Intel® Parallel Studio XE 2015 Composer Edition for Fortran OS X* Installation Guide and Release Notes

6 August 2015

Table of Contents

1 Introduction

This document describes how to install the product, provides a summary of new and changed product features and notes about features and problems not described in the product documentation. For the most current update to these release notes, see the release notes posted at the Intel® Software Development Products Registration Center where you downloaded this product.

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

1.1 Change History

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

1.1.1 Changes in Update 5

- Intel® Visual Fortran Compiler updated to version 15.0.4
- Intel® Math Kernel Library 11.2 Update 4

- Corrections to reported problems

1.1.2 Changes in Update 4

- No update 4 for this product

1.1.3 Changes in Update 3

- Intel® Fortran Compiler updated to version 15.0.3
- Intel® Math Kernel Library 11.2 Update 3
- Corrections to reported problems

1.1.4 Changes in Update 2

- New Library Support for Assignment with Allocatable Polymorphic Components
- Intel® Fortran Compiler updated to version 15.0.2
- Intel® Math Kernel Library 11.2 Update 2
- GNU* Project Debugger (GDB*) 7.8
- Corrections to reported problems

1.1.5 Changes in Update 1

- Support for Intel® Advanced Vector Extensions 512 instructions for IA-32 and Intel® 64 architectures in 15.0.1
- MIN/MAX Reductions supported in SIMD Loop Directive
- OS X* 10.10 now supported
- Intel® Fortran Compiler updated to version 15.0.1
- Intel® Math Kernel Library 11.2 Update 1
- Corrections to reported problems

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

- Intel® Fortran Compiler updated to 15.0
    - New Optimization Report interface, structure, and options (users of existing options -opt-report, -vec-report, -openmp-report, and -par-report are strongly encouraged to consult the Intel Compiler User's Guide for additional details)
- Intel® Math Kernel Library updated to 11.2
- GNU* GDB 7.7 with improved Fortran support
- Intel® Debugger has been removed
- Select custom installation configurations with the online installer
- Corrections to reported problems

1.2 Product Contents

Intel® Parallel Studio XE 2015 Composer Edition for Fortran OS X Update 5* includes the following components:

- Intel® Fortran Compiler XE 15.0.4 for building applications that run on Intel-based Mac* systems running the OS X* operating system
- GNU* Project Debugger (GDB*) 7.8

- Intel® Math Kernel Library 11.2 Update 4
- Integration into the Xcode* development environment (Limited Feature)
- On-disk documentation

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

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

1.4 System Requirements

- An Intel® 64 architecture based Apple* Mac* system
- 2GB RAM minimum, 4GB RAM recommended
- 4GB free disk space
- One of the following combinations of OS X*, Xcode* and the Xcode SDK:
  - OS X 10.10 and Xcode* 6.0, 6.1, 6.2, 6.3 or 6.4
  - OS X 10.9 and Xcode* 5.0 or 5.1
- If doing command line development, the Command Line Tools component of Xcode* is required

Note: Advanced optimization options or very large programs may require additional resources such as memory or disk space.

1.5 Documentation

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

1.6 Optimization Notice

Optimization Notice
Intel's compilers may or may not optimize to the same degree for non-Intel microprocessors for optimizations that are not unique to Intel microprocessors. These optimizations include SSE2, SSE3, and SSSE3 instruction sets and other optimizations. Intel does not guarantee the availability, functionality, or effectiveness of any optimization on microprocessors not manufactured by Intel. Microprocessor-dependent optimizations in this product are intended for use with Intel microprocessors. Certain optimizations not specific to Intel microarchitecture are reserved for Intel microprocessors. Please refer to the applicable product User and Reference Guides for more information regarding the specific instruction sets covered by this notice. Notice revision #20110804

1.7 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

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 will be using Xcode*, please make sure that a supported version of Xcode is installed. If you install a new version of Xcode in the future, you must reinstall the Intel Fortran Compiler afterwards.

You will need to have administrative or "sudo" privileges to install, change or uninstall the product.

After downloading the compiler product, double-click the downloaded file. When the disk image opens, double-click on the `xxx.mpkg` file to begin installation.

Follow the prompts to complete installation.

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

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

2.1 Online Installer

The default electronic installation package now consists of a smaller installation package that dynamically downloads and then installs packages selected to be installed. This requires a working internet connection and potentially a proxy setting if you are behind an internet proxy. Full packages are provided alongside where you download this online install package if a working internet connection is not available. The online installer may be downloaded and saved as shell script which can then be launched from the command line.

2.1.1 Storing Online Installer Download Content

The online installer stores the downloaded content in the form-factor of the standard install package which can then be copied and reused offline on other systems. The default download location is /tmp/intel/downloads. This location may be changed with the INTEL_SWTOOLS_DOWNLOAD_DIR environment variable. The online installer also supports a download only mode which allows the user to create a package without installation. This mode is enabled with the INTEL_SWTOOLS_DOWNLOAD_ONLY environment variable.

2.2 Intel® Software Manager

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

You can also volunteer to provide Intel anonymous usage information about these products to help guide future product design. This option, the Intel® Software Improvement Program, is not enabled by default – you can opt-in during installation or at a later time, and may opt-out at any time. For more information please see Intel® Software Improvement Program.

2.2.1 Silent Install

For information on automated or "silent" install capability, please see Intel® Compilers for Mac OS X* Silent Installation Guide

2.2.2 Support of Non-Interactive Custom Installation

Intel® Parallel Studio XE 2015 Composer Edition supports the saving of user install choices during an 'interactive' install in a configuration file that can then be used for silent installs. This configuration file is created when the following environment variables are set:

- `export INTEL_SWTOOLS_DUPLICATE_MODE=config_file_name`: it specifies the configuration file name. If the full path name is specified, the environment variable `INTEL_SWTOOLS_DOWNLOAD_DIR` is ignored and the installable package will be created under the same directory as the configuration file.
- `export INTEL_SWTOOLS_DOWNLOAD_DIR =dir_name`: optional, it specifies where the configuration file will be created. If this environment variable is not set, the installation package and the configuration file will be created under the default download directory:

```
/tmp/intel/downloads/<package_id>
```

2.3 Using a License Server

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

2.4 Installation Folders

The compiler installs, by default, under `/opt/intel` – this is referenced as `<install-dir>` in the remainder of this document. You are able to specify a different location.

Under `<install-dir>` are the following directories:

- `bin` – contains symbolic links to executables for the latest installed version
- `lib` – symbolic link to the lib directory for the latest installed version
- `include` – symbolic link to the include directory for the latest installed version
- `ism` – contains the Intel® Software Manager

- `man` – symbolic link to the directory containing `man` pages for the latest installed version
- `mkl` – symbolic link to the directory for the latest installed version of Intel® Math Kernel Library
- `composerxe` – symbolic link to the *`composer`*`_xe_2015` directory
- `composer_xe_2015` – directory containing symbolic links to subdirectories for the latest installed Intel® Parallel Studio XE 2015 Composer Edition release
- `composer_xe_2015.<n>.<pkg>` - physical directory containing files for a specific compiler version. `<n>` is the update number, and `<pkg>` is a package build identifier.

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

- `bin` – directory containing scripts to establish the compiler environment and symbolic links to compiler executables for the host platform
- `pkg_bin` – symbolic link to the compiler `bin` directory
- `include` – symbolic link to the compiler `include` directory
- `lib` – symbolic link to the compiler `lib` directory
- `mkl` – symbolic link to the `mkl` directory
- `debugger` – symbolic link to the `debugger` directory
- `man` – symbolic link to the `man` directory
- `Documentation` – symbolic link to the `Documentation` directory
- `Samples` – symbolic link to the `Samples` directory

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

- `bin` – all executables
- `compiler` – shared libraries and header files
- `debugger` – debugger files
- `man` – `man` pages
- `Documentation` – documentation files
- `mkl` – Intel® Math Kernel Library libraries and header files
- `Samples` – Product samples and tutorial files

If you have both the Intel C++ and Intel Fortran compilers installed, they will share folders for a given version and update.

This directory layout allows you to choose whether you want the latest compiler, no matter which version, the latest update of the Intel® Parallel Studio XE 2015 Composer Edition compiler, or a specific update. Most users will reference `<install-dir>/bin` for the `compilervars.sh [.csh]` script, which will always get the latest compiler installed. This method should remain stable for future releases.

2.5 Removal/Uninstall

It is not possible to remove the compiler while leaving any of the performance library components installed.

1. Open Terminal and set default (`cd`) to any folder outside `<install-dir>`
2. Type the command:
   `<install-dir>/composer_xe_2015.<n>.<pkg>./uninstall_fcompxe.sh`
3. Follow the prompts

If you are not currently logged in as `root` you will be asked for the `root` password.

2.6 Known Installation Issues and changes

- Remote offline activation via a code has been removed. Use of a license file or a license manager remain as options for alternative activation.

3 Intel® Fortran Compiler

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

3.1 Compatibility

In general, object code and modules compiled with earlier versions of Intel® Fortran Compiler for OS X* may be used in a build with version 15. Exceptions include:

- Sources that use the CLASS keyword to declare polymorphic variables and were built with a compiler version earlier than 12.0 must be recompiled.
- Objects built with the multi-file interprocedural optimization (`-ipo`) option must be recompiled.
- Objects that use the REAL(16) , REAL*16, COMPLEX(16) or COMPLEX*32 datatypes and 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

In releases earlier than Intel® Fortran Composer XE 2011 (compiler version 12.0), when a REAL(16) or COMPLEX(16) (REAL*16 or COMPLEX*32) item was passed by value, the stack address was aligned at 4 bytes. For improved performance, the version 12.0 and later compilers align such items at 16 bytes and expect received arguments to be aligned on 16-byte boundaries.

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

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

3.1.2 New Library Support for Assignment with Allocatable Polymorphic Components (15.0.2)

In order to resolve incorrect behavior in some circumstances when an assignment is done, either explicitly or implicitly, of a derived type containing components that are allocatable and polymorphic, a new compiler support library routine was created for version 15.0.2 and is called by the generated code. This means that if a Fortran source that has such an assignment is compiled with the 15.0.2 (or newer) compiler, it must be linked with the Fortran libraries from 15.0.2 or later. Failure to do this may lead to link errors regarding a missing routine `for_alloc_assign_v2`. In general you should always link with a library version no older than the compiler.

3.2 New and Changed Features

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

3.2.1 Features from Fortran 2003

- Parameterized Derived Types

3.2.2 Features from Fortran 2008

- BLOCK construct
- intrinsic subroutine EXECUTE_COMMAND_LINE

3.2.3 Features from OpenMP* 4.0

The following directives, clauses and procedures, from OpenMP 4.0, are supported by the compiler. For more information, see the compiler documentation or the link to the OpenMP Technical Report.

SIMD Directives:

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

Other Directives:

- OMP PARALLEL PROC_BIND
- OMP TASKGROUP

- OMP CANCEL
- OMP CANCELLATION POINT

Clauses:

- DEPEND

3.2.3.1 KMP_DYNAMIC_MODE Environment Variable Support for "asat" Deprecated

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

3.2.4 New and Changed Directives

The following compiler directives are new or changed in Intel® Parallel Studio XE 2015 Composer Edition – please see the documentation for details:

- ATTRIBUTES OPTIMIZATION_PARAMETER INLINE-MAX-TOTAL-SIZE=N
- ATTRIBUTES OPTIMIZATION_PARAMETER INLINE-MAX-PER-ROUTINE=N

3.2.4.1 ATTRIBUTES STDCALL now allowed with BIND(C)

As of compiler version 15.0, the ATTRIBUTES STDCALL directive may be specified for an interoperable procedure (a procedure whose declaration includes the BIND(C) language binding attribute.)

No other effects from STDCALL, such as pass-by-value, are provided. The Fortran standard VALUE attribute (not ATTRIBUTES VALUE) may be used if desired. For all other platforms, specifying STDCALL with BIND(C) has no effect.

3.2.5 New run-time routines to get Fortran library version numbers

- FOR_IFCORE_VERSION returns the version of the Fortran run-time library (ifcore).
- FOR_IFPORT_VERSION returns the version of the Fortran portability library (ifport).

3.2.6 Support for Intel® Advanced Vector Extensions 512 (Intel® AVX-512) instructions for IA-32 and Intel® 64 architectures in 15.0.1

The Intel® Compiler 15.0.1 now supports Intel® AVX-512 instructions for processors based on IA-32 and Intel® 64 architectures that support that instruction set. The instructions are supported via inline assembly, and/or the -[a]xCORE-AVX512 compiler options.

3.2.7 MIN/MAX Reductions supported in SIMD Loop Directive

Starting with the Intel® Compilers version SIMD Loop Directive now supports MIN/MAX reductions, like so:

```
!DIR$ SIMD REDUCTION(MAX:SIMDMAX)
      DO I = 1, SIZE
         IF (X(I) > SIMDMAX) SIMDMAX = X(I)
      END DO
```

```
!DIR$ SIMD REDUCTION(MIN:SIMDMIN)
      DO I = 1, SIZE
         IF (X(I) < SIMDMIN) SIMDMIN = X(I)
      END DO


!DIR$ SIMD REDUCTION(MAX:XMAX)
      DO I = 1, SIZE
         XMAX = MAX (XMAX, X(I))
      END DO


!DIR$ SIMD REDUCTION(MIN:XMIN)
      DO I = 1, SIZE
         XMIN = MIN (XMIN, X(I))
      END DO
```

3.3 New and Changed Compiler Options

Please refer to the compiler documentation for details.

- -assume [no]std_value
- -assume ieee_fpe_flags
- -fast
- -f[no-]eliminate-unused-debug-types
- -init=snan
- -qopt-report
- –[no-]qopt-dynamic-align
- -prof-gen=[no]threadsafe

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

3.3.1 Compiler options starting with –o are deprecated

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

3.3.2 -assume std_value is now the default

As of compiler version 15.0, the Fortran standard VALUE attribute, (not ATTRIBUTES VALUE), when specified for a dummy argument of a non-interoperable procedure (a procedure whose declaration does not include the BIND(C) language binding attribute), applies Fortran standard semantics by default. The standard specifies that for a non-interoperable procedure, VALUE causes a temporary, redefinable copy of the actual argument to be passed using the default passing mechanism. In earlier compiler versions, VALUE always caused the actual argument to be passed by value. Compiler version 14.0 introduced –assume std_value to specify the standard-conforming semantics and this was enabled if -standard-semantics was specified.

3.3.3 –assume ieee_fpe_flags enabled with -standard-semantics and –fp-model strict or –fp-model except

As of compiler version 15.0, if -standard-semantics and one of –fp-model strict or –fp-model except is specified, -assume ieee_fpe_flags is also enabled. This option causes the state of floating point exceptions to be saved on entry to a procedure and restored on exit. The save and restore operation has a significant performance penalty so it should be used only by applications that manipulate or query the floating point exception environment. Note that Intel Fortran requires that you specify –fp-model strict if you are using the Fortran standard intrinsic modules IEEE_ARITHMETIC, IEEE_EXCEPTIONS and/or IEEE_FEATURES.

3.3.4 Change to –fast option

-fp-model fast=2 has been added to the –fast option. This change makes it easier to tune for performance.

3.3.5 New –init=snan Compiler Option

This is a new command line option to help find a class of uninitialized variables at run-time by initializing floating-point variables to signaling NaNs which can then be trapped if their values are fetched before being set.

3.3.6 New Optimization Report interface, report structure, and options in Intel® Parallel Studio XE 2015 Composer Edition

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

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

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

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

3.3.7 New –[no-]qopt-dynamic-align Compiler Option

When this option is set the compiler implements conditional optimizations based on dynamic alignment of the input data for maximum performance of vectorized code especially for long trip count loops. This, however, may result in different bitwise results for aligned and unaligned data with the same values. When unset the compiler will not perform these optimizations providing bitwise reproducibility.

3.3.8 New mode for PGO instrumentation -prof-gen=[no]threadsafe

This change adds a mode to the PGO instrumentation that allows for the collection of PGO data on applications that use a high level of parallelism, such as from OpenMP.

3.4 Other Changes and Notes

3.4.1 Other Features

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

- New environment variable INTEL_PROF_DYN_PREFIX. Allows the user to have some control over the naming PGO generated ".dyn" files to make it easy to distinguish files from different runs. By setting this environment variable to the desired character string prior to starting the instrumented application, the string will be included as prefix to the .dyn file names.

3.5 Known Issues

3.5.1 Certain uses of length type parameters in parameterized derived types are not yet fully implemented

The following uses of length type parameters in parameterized derived types (PDTs) are not yet fully implemented:

- PDT parameter constants with length type parameters
- %RE and %IM are not yet implemented

3.5.2 –warn interfaces may cause intermittent build fails with parallel builds

3.6 Using –warn interfaces may cause intermittent build failures when performing parallel builds. When using –warn interfaces, it is recommended to perform a serial build.Establishing the Compiler Environment

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

The command takes the form:

```
source <install-dir>/bin/compilervars.sh argument
```

Where *xxx* is the package identifier and *argument* is either `ia32` or `intel64` as appropriate for the architecture you are building for. Establishing the compiler environment also establishes

the environment for the provided GNU* GDB (`gdb-ia`), Intel® Performance Libraries and, if present, Intel® C++ Compiler.

3.7 Fortran 2003 and Fortran 2008 Feature Summary

The Intel Fortran Compiler supports all features from the Fortran 2003 standard. The Intel® Fortran Compiler also supports many 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)
- 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 intrinsic function NULL, 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.
- BLOCK construct
- intrinsic subroutine EXECUTE_COMMAND_LINE

Coarrays are not supported on OS X.

4 GNU* Project Debugger (GDB)

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

4.1 Features

GNU* GDB provided with Intel® Parallel Studio XE 2015 Composer Edition is based on GDB 7.8 with enhancements provided by Intel. This debugger replaces the Intel® Debugger previous releases. In addition to features found in GDB 7.8, there are several other new features:

- Support for Intel® Transactional Synchronization Extensions (Intel® TSX)
- Improved Fortran support
- Moved from GNU* GDB 7.7 to GDB 7.8 with Intel® Parallel Studio XE 2015 Update 2 Composer Edition.

4.2 Using GNU* GDB

This debugger is designed to debug IA-32 or Intel® 64 architecture applications natively. Its use is no different than with traditional GNU* GDB debuggers. There are some extensions, though, which can be found in the documentation.

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

4.3 Documentation

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

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

4.4 Known Issues and Changes

4.4.1 Debugger and debugged application required to be located on local drive

In order to use the provided GNU* GDB (`gdb-ia`), it must be installed on a local drive. As such, the entire Intel® Parallel Studio XE package must be installed locally. Any application that is being debugged needs to be located on a local drive, too. This is a general requirement that is inherent to GNU GDB with OS X*.

4.4.2 Nested variable length arrays not working with user derived types

If using a variable length array (VLA) of user derived types that contain nested variable length arrays, the debugger fails to resolve the latter. A future version of the debugger will fix this issue.

5 Intel® Math Kernel Library (Intel® MKL)

This section summarizes changes, new features and late-breaking news about this version of the Intel® Math Kernel Library (Intel® MKL). Bug fixes can be found here.

5.1 What's New in Intel MKL 11.2 Update 4

- BLAS:

- Improved parallel and serial performance of ?TRSM on Intel® Advanced Vector Extensions 2 (Intel® AVX2) for 64-bit Intel MKL
- Improved ?SYRK/?HERK/?SYR2K/?HER2K performance for beta=0 on Intel® Advanced Vector Extensions 2 (Intel® AVX2) for 64-bit Intel MKL
- Improved serial performance of STRMM for small triangular matrices (dimension less than or equal to 10) on Intel® AVX2 for 64-bit Intel MKL

5.2 What's New in Intel MKL 11.2 Update 3

- Extended the Intel MKL memory manager to improve scaling on large SMP systems.
- BLAS:
  - Improved parallel performance of (D/S)SYMV on all Intel Xeon processors.
  - Improved (C/D/S/Z/DZ/SC)ROT performance for Intel Advanced Vector Extensions (Intel AVX) architectures in 64-bit Intel MKL.
  - Improved (C/Z)ROT performance for Intel Advanced Vector Extensions 2 (Intel AVX2) architectures in 64-bit Intel MKL.
  - Improved parallel performance of ?SYRK/?HERK, ?SYR2K/?HER2K, and ?GEMM for cases with large *k* sizes on Intel AVX2 architectures in 64-bit Intel MKL.
- LAPACK:
  - Improved performance of SVD, for cases where singular vectors are computed, on multi-socket systems based on Intel AVX or Intel AVX2 architectures.
  - Added new routines for incomplete LU factorization without pivoting.

5.3 What's New in Intel MKL 11.2 Update 2

- BLAS:
  - Improved parallel and serial performance of ?HEMM/?SYMM for on Intel® Advanced Vector Extensions 2 (Intel® AVX2)for the 64-bit Intel MKL.
  - Improved parallel and serial performance of ?HERK/?SYRK and and ?HER2K/?SYR2K for Intel AVX2.
  - Added MKL_DIRECT_CALL support for CBLAS interfaces and ?GEMM3M routines.
  - Improved CGEMM performance for Intel® Advanced Vector Extensions 512 (Intel® AVX-512).
  - Small performance improvement for CGEMM and ZGEMM for Intel AVX2 for the 64-bit Intel MKL.
- LAPACK:
  - Improved symmetric eigensolvers performance by up to 3x, for the cases when eigenvectors are not needed.
  - Improved ?GESVD performance by 2-3x, for the cases when singular vectors are required.

    - Improved ?GETRF performance for Intel AVX2 by up to 14x for non-square matrices.
    - Narrowed the ?GETRF performance gap between CNR (Conditional Numerical Reproducibility)-enabled and CNR-disabled cases. The gap is now below 5%.
    - Improved Intel® Optimized LINPACK Benchmark shared memory (SMP) implementation performance for Intel AVX2 by up to 40%.
- Intel® MKL PARDISO:
    - Improved the scalability of the solving step for Intel® Xeon® processors.
    - Reduced memory footprint in the out-of-core mode.
    - Added ability to free up memory used by the input matrix after the factorization step. This helps to reduce memory consumption when iterative refinement is not needed and disabled by the user.
- Extended Eigensolver:
    - Improved performance for Intel Xeon processors
- VSL:
    - Summary Statistics:
        - Improved performance of variance-covariance matrices computation and correlation matrices computation routines for cases when the task dimension is approximately equal to the number of observations.
    - RNG:
        - Improved performance of the Sobol and the Niederreiter Quasi-RNGs for Intel Xeon processors.
- Convolution and correlation:
    - Improved 3D convolution performance.

5.4 What's New in Intel MKL 11.2 Update 1

- BLAS:
    - Optimized following BLAS Level-1 functions on Intel AVX2 both for Intel® 64 and IA-32 architectures
        - (S/D)DOT,(S/D)SCAL,(S/D)ROT,(S/D)ROTM,(S/D/C/Z)SWAP,(S/D/SC/DZ)ASUM
    - Improved ?GEMM performance (serial and multithreaded) on Intel AVX2 (for IA-32 architectures)
    - Improved ?GEMM performance for beta==0 on Intel AVX and Intel AVX2 (for Intel 64 architectures)
    - Improved DGEMM performance (serial and multithreaded) on Intel AVX (for Intel 64 architectures)

- LAPACK:
    - Introduced support for LAPACK version 3.5. New features introduced in this version are:
        - Symmetric/Hermitian LDLT factorization routines with rook pivoting algorithm
        - 2-by-1 CSD for tall and skinny matrix with orthonormal columns
    - Improved performance of (C/Z)GE(SVD/SDD) when M>=N and singular vectors are not needed
- FFT:
    - Improved performance of Hybrid (OpenMP+MPI) Cluster FFT
    - Improved accuracy of large 1D real-to-complex transforms
- Parallel Direct Sparse Solver for Clusters:
    - Added support for many factorization steps with the same reordering (maxfct > 1)
- Intel MKL PARDISO:
    - Added support for Schur complement, including getting explicit Schur complement matrix and solving the system through Schur complement
- Sparse BLAS:
    - Added Sparse Matrix Checker functionality as standalone API to simplify validation of matrix structure and indices(see Sparse Matrix Checker Routines in Intel® Math Kernel Library (Intel® MKL) Reference Manual
    - Sparse BLAS API for C/C++ uses const modifier for constant parameters
- VML:
    - Introduced new Environment variable, MKL_VML_MODE to control the accuracy behavior. This Environment variable can be used to control VML functions behavior (analog of vmlSetMode() function)

5.5 What's New in Intel MKL 11.2

- Introduced support for Intel® Advanced Vector Extensions 512 (Intel® AVX-512) instruction set with limited optimizations in BLAS, DFT and VML
- Introduced Verbose support for BLAS and LAPACK domains, which enables users to capture the input parameters to Intel MKL function calls
- Introduced support for Intel® MPI Library 5.0

- Introduced the Intel Math Kernel Library Cookbook (http://software.intel.com/en-us/mkl_cookbook) , a new document that describes how to use Intel MKL routines to solve certain complex problems
- Introduced the MKL_DIRECT_CALL or MKL_DIRECT_CALL_SEQ compilation feature that provides ?GEMM small matrix performance improvements for all processors (see the *Intel® Math Kernel Library User's Guide* for more details)
- Added a customizable error handler. See the *Intel Math Kernel Library Reference Manual* description of mkl_set_exit_handler() for further details
- Parallel Direct Sparse Solver for Clusters:
    - Introduced Parallel Direct Sparse Solver for Clusters, a distributed memory version of Intel MKL PARDISO direct sparse solver
    - Improved performance of the matrix gather step for distributed matrices
    - Enabled reuse of reordering information on multiple factorization steps
    - Added distributed CSR format, support of distributed matrices, RHS, and distributed solutions
    - Added support of solving of systems with multiple right hand sides
    - Added cluster support of factorization and solving steps
    - Added support for pure MPI mode and support for single OpenMP thread in hybrid configurations
- BLAS:
    - Improved threaded performance of ?GEMM for all 64-bit architectures supporting Intel® Advanced Vector Extensions 2 (Intel® AVX2)
    - Optimized ?GEMM, ?TRSM, DTRMM for the Intel AVX-512 instruction set
    - Improved performance of Level 3 BLAS functions for 64-bit processors supporting Intel AVX2
    - Improved ?GEMM performance on small matrices for all processors when MKL_DIRECT_CALL or MKL_DIRECT_CALL_SEQ is defined during compilation (see the *Intel® Math Kernel Library User's Guide for more details*)
    - Improved performance of DGER and DGEMM for the beta=1, k=1 case for 64-bit processors supporting Intel SSE4.2, Intel® Advanced Vector Extensions (Intel® AVX), and Intel AVX2 instruction sets
    - Optimized (D/Z)AXPY for the Intel AVX-512 instruction set
    - Optimized ?COPY for Intel AVX2 and AVX-512 instruction sets
    - Optimized DGEMV for Intel AVX-512 instruction set
    - Improved performance of SSYR2K for 64-bit processors supporting Intel AVX and Intel AVX2
    - Improved threaded performance of ?AXPBY for all Intel processors
    - Improved DTRMM performance for the side=R, uplo={U,L}, transa=N, diag={N,U} cases for Intel AVX-512
- LINPACK:

    - Improved performance of matrix generation in the heterogeneous Intel® Optimized MP LINPACK Benchmark for Clusters
    - Improved performance of the Intel Optimized MP LINPACK for Clusters package for 64-bit processors supporting Intel AVX2
- LAPACK:
    - Improved performance of ?(SY/HE)RDB
    - Improved performance of ?(SY/HE)EV when eigenvectors are needed
    - Improved performance of ?(SY/HE)(EV/EVR/EVD) when eigenvectors are not needed
    - Improved performance of ?GELQF,?GELS and ?GELSS for underdetermined case (M less than N)
    - Improved performance of ?GEHRD,?GEEV and ?GEES
    - Improved performance of NaN checkers in LAPACKE interfaces
    - Improved performance of ?GELSX, ?GGSVP
    - Improved performance of ?(SY/HE)(EV/EVR/EVD) when eigenvectors are not needed
    - Improved performance of ?GETRF
    - Improved performance of (S/D)GE(SVD/SDD) when M>=N and singular vectors are not needed
- PBLAS and ScaLAPACK:
    - Enabled Automatic Offload in P?GEMM routines for large distribution blocking factors
- Sparse BLAS:
    - Optimized SpMV kernels for Intel AVX-512 instruction set
    - Added release example for diagonal format use in Sparse BLAS
    - Improved Sparse BLAS level 2 and 3 performance for systems supporting Intel SSE4.2, Intel AVX and Intel AVX2 instruction sets
- Intel MKL PARDISO:
    - Added the ability to store Intel MKL PARDISO handle to the disk for future use at any solver stage
    - Added pivot control support for unsymmetric matrices and out-of-core mode
    - Added diagonal extraction support for unsymmetric matrices and out-of-core mode
    - Added example demonstrating use of Intel MKL PARDISO as iterative solver for non-linear systems
    - Added capability to free memory taken by original matrix after factorization stage if iterative refinement is disabled
    - Improved memory estimation of out-of-core (OOC) portion size for reordering algorithm leading to improved factorization-solve performance in OOC mode

    - Improved message output from Intel MKL PARDISO
    - Added support of zero pivot during factorization for structurally symmetric cases
- Poisson library:
    - Added example demonstrating use of the Intel MKL Poisson library as a preconditioner for linear systems solves
- Extended Eigensolver:
    - Improved message output
    - Improved examples
    - Added input and output iparm parameters in predefined interfaces for solving sparse problems
- FFT:
    - Optimized FFTs for the Intel AVX-512 instruction set
- VML: Added v[d|s]Frac function computing fractional part for each vector element
- VSL RNG:
    - Added support of ntrial=0 in Binomial Random Number Generator
    - Improved performance of MT2203 BRNG on CPUs supporting Intel AVX and Intel AVX2 instruction sets
- VSL Summary Statistics:
    - Added support for group/pooled mean estimates (VSL_SS_GROUP_MEAN/VSL_SS_POOLED_MEAN)
- Data Fitting: Fixed incorrect behavior of the natural cubic spline construction function when number of breakpoints is 2 or 3
- Introduced an Intel MKL mode that ignores all settings specified by Intel MKL environment variables
    - User can set up the mode by calling mkl_set_env_mode() routine which directs Intel MKL to ignore all environment settings specific to Intel MKL so that all Intel MKL related environment variables such as MKL_NUM_THREADS, MKL_DYNAMIC and others are ignored; users can instead set needed parameters via Intel MKL service routines such as mkl_set_num_threads()

5.6 Notes

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

- API symbols, order of arguments and link line have changed since Intel MKL 11.2 Beta Update 2 . (see the *Intel® Math Kernel Library User's Guide* for more details)
- Important deprecations are listed in Intel® Math Kernel Library (Intel® MKL) 11.2 Deprecations

5.7 Known Issues

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

- An application built on OS X* and linked with libmkl_rt.so library where the first call to Intel® MKL was made in parallel section will crash with segfault or with either of these messages:

  “malloc: *** error for object xxxxx: pointer being freed was not allocated *** set a breakpoint in malloc_error_break to debug”

  or

  “malloc: *** error for object xxxxx: double free !!! *** set a breakpoint in malloc_error_break to debug”

  Workaround: call any Intel® MKL function before the parallel section

5.8 Attributions

As referenced in the End User License Agreement, attribution requires, at a minimum, prominently displaying the full Intel product name (e.g. "Intel® Math Kernel Library") and providing a link/URL to the Intel® MKL homepage (www.intel.com/software/products/mkl) in both the product documentation and website.

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

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

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

PARDISO in Intel® MKL is compliant with the 3.2 release of PARDISO that is freely distributed by the University of Basel. It can be obtained at http://www.pardiso-project.org.

Some FFT functions in this release of Intel® MKL have been generated by the SPIRAL software generation system (http://www.spiral.net/) under license from Carnegie Mellon University. The Authors of SPIRAL are Markus Puschel, Jose Moura, Jeremy Johnson, David Padua, Manuela Veloso, Bryan Singer, Jianxin Xiong, Franz Franchetti, Aca Gacic, Yevgen Voronenko, Kang Chen, Robert W. Johnson, and Nick Rizzolo.

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

6 Disclaimer and Legal Information

INFORMATION IN THIS DOCUMENT IS PROVIDED IN CONNECTION WITH INTEL(R) PRODUCTS. NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS DOCUMENT. EXCEPT AS PROVIDED IN INTEL'S TERMS AND CONDITIONS OF SALE FOR SUCH PRODUCTS, INTEL ASSUMES NO LIABILITY WHATSOEVER, AND INTEL DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY, RELATING TO SALE AND/OR USE OF INTEL PRODUCTS INCLUDING LIABILITY OR WARRANTIES RELATING TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT. UNLESS OTHERWISE AGREED IN WRITING BY INTEL, THE INTEL PRODUCTS ARE NOT DESIGNED NOR INTENDED FOR ANY APPLICATION IN WHICH THE FAILURE OF THE INTEL PRODUCT COULD CREATE A SITUATION WHERE PERSONAL INJURY OR DEATH MAY OCCUR.

Intel may make changes to specifications and product descriptions at any time, without notice. Designers must not rely on the absence or characteristics of any features or instructions marked "reserved" or "undefined." Intel reserves these for future definition and shall have no responsibility whatsoever for conflicts or incompatibilities arising from future changes to them. The information here is subject to change without notice. Do not finalize a design with this information.

The products described in this document may contain design defects or errors known as errata which may cause the product to deviate from published specifications. Current characterized errata are available on request.

Contact your local Intel sales office or your distributor to obtain the latest specifications and before placing your product order.

Copies of documents which have an order number and are referenced in this document, or other Intel literature, may be obtained by calling 1-800-548-4725, or go to:

http://www.intel.com/design/literature.htm

Intel processor numbers are not a measure of performance. Processor numbers differentiate features within each processor family, not across different processor families. Go to:

http://www.intel.com/products/processor_number/

for details.

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

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

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.