see one or more warning boxes saying that there are compatibility issues with Visual Studio 2005. In some cases, these warning boxes may be hidden behind the installer window making it appear that the installation has stalled. Look in the Windows task bar for additional windows that require acknowledgement before proceeding. You may safely allow Visual Studio 2005 to run as part of the compiler install process – after installation is complete, be sure to install the two Service Pack 1 updates as described in the paragraph above.

2.2 Installation

If you are installing the product for the first time, please be sure to have the product serial number available as you will be asked for it during installation. A valid license is required for installation and use.

To begin installation, 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

2.3 Changing, Updating and Removing the Product

Use the Windows Control Panel "Add or Remove Products" applet to change which product components are installed or to remove the product.

When installing an updated version of the product, you do not need to remove the older version first. You can have multiple versions of the compiler installed and select among them. 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 Installation Folders

The 11.1 product installs into a different arrangement of folders than in previous versions. The new arrangement is shown in the diagram below. Not all folders will be present in a given installation.

- `C:\Program Files\Intel\Compiler\11.1\`*`xxx`*
    - `bin`
        - `ia32`
        - `ia32_intel64`
        - `ia32_ia64`
        - `intel64`
        - `ia64`
    - `Documentation`
    - `help`
    - `include`
    - `ipp`
        - `em64t`
        - `ia32`
        - `ia64`
    - `lib`
        - `ia32`
        - `intel64`
        - `ia64`
    - `mkl`

        - `benchmarks`
        - `em64t`
        - `examples`
        - `ia32`
        - `ia64`
        - `include`
        - `interfaces`
        - `tests`
        - `tools`
    - `perf_headers`
    - `Samples`
    - `tbb`
        - `em64t`
        - `examples`
        - `ia32`
        - `ia64`
        - `include`
    - `setup_c`
    - `vsDebuggerExtension`

Where *`xxx`* is the three-digit update number and the folders under `bin`, `include` and `lib` are used as follows:

- `ia32`: Files used to build applications that run on IA-32
- `intel64` or `em64t`: Files used to build applications that run on Intel® 64
- `ia64`: Files used to build applications that run on IA-64
- `ia32_intel64`: Compiler that runs on IA-32 to build applications that run on Intel®64
- `ia32_ia64`: Compiler that runs on IA-32 (or Intel® 64) to build applications that run on IA-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 and IA-64 architecture systems, the folder name is `Program Files (X86)` or the equivalent.

3 Intel® C++ Compiler

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

3.1 Compatibility

In version 11, the IA-32 architecture default for code generation has changed to assume that Intel® Streaming SIMD Extensions 2 (Intel® SSE2) instructions are supported by the processor on which the application is run. See below for more information.

3.2 New and Changed Features

Please refer to the compiler documentation for details

- Additional features from C++0x
- C++ lambda functions
- Language extensions for parallel execution
- Asynchronous I/O extensions
- Decimal floating point
- #pragma vector_nontemporal
- #pragma unroll_and_jam
- valarray implementation using IPP option
- Parallel debug code instrumentation useful for thread data sharing and reentrant call detection

3.2.1 Intel® C++ Project File Compatibility

The Intel C++ project file (`.icproj`) format changed in version 11.0. If you open a project created with an earlier version of Intel C++, you will get a message indicating that the project needs to be converted. A version 11 project cannot be used by an earlier version of the Intel C++ integration (but you can use older versions of the compiler that you have installed through Tools > Options > Intel C++ > Compilers.)

3.3 New and Changed Compiler Options

Please refer to the compiler documentation for details

- /hotpatch
- /Qdiag-enable:sc-parallel
- /Yd

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

3.4 Other Changes and Known Issues

3.4.1 Optimization Reports Disabled by Default

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

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

3.4.2 Build Environment Command Script Change

The command window script used to establish the build environment changed in version 11.0. 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:

```
"C:\Program Files\Intel\Compiler\11.1\xxx\Bin\iclvars.bat" argument
```

Where `xxx` is the update number and `argument` is one of `ia32`, `ia32_intel64`, `intel64`, `ia32_ia64`, `ia64` as described above under Installation Folders. If you have installed the compiler into a different path, make the appropriate adjustments in the command.

3.4.3 Instruction Set Default Changed to Require Intel® Streaming SIMD Extensions 2 (Intel® SSE2)

When compiling for the IA-32 architecture, `/arch:SSE2` (formerly `/QxW`) is the default as of version 11.0. Programs built with `/arch:SSE2` in effect require that they be run on a processor that supports the Intel® Streaming SIMD Extensions 2 (Intel® SSE2), such as the Intel® Pentium® 4 processor and certain AMD* processors. No run-time check is made to ensure compatibility – if the program is run on a processor that does not support the instructions, an invalid instruction fault may occur. Note that this may change floating point results since the Intel® SSE instructions will be used instead of the x87 instructions and therefore computations will be done in the declared precision rather than sometimes a higher precision.

All Intel® 64 architecture processors support Intel® SSE2.

To specify the older default of generic IA-32, specify `/arch:IA32`

3.4.4 OpenMP* Libraries Default to "compat"

In version 10.1, a new set of OpenMP* libraries was added that allowed applications to use OpenMP code from both Intel and Microsoft compilers. These "compatibility" libraries can provide higher performance than the older "legacy" libraries. Beginning in version 11.0, the compatibility libraries are used by default for OpenMP applications, equivalent to `/Qopenmp-lib:compat`. If you wish to use the older libraries, specify `/Qopenmp-lib:legacy`

The "legacy" libraries (libguide.lib, libguide40.lib, etc.) will be removed in a future release of the Intel compilers.

3.4.5 Samples Provided as ZIP Archives

As of version 11.1, the compiler programming samples are provided as ZIP archives. Unpack each ZIP archive to a writable folder. All samples are now provided as Visual Studio* solutions; command-line build instructions are also provided. Please read the `samples.htm` file for more information.

3.4.6 Error Viewing Documentation in Visual Studio .NET 2003

If you are using Microsoft Visual Studio .NET 2003 but have not installed the Microsoft MSDN Library feature, you will get an error "Help Is Not Installed for Visual Studio" when using the Help menu item for *Intel® C++ Compiler Professional Edition Help*, or when using F1 contextsensitive help. As a workaround, select Help > Contents and click OK on the error message box which is then displayed. You will then be able to access the product help in the Contents pane. This issue does not affect Microsoft Visual Studio 2005 or 2008.

3.4.7 Using Intel C++ Projects with a Source Control System

If your project is managed under a source control system, for example, Microsoft Visual Source Safe* or Microsoft Visual Studio Team Foundation Server*, there are additional steps you must follow in order to use the Intel C++ project system with your project. A detailed article on this topic is available at http://software.intel.com/en-us/articles/tips-on-using-the-intel-c-compiler-with-source-code-control-software/

3.4.8 New Option to Clean Project after Conversion to use Intel C++

When the option is selected to use Intel C++ for a project, a new dialog box appears offering to “clean” the project (remove results of previous builds). This step is recommended and is selected by default; you may choose to disable the “clean” operation so that you may perform the “clean” manually.

However, if you are using a non-English version of Microsoft Visual Studio, this dialog does not appear and you must “clean” the project manually.

4 Intel® Integrated Performance Primitives

This section summarizes changes, new features and late-breaking news about the Intel® Integrated Performance Primitives as part of Intel C++ Compiler.

4.1 New and Changed Features

- New function implementation in Image Processing domain ippiCopy* and ippiTranspose* functions
- Other new function implementations in speech coding and signal processing domains. Check "NewFunctionsList.txt" for more details
- New unified image codec (UIC) frameworks implementation to standardize the interfaces as plug-and-play of various image codecs
- Intel® Atom™ Processor support
- High-level Data Compression library Support lzo and new continued performance improvement for zlib, gzip, bzip2
- A new sample for DMIP Deferred Mode of Image Processing over Intel IPP binary and API
- Intel® QuickAssist functional API for Cryptography
- New Domain - Data Integrity Functions based on operations over finite fields for error-correcting coding
- Generated domain/functionality (Spiral)
- Video Enhancement Denoising / Deinterlacing / Demosaicing
- Image Search descriptors (MPEG7), Color layout, Edge Histogram
- Microsoft RT Audio Support (enhancement)
- New Speech Coding Standard G729.1 Codec Support

- Super Resolution Technology, Optical Flow
- New Video AVS Codec Support for Decoding
- New Image Processing functions for 3D Support, Geom WarpAffine
- New Cryptography function support for Reed-Solomon Algorithm
- Threaded Static Libraries
- ALS Decoder Profile support in AAC Decoding

4.2 Intel® IPP Cryptography libraries are available as a separate download

The Intel® IPP cryptography libraries are available as a separate download. For download and installation instructions, login to the Intel® Registration Center and review the product description.

5 Intel® Math Kernel Library

This section summarizes changes, new features and late-breaking news about the Intel® Math Kernel Library as part of Intel C++ Compiler.

5.1 Change History

- Performance Improvements in the BLAS:
  - 32-bit improvements
    - 40-50% improvement for (Z,C)GEMM on Quad-Core Intel® Xeon® processor 5300 series
    - 10% improvement for all GEMM code on Quad-Core Intel® Xeon® processor 5400 series
  - 64-bit improvements
    - 2.5-3% improvement for DGEMM on 1 thread on Quad-Core Intel® Xeon® processor 5400 series
    - 50% improvement for SGEMM on the Intel® Core™ i7 processor family
    - 3% improvement for CGEMM on 1 thread on the Intel® Core™ i7 processor family
    - 2-3% improvement for ZGEMM on 1 thread on the Intel® Core™ i7 processor family
    - 30% improvement for right-side cases of DTRSM on the Intel® Core™ i7 processor family
- Improvements to the direct sparse solver (DSS/PARDISO):
  - The performance of out-of-core PARDISO was improved by 35% on average.
  - Support of separate backward/forward substitution for DSS/PARDISO has been added.
  - A new parameter for turning off iterative refinement for DSS interface has been introduced.
  - A new parameter for checking sparse matrix structure has been introduced for PARDISO interface.
- The capability to track the progress of a lengthy computation and/or interrupt the computation has been added via a callback function mechanism. A function called mkl_progress can be defined in a user application, which will be called regularly from a subset of the MKL LAPACK routines. See the LAPACK Auxiliary and Utility Routines

chapter in the reference manual for more information. Refer to the specific function descriptions to see which LAPACK functions support the feature.

- Transposition functions have been added to Intel MKL. See the reference manual for further detail.
- The C++ std::complex type can now be used instead of MKL-specific complex types.
- An implementation of the Boost uBLAS matrix-matrix multiplication routine is now provided which will make use of the highly optimized version of DGEMM in the Intel MKL BLAS. See the User guide for more information.
- Improvements to the sparse BLAS:
    - Support for all data types (single precision, complex and double complex) has been added.
    - Routines for computing the sum and product of two sparse matrices stored, both stored in the compressed sparse row format have been added.
- The Vector Math Library functions, CdfNorm, CdfNormInv, and ErfcInv, have been optimized to achieve much improved performance.
- Performance improvement on the Intel® Core™ i7 processor family:
    - 3-17% improvement for the following VML functions: Asin, Asinh, Acos, Acosh, Atan, Atan2, Atanh, Cbrt, CIS, Cos, Cosh, Conj, Div, ErfInv, Exp, Hypot, Inv, InvCbrt, InvSqrt, Ln, Log10, MulByConj, Sin, SinCos, Sinh, Sqrt, Tanh.
    - 7-67% improvement for uniform random number generation.
    - 3-10% improvement for VSL distribution generators based on Wichmann-Hill, Sobol, and Niederreiter BRNGs (64-bit only).
- The configuration file functionality has been removed. See the user guide for alternative means to configure the behavior of Intel MKL.
- All hurdles to creation of DLLs from the static libraries have been removed.
- When functions in Intel MKL are called from an MPI program they will be run on 1 thread by default (i.e., in the absence of explicit controls).
- The following VML functions have been added: CdfNorm, CdfNormInv, and ErfcInv.
- The DftiCopyDescriptor function has been added.
- The LP64 interface of DSS/PARDISO now uses 64-bit addressing for internal arrays on 64-bit operating systems. This allows the direct solver to solve larger systems.
- The default OpenMP runtime library for Intel MKL has been changed from libguide to libiomp. See the User Guide in the doc directory for more information.
- Documentation updates:
    - The parallel BLAS (PBLAS) which support ScaLAPACK are now documented in the Intel MKL reference manual.
    - Added instructions for using example programs in Microsoft* Visual Studio to the User Guide.
    - MKL Documentation is now accessible from the Microsoft* Visual Studio 'Help' menu with F1 Help and Dynamic Help features provided in the code editor. For more information, see Intel MKL User's Guide.
- It is no longer possible to set environment variables during the installation process. Three script files, mklvars32.bat, mklvarsem64t.bat, and mklvars64.bat are available in the tools\environment directory to set the PATH, LIB, and INCLUDE environment variables at the command prompt.
- The optimized code paths for the Intel® Pentium® III processor have been removed from Intel MKL along with the associated processor specific dynamic link libraries. We continue to support the use of Intel MKL on this processor, but the default code path will be used and as a result performance may be reduced.
- The interval linear solver functions have been removed from MKL.
- Documentation updates:

- The FFTW Wrappers for MKL Notes have been removed from the product package after their content was integrated into the Intel MKL Reference Manual (Appendix G).
- New functions have been documented in the reference manual, and support for Boost uBLAS matrix-matrix multiplication has been described in the User Guide.

5.2 Known Limitations

The section of the Intel MKL User's Guide titled "Integrating a Microsoft Visual Studio* IDE Project with Intel MKL" does not apply to this version. You will need to manually add the paths for the Intel MKL Include and Lib folders to the lists of folders used by the compiler as shown in the preceding sections of the Intel MKL User's Guide.

Limitations to the sparse solver and optimization solvers:

- Sparse and optimization solver libraries functions are only provided in static form

Limitations to the FFT functions:

- Mode DFTI_TRANSPOSE is implemented only for the default case
- Mode DFTI_REAL_STORAGE can have the default value only and cannot be set by the DftiSetValue function (i.e. DFTI_REAL_STORAGE = DFTI_REAL_REAL)
- The ILP64 version of Intel® MKL does not currently support FFTs with any one dimension larger than 2^31-1. Any 1D FFT larger than 2^31-1 or any multi-dimensional FFT with any dimension greater than 2^31-1 will return the "DFTI_1D_LENGTH_EXCEEDS_INT32" error message. Note that this does not exclude the possibility of performing multi-dimensional FFTs with more than 2^31-1 elements; as long as any one dimension length does not exceed 2^31-1
- Some limitations exist on arrays sizes for Cluster FFT functions. See mklman.pdf for a detailed description
- When a dynamically linked application uses Cluster FFT functionality, it is required to put the static Intel® MKL interface libraries on the link line as well. For example: -Wl,--start-group $MKL_LIB_PATH/libmkl_intel_lp64.a $MKL_LIB_PATH/libmkl_cdft_core.a -Wl,--end-group $MKL_LIB_PATH/libmkl_blacs_intelmpi20_lp64.a -L$MKL_LIB_PATH -lmkl_intel_thread -lmkl_core -liomp5 -lpthread

Limitations to the LAPACK functions:

- The ILAENV function, which is called from the LAPACK routines to choose problem-dependent parameters for the local environment, cannot be replaced by a user's version
- second() and dsecnd() functions may not provide correct answers in the case where the CPU frequency is not constant.
- As of version 10.0 the following two issues apply when linking to the dynamic libraries:

    - A user provided XERBLA will not be invoked if LAPACK is called with illegal input parameters. The default XERBLA will be used instead.
    - A user may encounter a segment violation if they call the LP64 interface to LAPACK with illegal parameters. This is because a request may be made to allocate a negative amount of memory.

Limitations to the Vector Math Library (VML) and Vector Statistical Library (VSL) functions:

- Usage of mkl_vml.fi may produce warning about TYPE ERROR_STRUCTURE length
- In case user needs to build custom DLL that contains references to Intel® MKL functions, the Intel® MKL DLL Builder Tool should be used. Other DLL build techniques are not supported

Limitations to the ScaLAPACK functions:

- The user can not substitute PJLAENV for their own version. This function is called by ScaLAPACK routines to choose problem-dependent parameters for the local environment.
- There are possible problems with getting global environment variables such as MKL_BLACS_MPI by -genvlist by MPICH2. In this case, try to set all necessary environment variables by using the control panel. From the System control panel select the "Advanced" tab and click the "Environment Variables" button.

Limitations to the ILP64 version of Intel® MKL:

- The ILP64 version of Intel® MKL does not contain the complete functionality of the library. For a full listing of what is in the ILP64 version refer to the user's guide in the doc directory.

Limitations to the Java examples:

- The Java examples don't work if the path to the JDK contains spaces. Please use quotes to set JAVA_HOME in those cases. For example: set JAVA_HOME="C:\Program Files\Java\jdk1.6.0_06"

The DHPL_CALL_CBLAS option is not allowed when building the hybrid version of MP LINPACK.

We recommend that /Od be used for the Intel® compilers when compiling test source code available with Intel® MKL. Current build scripts do not specify this option and default behavior for these compilers has changed to provide vectorization.

Limitations to dummy libraries:

- Dummy libraries cannot be used in #pragma constructions. Dummy libraries cannot be linked by Intel compiler as a driver. Please see Chapter 3 of User's Guide for more information

All VSL functions return an error status, i.e., default VSL API is a function style now rather than a subroutine style used in earlier Intel® MKL versions. This means that Fortran users should call VSL routines as functions. For example:

```
  errstatus = vslrnggaussian(method, stream, n, r, a, sigma)
```

rather than subroutines:

```
  call vslrnggaussian(method, stream, n, r, a, sigma)
```

Nevertheless, Intel® MKL provides a subroutine-style interface for backward compatibility. To use subroutine-style interface, manually include mkl_vsl_subroutine.fi file instead of mkl_vsl.fi by changing the line include 'mkl_vsl.fi' mkl.fi (in the include directory) with the line include 'mkl_vsl_subroutine.fi'. VSL API changes don't affect C/C++ users.

Memory Allocation: In order to achieve better performance, memory allocated by Intel® MKL is not released. This behavior is by design and is a one-time occurrence for Intel® MKL routines that require workspace memory buffers. Even so, the user should be aware that some tools may report this as a memory leak. Should the user wish, memory can be released by the user program through use of a function (MKL_FreeBuffers()) made available in Intel® MKL or memory can be released after each call by setting the environment variable MKL_DISABLE_FAST_MM (see User's Guide in the doc directory for more details). Using one of these methods to release memory will not necessarily stop programs from reporting memory leaks, and in fact may increase the number of such reports should you make multiple calls to the library thereby requiring new allocations with each call. Memory not released by one of the methods described will be released by the system when the program ends. To avoid this restriction disable memory management as described above.

Other: The GMP component is located in the solver library. For Intel® 64 and IA-64 platforms these components support only LP64 interface.

Using /MT when linking with multi-threaded Intel® MKL is recommended. Use of /MD with Microsoft* Visual C++* .NET 2003 may cause linking errors.

6 Intel® Threading Building Blocks

This section summarizes changes, new features and late-breaking news about the Intel Threading Building Blocks as part of Intel C++ Compiler.

6.1 Known Issues

Please note the following with respect to this particular release of Intel Threading Building Blocks.

6.1.1 Issue When Multiple Visual Studio Versions Are Installed

Intel Threading Building Blocks (Intel® TBB) provides separate sets of dynamic-linked libraries (DLLs) for the various supported versions of Microsoft Visual Studio. If you have more than one supported Visual Studio version installed, for example, 2005 and 2008, installing Intel Threading Building Blocks will add the DLL folders for all supported Visual Studio versions to the system `PATH` environment variable; typically with the newest version listed first. Since the DLLs have the same names for the different Visual Studio versions, this means that only the first set as found on `PATH` will be used. This may cause problems for applications built with one version of Visual Studio but run using the DLLs for a different version.

If you have more than one Visual Studio version installed, Intel recommends that you choose one version for your Intel® TBB application development and remove the folder for the other version from `PATH`. The folder names are of the form:

- VS2003: C:\Program Files\Intel\Parallel Studio\Composer\tbb\ia32\vc7.1\bin
- VS2005: C:\Program Files\Intel\Parallel Studio\Composer\tbb\ia32\vc8\bin
- VS2008: C:\Program Files\Intel\Parallel Studio\Composer\tbb\ia32\vc9\bin

The folder path may vary depending on architecture and system language. Intel Threading Building Blocks does not provide static libraries.

6.1.2 Library Issues

- To allow more accurate results to be obtained with Intel® Thread Checker or Intel® Thread Profiler, download the latest update releases of these products before using them with Intel Threading Building Blocks.
- If you are using Intel Threading Building Blocks and OpenMP* constructs mixed together in rapid succession in the same program, and you are using Intel compilers for your OpenMP* code, set KMP_BLOCKTIME to a small value (e.g., 20 milliseconds) to improve performance. This setting can also be made within your OpenMP* code via the kmp_set_blocktime() library call. See the Intel compiler OpenMP* documentation for more details on KMP_BLOCKTIME and kmp_set_blocktime().
- In general, non-debug ("release") builds of applications or examples should link against the non-debug versions of the Intel Threading Building Blocks libraries, and debug builds should link against the debug versions of these libraries. On Windows systems, compile with /MD and use Intel Threading Building Blocks release libraries, or compile with /MDd and use debug libraries; not doing so may cause run-time failures. See the Tutorial in the product "Documentation" sub-directory for more details on debug vs. release libraries.

7 Intel® Parallel Debugger Extension

This section summarizes changes, new features and late-breaking news about the Intel Parallel Debugger Extension as part of Intel C++ Compiler.

7.1 Known Issues

- If you are using Microsoft Visual Studio 2005, the “Intel(R) Debug Exception” needs to be enabled manually. Select `Debug` > `Exceptions`, expand the `Win32 Exceptions` tree, and enable item `a1a01db0 Intel(R) Debug Exception`. This needs to be done once per project.

- If you are using Microsoft Visual Studio 2008 and debugging 64-bit applications, you must have Visual Studio 2008 Service Pack 1 installed.
- If you also have Intel® Visual Fortran Compiler installed, you may see an error message from the Intel Parallel Debugger Extension when running Fortran programs under the Visual Studio debugger. This message can be ignored.
- Local variables are displayed as "`???`" in the Data Sharing Events window.
- The SSE Registers window does not work for 64-bit applications The window shows "`???`"
- Double-clicking on a data-sharing event may open a separate source window.
- Filters cannot be set on an offset from a symbol, e.g. "`my_variable + 0x4`" will not be accepted. Please use the address, instead.
- Taskwaits, Locks, and Barriers are currently not displayed.
- OpenMP and Data Sharing windows will not work free-floating. They work when docked.
- Attach to an already running application with debug:parallel instrumentation does not work. The application needs to be started within the debugger.
- OpenMP Task IDs are not displayed correctly until a breakpoint is reached.
- The debugger may not stop at a source code line where memory sharing was detected. The displayed event will contain the correct line number.
- Filters may not be set correctly when a program is rerun. The filter may be displayed as being active when it is not. Use the `Reevaluate Context` menu item to reestablish the filter.
- Filters need to be reevaluated manually to cover newly loaded shared libraries.
- Code filters do not filter tasks correctly.
- Performance may decrease significantly when the number of tasks exceeds the number of cores.
- Reentrant call detection does not work correctly for static functions. When used in design mode, the function should be preceded by a suitable context operator, for example, `{,,myapp.exe} my_extern_function`
- If you previously had a Beta version of Intel® Parallel Composer installed you may find that Microsoft Visual Studio will popup an error message "Microsoft Visual Studio has encountered an internal error. …" after starting or stopping debugging of an application. To solve this please delete the following file:

  `C:\Documents and Settings\<USERNAME>\Application Data\Microsoft\VisualStudio\9.0\windows.prf` for Visual Studio 2008 on Windows XP

  `C:\Users\<USERNAME>\AppData\Roaming\Microsoft\VisualStudio\9.0\windows.prf` for Visual Studio 2008 on Windows Vista

`C:\Documents and Settings\<USERNAME>\Application Data\Microsoft\VisualStudio\8.0\windows.prf` for Visual Studio 2005 on Windows XP

`C:\Users\<USERNAME>\AppData\Roaming\Microsoft\VisualStudio\8.0\windows.prf` for Visual Studio 2005 on Windows Vista

You may need to enable viewing of hidden files and folders to see this file. To do this, open a folder window and select Tools > Folder Options. Click the View tab and select the option Files and Folders > Hidden files and folders > Show hidden files and folders. Click “Apply to All Folders” then click OK, OK.

Deleting this file will reset all Visual Studio windows to default configurations and resolve this problem.

7.2 Documentation

Intel Parallel Debugger Extension Documentation can be accessed via the Help menu of Microsoft Visual Studio or by clicking the Help button of specific dialog boxes.

8 Disclaimer and Legal Information