Get Started with the Intel® oneAPI DPC ++/C++ Compiler

Contents

Get Started with the Intel® oneAPI DPC++/C++ Compiler 1

The Intel® oneAPI DPC++/C++ Compiler provides optimizations that help your applications run faster on Intel® 64 architectures on Windows* and Linux*, with support for the latest C, C++, and SYCL language standards. This compiler produces optimized code that can run significantly faster by taking advantage of the ever-increasing core count and vector register width in Intel® Xeon® processors and compatible processors. The Intel® Compiler will help you boost application performance through superior optimizations and Single Instruction Multiple Data (SIMD) vectorization, integration with Intel® Performance Libraries, and by leveraging the OpenMP* 5.0/5.1 parallel programming model.

The Intel® oneAPI DPC++/C++ Compiler compiles C++-based SYCL* source files for a wide range of compute accelerators.

The Intel® oneAPI DPC++/C++ Compiler is part of the Intel® oneAPI Toolkits.

Find More

Content	Description and Links
Release Notes	Visit the Release Notes page for known issues and the most up-to-date information.
Intel® oneAPI Programming Guide	Provides details on the Intel® oneAPI DPC++/C++ Compiler programming model, including details about SYCL* and OpenMP* offload, programming for various target accelerators, and introductions to the Intel® oneAPI libraries.
Intel® oneAPI DPC++/C++ Compiler Developer Guide and Reference	Explore Intel® oneAPI DPC++/C++ Compiler features and setup and get more detailed information about compiler options, attributes, and more.
oneAPI Code Samples	Explore the latest oneAPI code samples.
• Intel® oneAPI Data Parallel C++ Forum • Intel® C++ Compiler Forum	Ask questions and find answers in the Intel® oneAPI Data Parallel C++ and Intel® C++ Compiler forums.
Intel® oneAPI DPC++/C++ Compiler Documentation	Explore tutorials, training materials, and other Intel® oneAPI DPC++/C++ Compiler documentation.
SYCL Specification Version 1.2.1 PDF	The SYCL specification, explains how SYCL integrates OpenCL devices with modern C++.
https://www.khronos.org/sycl/	An overview of SYCL.
The GNU* C++ Library - Using Dual ABI	The GNU* C++ Library documentation on using dual ABI.
Layers for Yocto* Project	Add oneAPI components to a Yocto project build using the meta-intel layers.

Notices and Disclaimers

Intel technologies may require enabled hardware, software or service activation.

No product or component can be absolutely secure.

Your costs and results may vary.

Get Started on Linux*

Before You Begin

Set Environment Variables

Before you can use the compiler, you must first set the environment variables by sourcing the environment script using the initialization utility. This initializes all the tools in one step.

1. Determine your installation directory,`<install_dir>`:
   a. If your compiler was installed in the default location by a root user or sudo user, the compiler will be installed under`/opt/intel/oneapi`. In this case, `<install_dir>` is `/opt/intel/oneapi`.
   b. For non-root users, your home directory under `intel/oneapi` is used. In this case, `<install_dir>` will be `$HOME/intel/oneapi`.
   c. For cluster or enterprise users, your admin team may have installed the compilers on a shared network file system. Check with your local admin staff for the location of installation (`<install_dir>`).
2. Source the environment-setting script for your shell:
   a. bash: `source <install_dir>/setvars.sh intel64`
   b. csh/tcsh: `source <install_dir>/setvars.csh intel64`

Install GPU Drivers or Plug-ins (Optional)

You can develop oneAPI applications using C++ and SYCL* that will run on Intel, AMD*, or NVIDIA* GPUs.

To develop and run applications for specific GPUs you must first install the corresponding drivers or plug-ins:

- To use an Intel GPU, install the latest Intel GPU drivers.
- To use an AMD GPU, install the oneAPI for AMD GPUs plugin from Codeplay.
- To use an NVIDIA GPU, install the oneAPI for NVIDIA GPUs plugin from Codeplay.

Option 1: Use the Command Line

The Intel® oneAPI DPC++/C++ Compiler provides multiple drivers:

Language	Linux Drivers	Windows Drivers	Option Style	Notes
C	**icx** icx-cc	icx-cc	Linux-style	**icx is the recommended default C driver for Linux.**

Language	Linux Drivers	Windows Drivers	Option Style	Notes
C++	**icpx**	icpx	Linux-style	If you use icx with a C++ source file, it is compiled as a C++ file. Use icx to link C object files. icx-cc is the Microsoft-compatible variant of icx. **icpx is the recommended default C++ driver for Linux.** If you use icpx with a C source file, it is compiled as an C++ file. Use icpx to link C++ object files.
C/C++	icx-cl (see notes)	**icx** icx-cl	Windows-style	**icx is the recommended default driver for Windows.** icx-cl is the Microsoft-compatible variant of icx. **NOTE** On Linux, icx-cl is experimental and requires the Microsoft Visual Studio Package.

Invoke the compiler using the following syntax:

```
{compiler driver} [option] file1 [file2...]
```

For example:

```
icpx hello-world.cpp
```

For SYCL compilation, use the `-fsycl` option with the C++ driver:

```
icpx -fsycl hello-world.cpp
```

NOTE When using `-fsycl`, `-fsycl-targets=spir64` is assumed unless the `-fsycl-targets` is explicitly set in the command.

If you are targeting an AMD or NVIDIA GPU, refer to the corresponding Codeplay plugin get started guide for detailed compilation instructions:

- oneAPI for AMD GPUs Get Started Guide
- oneAPI for NVIDIA GPUs Get Started Guide

Option 2: Use the Eclipse* CDT

Follow these steps to invoke the compiler from within the Eclipse* CDT.

Install the Intel® Compiler Eclipse CDT plugin.

1. Start Eclipse
2. Select **Help > Install New Software**
3. Select **Add** to open the Add Site dialog
4. Select **Archive**, browse to the directory `<install_dir>/compiler/<version>/linux/ide_support`, select the .zip file that starts with `com.intel.dpcpp.compiler`, then select **OK**
5. Select the options beginning with Intel, select **Next**, then follow the installation instructions
6. When asked if you want to restart Eclipse*, select **Yes**

Build a new project or open an existing project.

1. Open **Existing Project** or **Create New Project** on Eclipse
2. Right click on **Project > Properties > C/C++ Build > Tool chain Editor**
3. Select **Intel DPC++/C++ Compiler** from the right panel

Set build configurations.

1. Open **Existing Project** on Eclipse
2. Right click on **Project > Properties > C/C++ Build > Settings**
3. Create or manage build configurations in the right panel

Build a Program From the Command Line

Use the following steps to test your compiler installation and build a program.

1. Use a text editor to create a file called `hello-world.cpp` with the following contents:

```
#include <iostream>

int main()
{
    std::cout << "Hello, world!\n";

    return 0;
}
```

2. Compile `hello-world.cpp`:

```
icpx hello-world.cpp -o hello-world
```

   The `-o` option specifies the file name for the generated output.

3. Now you have an executable called `hello-world` which can be run and will give immediate feedback:

```
hello-world
```

   Which outputs:

```
Hello, world!
```

You can direct and control compilation with compiler options. For example, you can create the object file and output the final binary in two steps:

1. Compile `hello-world.cpp`:

```
icpx hello-world.cpp -c
```

   The `-c` option prevents linking at this step.
2. Use the `icpx` compiler to link the resulting application object code and output an executable:

```
icpx hello-world.o -o hello-world
```

   The `-o` option specifies the generated executable file name.

Refer to Compiler Options for details about available options.

Get Started on Windows*

Before You Begin

Set Environment Variables

The compiler integrates into the following versions of Microsoft Visual Studio*:

- Visual Studio 2022
- Visual Studio 2019
- Visual Studio 2017

> **NOTE** Support for Microsoft Visual Studio 2017 is deprecated as of the Intel® oneAPI 2022.1 release and will be removed in a future release.

For full functionality within Visual Studio, including debugging and development, Visual Studio Community Edition or higher is required. Visual Studio Express Edition allows only command-line builds. For all versions, Microsoft C++ support must be selected as part of the Visual Studio install. For Visual Studio 2017 and later, you must use a custom install to select this option.

You typically do not need to set the environment variables on Windows, as the compiler command-line window sets these variables for you automatically. If you need to set the environment variables, run the environment script as described in the suite-specific Get Started documentation.

The default installation directory (`<install_dir>`) is `C:\Program Files (x86)\Intel\oneAPI`.

Install GPU Drivers (Optional)

To develop and run applications for Intel GPUs you must first install the latest Intel GPU drivers.

Option 1: Use the Command Line in Microsoft Visual Studio

The Intel® oneAPI DPC++/C++ Compiler provides multiple drivers:

Language	Linux Drivers	Windows Drivers	Option Style	Notes
C	**icx** icx-cc	icx-cc	Linux-style	**icx is the recommended default C driver for Linux.**

Language	Linux Drivers	Windows Drivers	Option Style	Notes
C++	**icpx**	icpx	Linux-style	If you use icx with a C++ source file, it is compiled as a C++ file. Use icx to link C object files. icx-cc is the Microsoft-compatible variant of icx. **icpx is the recommended default C++ driver for Linux.** If you use icpx with a C source file, it is compiled as an C++ file. Use icpx to link C++ object files.
C/C++	icx-cl (see notes)	**icx** icx-cl	Windows-style	**icx is the recommended default driver for Windows.** icx-cl is the Microsoft-compatible variant of icx. **NOTE** On Linux, icx-cl is experimental and requires the Microsoft Visual Studio Package.

Invoke the compiler using the following syntax:

```
{compiler driver} [option] file1 [file2...]
```

To invoke the compiler using the command line from within Microsoft Visual Studio, open a command prompt and enter your compilation command. For example:

```
icx hello-world.cpp
```

For SYCL compilation, use the `-fsycl` option with the C++ driver:

```
icx -fsycl hello-world.cpp
```

NOTE When using `-fsycl`, `-fsycl-targets=spir64` is assumed unless the `-fsycl-targets` is explicitly set in the command.

Option 2: Use Microsoft Visual Studio

Project Support for the Intel® DPC++/C++ Compiler in Microsoft Visual Studio

New Microsoft Visual Studio projects for DPC++ are automatically configured to use the Intel® oneAPI DPC++/C++ Compiler.

New Microsoft Visual C++* (MSVC) projects must be manually configured to use the Intel® oneAPI DPC++/C++ Compiler.

NOTE .NET-based CLR C++ project types are not supported by the Intel® oneAPI DPC++/C++ Compiler. The specific project types will vary depending on your version of Visual Studio, for example: CLR Class Library, CLR Console App, or CLR Empty Project.

Use the Intel® DPC++/C++ Compiler in Microsoft Visual Studio

Exact steps may vary depending on the version of Microsoft Visual Studio in use.

1. Create a Microsoft Visual C++ (MSVC) project or open an existing project.
2. In **Solution Explorer**, select the project(s) to build with the Intel® oneAPI DPC++/C++ Compiler.
3. Open **Project > Properties** .
4. In the left pane, expand the **Configuration Properties** category and select the **General** property page.
5. In the right pane change the **Platform Toolset** to the compiler you want to use:
   - For C++ with SYCL, select **Intel® oneAPI DPC++ Compiler**.
   - For C/C++, there are two toolsets.

     Select **Intel C++ Compiler <major version>** (example 2021) to invoke `icx`.

     Select **Intel C++ Compiler <major.minor>** (example 19.2) to invoke `icl`.

     Alternatively, you can specify a compiler version as the toolset for all supported platforms and configurations of the selected project(s) by selecting **Project > Intel Compiler > Use Intel oneAPI DPC++/C++ Compiler**.
6. Rebuild, using either **Build > Project only > Rebuild** for a single project or **Build > Rebuild Solution** for a solution.

Select Compiler Version

If you have multiple versions of the Intel® oneAPI DPC++/C++ Compiler installed, you can select which version you want from the Compiler Selection dialog box:

1. Select a project, then go to **Tools > Options > Intel Compilers and Libraries > <compiler> > Compilers**, where <compiler> values are **C++** or **DPC++**.
2. Use the **Selected Compiler** drop-down menu to select the appropriate version of the compiler.
3. Select **OK**.

Switch Back to the Microsoft Visual Studio C++ Compiler

If your project is using the Intel® oneAPI DPC++/C++ Compiler, you can choose to switch back to the Microsoft Visual C++ compiler:

1. Select your project in Microsoft Visual Studio.
2. Right-click and select **Intel Compiler > Use Visual C++** from the context menu.

This action updates the solution file to use the Microsoft Visual Studio C++ compiler. All configurations of affected projects are automatically cleaned unless you select **Do not clean project(s)**. If you choose not to clean projects, you will need to rebuild updated projects to ensure all source files are compiled with the new compiler.

Build a Program From the Command Line

Use the following steps to test your compiler installation and build a program.

1. Use a text editor to create a file called `hello-world.cpp` with the following contents:

```
#include <iostream>

int main()
{
    std::cout << "Hello, world!\n";

    return 0;
}
```

2. Compile `hello-world.cpp`:

```
icx hello-world.cpp
```

3. Now you have an executable called `hello-world.exe` which can be run and will give immediate feedback:

```
hello-world.exe
```

   Which outputs:

```
Hello, world!
```

You can direct and control compilation with compiler options. For example, you can create the object file and output the final binary in two steps:

1. Compile `hello-world.cpp`:

```
icx hello-world.cpp /c /Fohello-world.obj
```

   The `/c` option prevents linking at this step and `/Fo` specifies the name for the object file.
2. Use the `icx` compiler to link the resulting application object code and output an executable:

```
icx hello-world.obj /Fehello-world.exe
```

   The `/Fe` option specifies the generated executable file name.

Refer to Compiler Options for details about available options.

Compile and Execute Sample Code

Multiple code samples are provided for the Intel® oneAPI DPC++/C++ Compiler so that you can explore compiler features and familiarize yourself with how it works. For example:

Sample Project	Description
OpenMP Offload Sample	The OpenMP* Offload sample demonstrates some of the new OpenMP Offload features supported by the Intel® oneAPI DPC++/C++ Compiler.

Base: Vector Add Sample	The Vector Add sample is the equivalent of a 'Hello, World!' sample for data parallel programs. Building and running the code sample verifies that your development environment is set up correctly and demonstrates the use of the core features of DPC++.
Matrix Multiply Sample	The Matrix Multiply sample is a simple program that multiplies together two large matrices and verifies the results. This program is implemented in two ways: Using Data Parallel C++ (DPC++) and using OpenMP (OMP).
Adaptive Noise Reduction Sample	The Adaptive Noise Reduction sample is a DPC++ reference design that demonstrates a highly optimized image sensor adaptive noise reduction (ANR) algorithm on an FPGA.

Next Steps

- Use the latest oneAPI Code Samples and follow along with the Intel® oneAPI Training Resources.
- Explore the Intel® oneAPI DPC++/C++ Compiler Developer Guide and Reference on the Intel® Developer Zone.