

# **Get Started with the Intel® oneAPI Base Toolkit for Windows\***

# Contents

## **Chapter 1: Get Started with the Intel® oneAPI Base Toolkit for Windows\***

Configure Your System .....	3
Build and Run a Sample Project Using the Visual Studio* Command Line .....	4
Next Steps .....	9
Build and Run a Sample Project Using Visual Studio* .....	10
Next Steps .....	12
Build and Run a Sample Project Using Visual Studio Code .....	14
Next Steps .....	19
Using Containers .....	21
Using Cloud CI Systems .....	21
Troubleshooting.....	21
Next Steps .....	24
Notices and Disclaimers.....	26

# Get Started with the Intel® oneAPI Base Toolkit for Windows\*

# 1

The following instructions assume you have installed the Intel® oneAPI software. Please see the [Intel oneAPI Toolkits page](#) for installation options.

## Follow These Steps for the Intel® oneAPI Base Toolkit:

1. [Configure Your System.](#)
2. Build and Run a sample project using one of these methods:
  - [Command Line](#)
  - [Visual Studio\\*](#)
3. After you have run a sample, learn more about the Intel® oneAPI Base Toolkit (Base Kit) in [Next Steps](#).

An offline copy of this Get Started is available on the [Downloadable Documentation](#) page.

## Choose Your Toolkit

---

**NOTE** If you are using the Intel® oneAPI Base Toolkit, go to Configure Your System, as described in step 1 above.

---

If you are using a toolkit other than the Base Kit, follow the links below to find steps to configure your system and run a sample project:

- [Get Started with the Intel® HPC Toolkit for Windows](#)
- [Get Started with the Intel® System Bring Up Toolkit for Windows\\*](#)
- [Get Started with the Intel® Rendering Toolkit for Windows\\*](#)

## Configure Your System

### Intel® oneAPI Base Toolkit

- [CPU, GPU and FPGA users, install CMake\\* to build most samples](#)
- [For GPU users, install GPU drivers](#)

### Install CMake\* to Build Samples (optional)

This optional section only applies if you plan to use Microsoft Visual Studio for cross-platform development. Many of the samples require CMake in such a development environment.

CMake is not required to use the oneAPI tools and toolkits. Most of the oneAPI samples that can be built and run on Microsoft Windows\* include a Visual Studio project file to manage the build process.

To build those samples on Windows that require CMake and do not include a Visual Studio project file, you may need to add some additional Visual Studio *workloads* as part of your Visual Studio installation.

The Visual Studio CMake tools for Windows are part of the Desktop development with C++ workload and the Linux Development with C++ workload. Both of these Visual Studio workloads are required for cross-platform CMake development on Windows. To install these workloads, see the following links:

- [Visual Studio 2019](#)
- [Visual Studio 2022](#)

If you installed an Intel® oneAPI toolkit before installing Visual Studio, the oneAPI plug-ins for Visual Studio may be absent. In this case, install Visual Studio and then refer to the [Troubleshooting](#) section of this documentation for instructions on how to fix or add the missing oneAPI plug-ins for Visual Studio.

For more information about CMake, refer to [CMake.org](#).

For Microsoft Windows\*, the Visual Studio\* C++ Development tools must be installed, including the CMake tools in the Desktop development with C++ workload. Installing the Linux Development with C++ workload will not provide the necessary tools to run all samples. Use the links below to install Visual Studio and CMake.

If you already installed an Intel® oneAPI toolkit before installing Visual Studio, the oneAPI plug-ins for Visual Studio will be absent. If this happens, install Visual Studio and CMake using the links below. Then refer to the [Troubleshooting](#) section for instructions on how to fix the plug-ins.

- [Visual Studio 2019](#)
- [Visual Studio 2022](#)
- [CMake.org](#)

Visual C++ Tools for CMake is installed by default as part of the *Desktop Development with C++ workload*.

For more information about CMake, refer to [CMake.org](#).

### For GPU Users, Download and Install GPU Drivers

1. Download the [Intel® Driver & Support Assistant \(Intel® DSA\)](#) by following the link and clicking **Download now**. The Intel DSA tool will help you identify and install the correct driver for your system.
2. Run the Intel® Driver & Support Assistant Application and follow the on-screen prompts to install the latest version of **Intel Graphics - Windows 10 or 11 DCH Drivers**.
3. To troubleshoot any installation issues (or to manually install a driver without the use of the Intel DSA), see these [step-by-step instructions on downloading and installing an Intel® Graphics Driver in Windows® 10 & Windows 11](#).

### Next Step

Run a sample project using one of these methods:

- [Command Line](#)
- [Visual Studio\\*](#)

## Build and Run a Sample Project Using the Visual Studio\* Command Line

### Intel® oneAPI Base Toolkit

**NOTE** If you have not already configured your development environment, go to [Configure Your System](#) then return to this page. If you have already completed the steps to configure your system, continue with the steps below.

Command line development can be done with a terminal window or done through Visual Studio Code\*. Some tasks can be automated using extensions. To learn more, see [Using Visual Studio Code with Intel® oneAPI Toolkits](#).

To compile and run a sample:

1. Locate a sample project using the oneAPI CLI Samples Browser.
2. Build and run a sample project using Microsoft Build\*.

## Download Samples using the oneAPI CLI Samples Browser

Use the oneAPI CLI Samples Browser to browse the collection of online oneAPI samples. As you browse the oneAPI samples, you can copy them to your local disk as buildable sample projects. Most oneAPI sample projects are built using Make or CMake, so the build instructions are included as part of the sample in a README file. The oneAPI CLI utility is a single-file, stand-alone executable that has no dependencies on dynamic runtime libraries.

To see a list of components that support CMake, see [Use CMake with oneAPI Applications](#).

An internet connection is required to download the samples for oneAPI toolkits. For information on how to use this toolkit offline, see [Developing with Offline Systems](#) in the [Troubleshooting](#) section.

To watch a video presentation of how to create a project with the command line, see [Exploring Intel® oneAPI Samples from the Command Line](#).

---

**NOTE** The oneAPI CLI Samples Browser does not work with system proxy settings and does not support WPAD proxy. If you have trouble connecting from behind a proxy, please see [Troubleshooting](#).

---

1. Create a folder where you want to store your sample. For example, C:\samples\vector-add
2. Open a command window.
3. Set system variables by running setvars:

Component Directory Layout:

```
"C:\Program Files (x86)\Intel\oneAPI\setvars.bat"
```

---

**NOTE** For Windows PowerShell\* users, execute this command:

```
cmd.exe "/K" '"C:\Program Files (x86)\Intel\oneAPI\setvars.bat" && powershell'
```

---

Unified Directory Layout:

```
"C:\Program Files (x86)\Intel\oneAPI<toolkit-version>\oneapi-vars.bat"
```

---

**NOTE** For Windows PowerShell\* users, execute this command:

```
cmd.exe "/K" '"C:\Program Files (x86)\Intel\oneAPI<toolkit-version>\oneapi-vars.bat" && powershell'
```

---

The command above assumes you installed to the default folder. If you customized the installation folder, `setvars` | `oneapi-vars` is in your custom folder.

---

**NOTE** The `setvars.bat` script can be managed using a configuration file, which is especially helpful if you need to initialize specific versions of libraries or the compiler, rather than defaulting to the "latest" version. For more details, see [Using a Configuration File to Manage Setvars.bat](#). See [oneAPI Development Environment Setup](#) for more configuration options.

---

4. In the same command window, run the application :

```
oneapi-cli.exe
```

The oneAPI CLI menu appears:

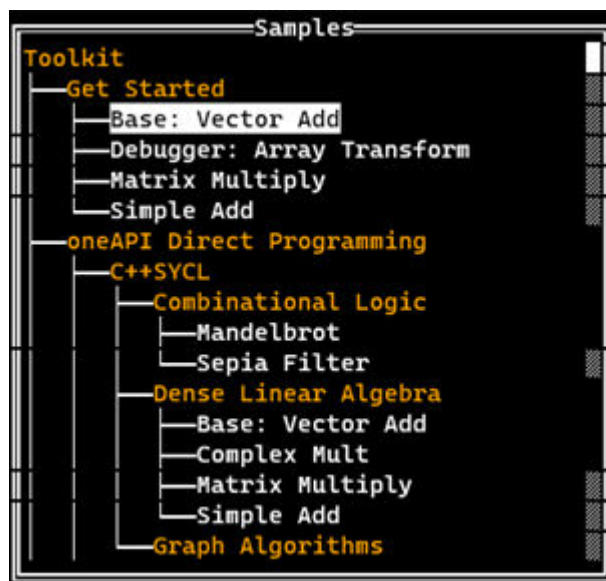
```
(1) Create a project
(2) View oneAPI docs in browser
(q) Quit
```

5. Use the up and down arrow keys to select **Create a project**, then press Enter
6. Move the arrow key down to select **Create a project**, then press Enter. The language selection will appear. If you want to run samples from a toolkit other than the Intel® oneAPI Base Toolkit, install the domain-specific toolkit before proceeding.

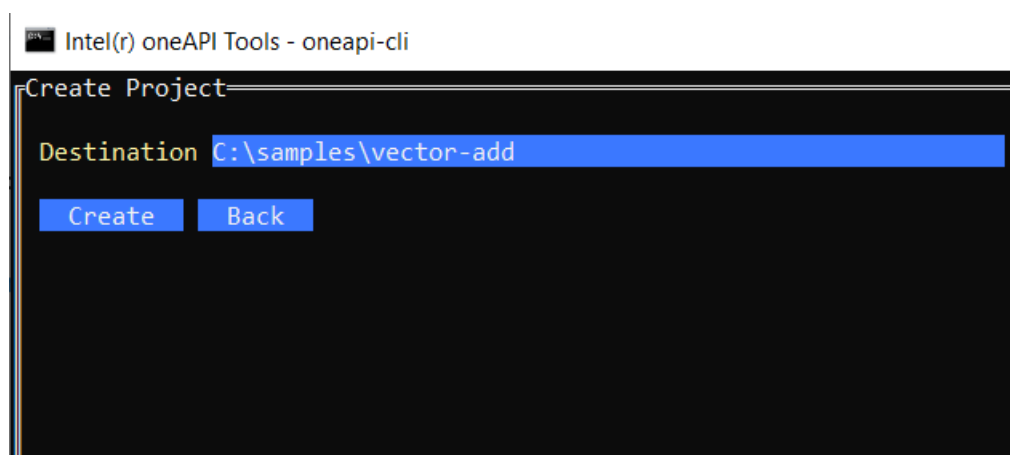
```
-----Select sample language-----
(1) cpp
(2) python
(b) Back
(q) Quit
```

Install a domain-specific toolkit  
for more code samples support

7. Select the language for your sample. For your first project, select **cpp**, then press Enter. The toolkit samples list appears.
8. Select the **Vector Add** sample. Vector Add is a simple test application that will help verify that the tools are setup correctly and can access your system's GPU:



9. After you select a sample, press Enter.
10. Specify the location for the project. The default location includes the path from where the utility was run and the name of the project.
11. Press Tab to select Create, then press Enter:



See [Explore SYCL\\* Through Samples](#) to learn more.

### Build and Run a CPU or GPU Sample Using Microsoft Build\*

1. Using the same command prompt window where you ran `setvars.bat`, navigate to the folder where you downloaded the sample.
2. Configure the project to use the buffer-based implementation (first box) or the Unified Shared Memory (USM) based implementation (second box):

```
mkdir build
cd build
cmake -G "NMake Makefiles" ..
```

```
mkdir build
cd build
cmake -G "NMake Makefiles" .. -DUSM=1
```

3. **NOTE** Some samples require additional steps or arguments for building and/or running the sample. Review the sample's `README.md` file for specific details regarding how to build and run the sample.

Build the program.

```
nmake cpu-gpu
```

4. Navigate to the output directory (example: `x64/Release`)

5. Run the program:

```
vector-add-usm.exe  
vector-add-buffers.exe
```

A success message will appear:

```
c:\oneapi-projects\vectoradd\x64\Release>vector-add-buffers.exe  
Running on device: Intel(R) Core(TM) i7-8665U CPU @ 1.90GHz  
Vector size: 10000  
[0]: 0 + 0 = 0  
[1]: 1 + 1 = 2  
[2]: 2 + 2 = 4  
...  
[9999]: 9999 + 9999 = 19998  
Vector add successfully completed on device.  
  
c:\oneapi-projects\vectoradd\x64\Release>
```

If Running on Device shows a GPU and an error occurs, see the Troubleshooting tip for fixing a [SPIRV Error](#).

1. Optional: Clean the program.

```
nmake clean
```

## Compile and run a sample for FPGA

You can run the `vector-add` sample (or any FPGA SYCL\* code) in the following modes:

- **Emulation:** Verifies the code correctness. Compilation completes in few seconds. Use this mode if you are using the Intel® oneAPI Base Toolkit
- **Report:** Generates a static optimization report for design analysis. Compilation can take a few minutes to complete. When completed, you can find the reports in `<project_name>.prj\reports\report.html`. This can be used with the Intel® oneAPI Base Toolkit. For more information about the reports, refer to the [FPGA Optimization Guide for Intel® oneAPI Toolkits](#).
- **Hardware:** Generates the actual bitstream on an FPGA device. Compilation can take few hours to complete. Use this mode to measure performance. To use this mode, download the Intel® Quartus® Prime Pro Edition software and third-party vendor-provided BSPs separately. For more information, refer to the [Intel® FPGA Add-On for oneAPI Base Toolkit](#) web page.

1. Using the same command prompt window where you ran `setvars.bat`, navigate to the folder where you downloaded the sample.

```
cd <vector-add directory on the same system>
```

2. View the `README.md` file in the sample folder for instructions on how to build and run the sample.

See [Explore SYCL\\* Through Samples](#) to learn more.



## Next Steps

After successfully building a sample application, [Explore SYCL with Samples from Intel](#) and explore the tools in the Intel® oneAPI Base Toolkit.

Tool	Description
Intel® DPC++ Compatibility Tool	The Intel® DPC++ Compatibility Tool assists in migration of CUDA* applications to SYCL* ready code that can use the Intel® oneAPI DPC++/C++ Compiler. <a href="#">Get started.</a>
Intel® oneAPI DPC++/C++ Compiler	The Intel® oneAPI DPC++/C++ Compiler targets CPUs and accelerators through single-source code while permitting custom tuning. <a href="#">Get Started.</a>
Intel® oneAPI DPC++ Library	This library is a companion to the Intel® oneAPI DPC++/C++ Compiler and provides an alternative for C++ developers who create heterogeneous applications and solutions. Its APIs are based on familiar standards-C++ STL, Parallel STL (PSTL), Boost.Compute, and SYCL*-to maximize productivity and performance across CPUs, GPUs, and FPGAs.
Intel® Distribution for GDB*	GDB, the GNU Project debugger, allows you to see what is going on 'inside' another program while it executes -- or what another program was doing at the moment it crashed. <a href="#">Get Started.</a>  <a href="#">Learn more.</a>
Intel® oneAPI Math Kernel Library (oneMKL)	The oneMKL helps you achieve maximum performance with a math computing library of highly optimized, extensively parallelized routines for CPU and GPU. The library has C and Fortran interfaces for most routines on CPU, and SYCL* interfaces for some routines on both CPU and GPU. <a href="#">Get started.</a>
Intel® oneAPI Threading Building Blocks (oneTBB)	oneTBB is a flexible performance library that simplifies the work of adding parallelism to complex applications, even if you're not a threading expert. <a href="#">Learn more.</a>
Intel® Integrated Performance Primitives	Intel® Integrated Performance Primitives (Intel® IPP) is an extensive library of ready-to-use, domain-specific functions that are highly optimized for diverse Intel® architectures. <a href="#">Get Started with Intel® IPP</a> or <a href="#">Get Started with Intel® IPP Cryptography.</a>
Intel® oneAPI Data Analytics Library	The Intel® oneAPI Data Analytics Library (oneDAL) is a library that helps speed up big data analysis by providing highly optimized algorithmic building blocks for all stages of data analytics (preprocessing, transformation, analysis, modeling, validation, and decision making) in batch, online, and distributed processing modes of computation. The current version of oneDAL provides SYCL* API extensions to the traditional C++ interface. <a href="#">Get started.</a>
Intel® oneAPI Deep Neural Network Library	The Intel® oneAPI Deep Neural Network Library (oneDNN) is an open-source performance library for deep learning applications. The library includes basic building blocks for neural networks optimized for Intel® Architecture Processors and Intel® Processor Graphics. oneDNN is intended for deep learning applications and framework developers interested in improving application performance on Intel CPUs and GPUs. <a href="#">Get started.</a>
Intel® VTune™ Profiler	Intel® VTune™ Profiler is a performance analysis tool targeted for users developing serial and multithreaded applications. The tool is delivered as

Intel® Advisor	a Performance Profiler with Intel Performance Snapshots and supports local and remote target analysis on the Windows* and Linux* platforms. <a href="#">Get started.</a>
Intel® FPGA Add-on for oneAPI Base Toolkit	Intel® Advisor gives software architects and developers the data and analysis tools they need to build well-threaded and vectorized code that exploits modern hardware capabilities. <a href="#">Get started.</a> Use reconfigurable hardware to accelerate data-centric workloads. <a href="#">Learn more.</a>

Learn more about SYCL and targeting other accelerators using the following resources:

Resource	Description
<a href="#">Intel® oneAPI Programming Guide</a>	Provides details on the oneAPI programming model, including details about SYCL, programming for various target accelerators, and introductions to the oneAPI libraries.
<a href="#">FPGA Optimization Guide for Intel® oneAPI Toolkits</a>	The oneAPI FPGA Optimization Guide provides guidance on leveraging the functionalities of SYCL to optimize your design.
<a href="#">Explore SYCL Through Intel® FPGA Code Samples</a>	Provides guidance on how to target and develop your design on an FPGA using the oneAPI programming model. It also provides guidance on how to optimize a design to achieve performance and latency targets for an application targeting the FPGA.
<a href="#">Intel® FPGA SDK for OpenCL™ Pro Edition: Custom Platform Toolkit User Guide</a>	Outlines the procedure for creating an Intel® FPGA Software Development Kit (SDK) for OpenCL™ Pro Edition Custom Platform. The Intel® FPGA SDK for OpenCL™ Pro Edition Custom Platform Toolkit provides the necessary tools for implementing a fully functional Custom Platform.
<a href="#">oneAPI GPU Optimization Guide</a>	The oneAPI GPU Optimization Guide demonstrates how to improve the behavior of your software by partitioning it across the host and accelerator to specialize portions of the computation that run best on the accelerator. Specialization includes restructuring and tuning the code to create the best mapping of the application to the hardware. The value of oneAPI is that it allows each of these variations to be expressed in a common language with device-specific variants launched on the appropriate accelerator.

For more information about this toolkit, see the [Intel® oneAPI Base Toolkit](#) page.

## Build and Run a Sample Project Using Visual Studio\*

### Intel® oneAPI Base Toolkit

Intel® oneAPI integrates with third-party IDEs on Windows\* to provide a seamless GUI experience for software development.

**NOTE** If you are using Visual Studio\* with FPGA, see the [FPGA Workflows on Third-Party IDEs for Intel® oneAPI Toolkits](#).

#### NOTE

An internet connection is required to download the samples for oneAPI toolkits. For information on how to use this toolkit offline, see [Developing with Offline Systems](#) in the Troubleshooting section.

You can use the Intel® oneAPI DPC++/C++ Compiler within the Microsoft Visual Studio\* integrated development environment (IDE) to develop C++ applications, including static library (.LIB), dynamic link library (.DLL), and main executable (.EXE) applications. This environment makes it easy to create, debug, and execute programs. You can build your source code into several types of programs and libraries using the IDE or from the command line.

The IDE offers these major advantages:

- Makes application development quicker and easier by providing a visual development environment.
- Provides integration with the native Microsoft Visual Studio\* debugger.
- Makes other IDE tools available.

### Define the SETVARS\_CONFIG Environment Variable:

The SETVARS\_CONFIG environment variable is not automatically defined during installation, you must add it to your environment before starting Visual Studio using the following. This only needs to be set once.

1. Open a command window.
2. Set system variables for Visual Studio:

```
. setx SETVARS_CONFIG " "
```

### Create a Project Using Microsoft Visual Studio\*

---

**NOTE** If you are using Visual Studio 2019, samples will only work with version 16.4.0 and later.

---

To watch a video presentation of how to create a project, see [Intel® oneAPI Visual Studio Samples Browser](#).

1. Open Microsoft Visual Studio\*.
2. For Visual Studio 2019 and 2022, a page may display showing recent projects. Click **Continue without code**.

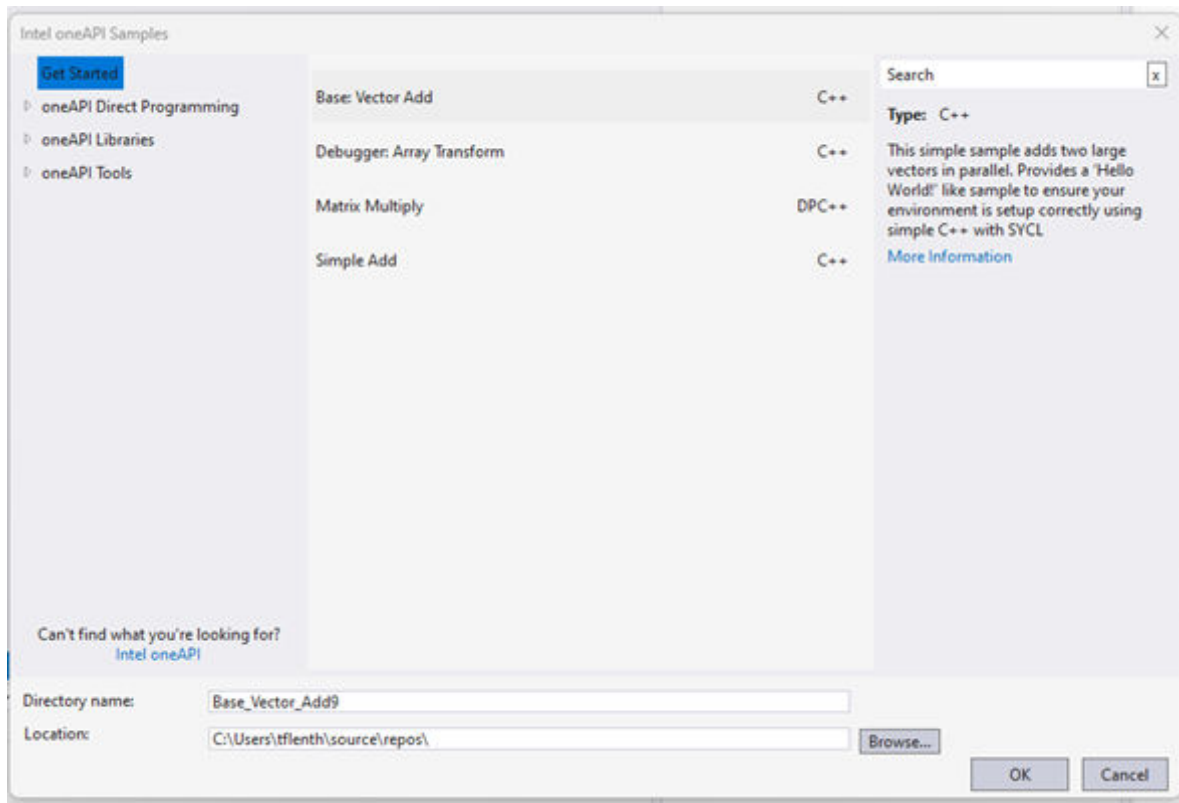
---

**NOTE** In the next step, you will look for a menu named **Extensions > Intel** or **Intel**. If you do not see either of these menu choices, then the plug-ins have not been installed. See [Troubleshooting](#) to fix the plug-ins.

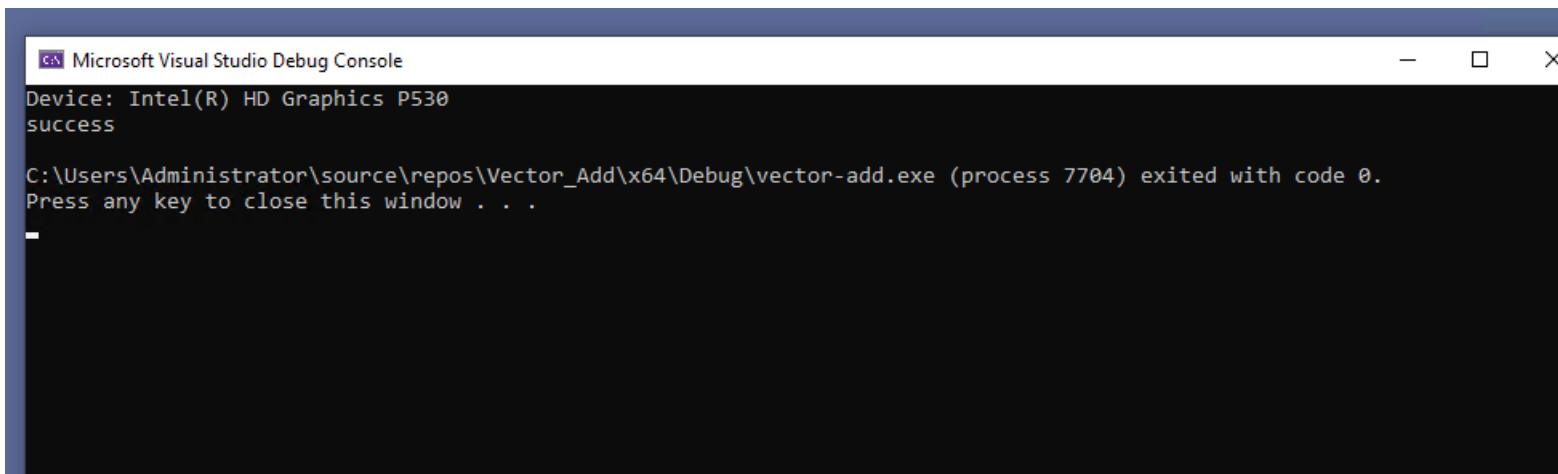
---

3. From the top menu, select:
  - Visual Studio 2019 and 2022: **Extensions > Intel > Browse an Intel oneAPI Sample**

The samples screen will open:



4. In the center area, select Vector Add. Vector Add is a simple test application that will help verify that the tools are setup correctly and can access your system's GPU.
5. Click OK.
6. From the Solution Explorer, right-click on vector-add and select **Rebuild**.
7. After the program is built, click **Debug** > **Start Without Debugging**. The results will display:



See [Explore SYCL Through Samples](#) to learn more.

## Next Steps

After successfully building a sample application, [Explore SYCL with Samples from Intel](#) and explore the tools in the Intel® oneAPI Base Toolkit.

Tool	Description
Intel® DPC++ Compatibility Tool	The Intel® DPC++ Compatibility Tool assists in migration of CUDA* applications to SYCL* ready code that can use the Intel® oneAPI DPC++/C++ Compiler. <a href="#">Get started.</a>
Intel® oneAPI DPC++/C++ Compiler	The Intel® oneAPI DPC++/C++ Compiler targets CPUs and accelerators through single-source code while permitting custom tuning. <a href="#">Get Started.</a>
Intel® oneAPI DPC++ Library	This library is a companion to the Intel® oneAPI DPC++/C++ Compiler and provides an alternative for C++ developers who create heterogeneous applications and solutions. Its APIs are based on familiar standards-C++ STL, Parallel STL (PSTL), Boost.Compute, and SYCL*-to maximize productivity and performance across CPUs, GPUs, and FPGAs.
Intel® Distribution for GDB*	GDB, the GNU Project debugger, allows you to see what is going on 'inside' another program while it executes -- or what another program was doing at the moment it crashed. <a href="#">Get Started.</a>  <a href="#">Learn more.</a>
Intel® oneAPI Math Kernel Library (oneMKL)	The oneMKL helps you achieve maximum performance with a math computing library of highly optimized, extensively parallelized routines for CPU and GPU. The library has C and Fortran interfaces for most routines on CPU, and SYCL* interfaces for some routines on both CPU and GPU. <a href="#">Get started.</a>
Intel® oneAPI Threading Building Blocks (oneTBB)	oneTBB is a flexible performance library that simplifies the work of adding parallelism to complex applications, even if you're not a threading expert. <a href="#">Learn more.</a>
Intel® Integrated Performance Primitives	Intel® Integrated Performance Primitives (Intel® IPP) is an extensive library of ready-to-use, domain-specific functions that are highly optimized for diverse Intel® architectures. <a href="#">Get Started with Intel® IPP</a> or <a href="#">Get Started with Intel® IPP Cryptography.</a>
Intel® oneAPI Data Analytics Library	The Intel® oneAPI Data Analytics Library (oneDAL) is a library that helps speed up big data analysis by providing highly optimized algorithmic building blocks for all stages of data analytics (preprocessing, transformation, analysis, modeling, validation, and decision making) in batch, online, and distributed processing modes of computation. The current version of oneDAL provides SYCL* API extensions to the traditional C++ interface. <a href="#">Get started.</a>
Intel® oneAPI Deep Neural Network Library	The Intel® oneAPI Deep Neural Network Library (oneDNN) is an open-source performance library for deep learning applications. The library includes basic building blocks for neural networks optimized for Intel® Architecture Processors and Intel® Processor Graphics. oneDNN is intended for deep learning applications and framework developers interested in improving application performance on Intel CPUs and GPUs. <a href="#">Get started.</a>
Intel® VTune™ Profiler	Intel® VTune™ Profiler is a performance analysis tool targeted for users developing serial and multithreaded applications. The tool is delivered as a Performance Profiler with Intel Performance Snapshots and supports local and remote target analysis on the Windows* and Linux* platforms. <a href="#">Get started.</a>

Intel® Advisor	Intel® Advisor gives software architects and developers the data and analysis tools they need to build well-threaded and vectorized code that exploits modern hardware capabilities. <a href="#">Get started.</a>
Intel® FPGA Add-on for oneAPI Base Toolkit	Use reconfigurable hardware to accelerate data-centric workloads. <a href="#">Learn more.</a>

Learn more about SYCL and targeting other accelerators using the following resources:

Resource	Description
<a href="#">Intel® oneAPI Programming Guide</a>	Provides details on the oneAPI programming model, including details about SYCL, programming for various target accelerators, and introductions to the oneAPI libraries.
<a href="#">FPGA Optimization Guide for Intel® oneAPI Toolkits</a>	The oneAPI FPGA Optimization Guide provides guidance on leveraging the functionalities of SYCL to optimize your design.
<a href="#">Explore SYCL Through Intel® FPGA Code Samples</a>	Provides guidance on how to target and develop your design on an FPGA using the oneAPI programming model. It also provides guidance on how to optimize a design to achieve performance and latency targets for an application targeting the FPGA.
<a href="#">Intel® FPGA SDK for OpenCL™ Pro Edition: Custom Platform Toolkit User Guide</a>	Outlines the procedure for creating an Intel® FPGA Software Development Kit (SDK) for OpenCL™ Pro Edition Custom Platform. The Intel® FPGA SDK for OpenCL™ Pro Edition Custom Platform Toolkit provides the necessary tools for implementing a fully functional Custom Platform.
<a href="#">oneAPI GPU Optimization Guide</a>	The oneAPI GPU Optimization Guide demonstrates how to improve the behavior of your software by partitioning it across the host and accelerator to specialize portions of the computation that run best on the accelerator. Specialization includes restructuring and tuning the code to create the best mapping of the application to the hardware. The value of oneAPI is that it allows each of these variations to be expressed in a common language with device-specific variants launched on the appropriate accelerator.

For more information about this toolkit, see the [Intel® oneAPI Base Toolkit](#) page.

## Build and Run a Sample Project Using Visual Studio Code

### Intel® oneAPI Base Toolkit

Intel® oneAPI toolkits integrate with third-party IDEs to provide a seamless GUI experience for software development.

**NOTE** If you are using Visual Studio Code (VS Code) with FPGA, see the [FPGA Workflows on Third-Party IDEs for Intel® oneAPI Toolkits](#).

To see a list of components that support CMake, see [Use CMake with oneAPI Applications](#).

To watch a video presentation of how to install extensions and use them to set up your environment, explore sample code, and connect to the Intel® Developer Cloud using Visual Studio Code, see [oneAPI Visual Studio Code Extensions](#).

This procedure requires the Sample Browser extension to be installed. The next section will describe how to install it. If you have already installed it, skip to [Create a Project Using Visual Studio Code](#).

## Extensions for Visual Studio Code Users

To watch a video presentation of how to install extensions and use them to set up your environment, explore sample code, and connect to the Intel® Developer Cloud using Visual Studio Code, see [oneAPI Visual Studio Code Extensions](#).

You can use VS Code extensions to set your environment, create launch configurations, and browse and download samples:

1. From Visual Studio Code, click on the Extensions logo in the left navigation.





2. Locate the extension titled **Extension Pack for Intel® oneAPI Toolkits**, or visit <https://marketplace.visualstudio.com/publishers/intel-corporation> to browse available extensions.
3. Click **Install**.

For more information about VS Code extensions for Intel oneAPI Toolkits, see [Using Visual Studio Code\\* to Develop Intel® oneAPI Applications](#).

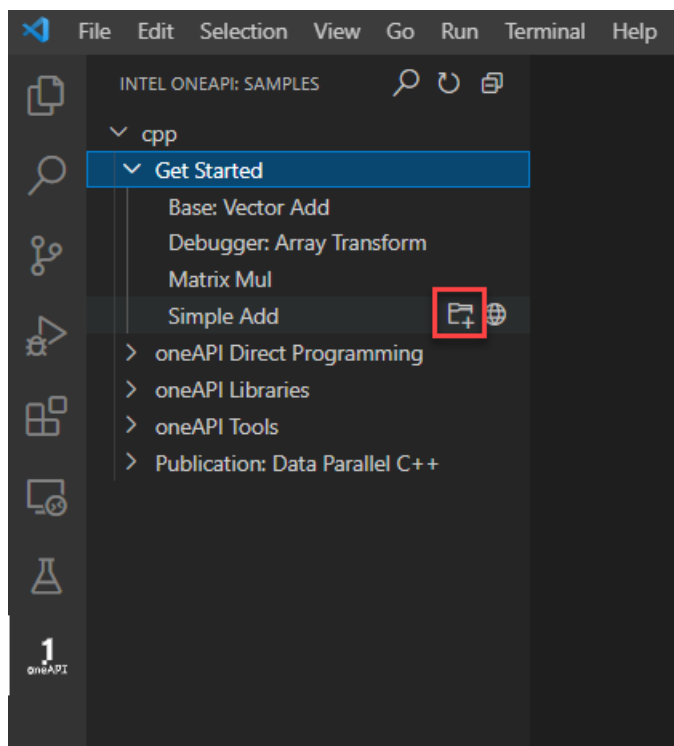
## Create a Project Using Visual Studio Code

1. Click on the icon to open the oneAPI Samples Browser:

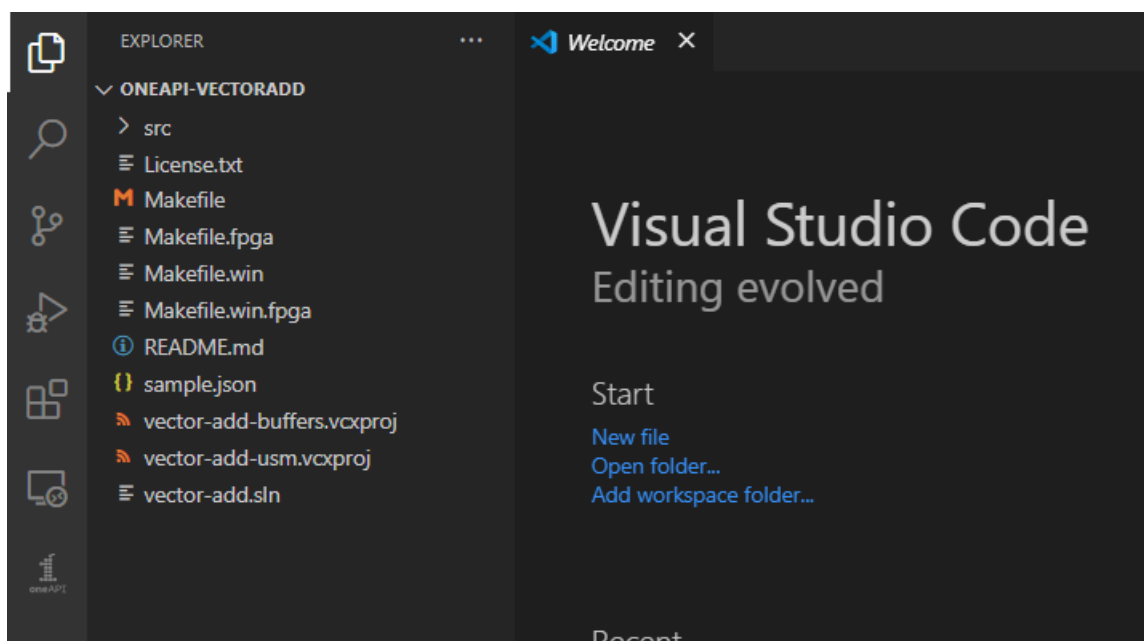


2. A list of available samples will open in the left navigation.
3. To view the readme for the sample, click the  next to the sample. If you choose to build and run the sample, the readme will also be downloaded with the sample.
4. Find the sample you want to build and run. Click the  to the right of the sample name.





5. Create a new folder for your sample and click OK.
6. The sample will load in a new window:



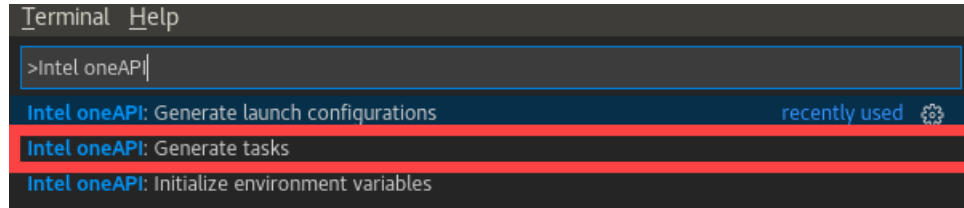
## Set the oneAPI Environment

1. Press **Ctrl+Shift+P** ( or **View -> Command Palette...** ) to open the Command Palette.
2. Type **Intel oneAPI: Initialize environment variables**. Click on **Intel oneAPI: Initialize environment variables**.

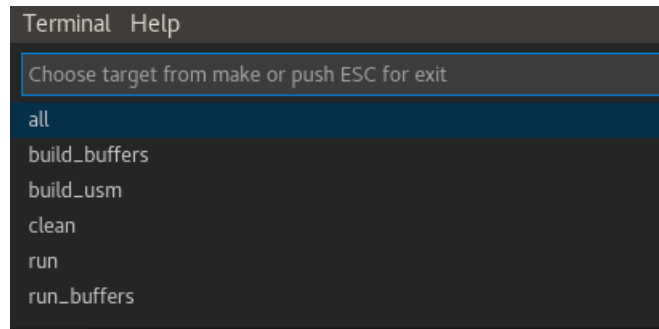


## Prepare Build Tasks from Make / CMake Files

1. Press **Ctrl+Shift+P** or **View -> Command Palette...** to open the Command Palette.
2. Type **Intel oneAPI** and select **Intel oneAPI: Generate tasks**.



3. Select the build tasks (target) from your Make/CMake oneAPI project that you want to use.



4. Run the task/target by selecting **Terminal -> Run task...**
5. Select the task to run.

**NOTE** Not all oneAPI sample projects use CMake. The `README.md` file for each sample specifies how to build the sample. We recommend that you check out the [CMake extension for VS Code](#) that is maintained by Microsoft.

## Build the Project

The oneAPI extensions enable the ability to prepare launch configurations for running and debugging projects created using Intel oneAPI toolkits:

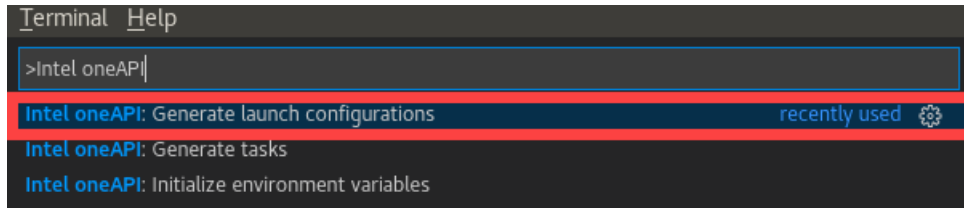
**NOTE** Debugging a local Windows host using VS Code is not supported. Debugging a local Linux host on CPU is supported using VS Code, but debugging on GPU requires a remote host. Debugging a remote Linux target from Windows or Linux host is supported using VS Code.

1. Press **Ctrl+Shift+B** or **Terminal -> Run Build Task...** to set the default build task.
2. Select the task from the command prompt list to build your project.
3. Press **Ctrl+Shift+B** or **Terminal -> Run Build Task...** again to build your project.

## Prepare Launch Configuration for Debugging

The oneAPI extensions enable the ability to prepare launch configurations for running and debugging projects created using Intel oneAPI toolkits:

1. Press **Ctrl+Shift+P** or **View -> Command Palette...** to open the Command Palette.
2. Type **Intel oneAPI** and select **Intel oneAPI: Generate launch configurations**.



3. Select the executable (target) you want to debug.  
Optional: select any task you want to run before and/or after launching the debugger (for example, build the project before debug, clean the project after debug).
4. The configuration is now available to debug and run using the gdb-oneapi debugger. You can find it in `.vscode/launch.json`. To debug and run, click on the Run icon or press **Ctrl+Shift+D**.

**NOTE** Debugging a local Windows host using VS Code is not supported. Debugging a local Linux host on CPU is supported using VS Code, but debugging on GPU requires a remote host. Debugging a remote Linux target from Windows or Linux host is supported using VS Code.

## Debug, Analyze, Develop with More Extensions

There are more oneAPI extensions for Visual Studio Code which enable:

- debugging
- remote development
- connection to Intel Developer Cloud
- analysis configuration

To learn more about the extensions, see [Intel® oneAPI Extensions for Visual Studio Code\\*](#).

To learn more about more capabilities and options, see [Using Visual Studio Code with Intel® oneAPI Toolkits](#).

### NOTE

An internet connection is required to download the samples for oneAPI toolkits. If you are using an offline system, download the samples from a system that is internet connected and transfer the sample files to your offline system. If you are using an IDE for development, you will not be able to use the oneAPI CLI Samples Browser while you are offline. Instead, download the samples and extract them to a directory. Then open the sample with your IDE. The samples can be downloaded from here: [Intel® oneAPI Toolkit Code Samples](#)

See [Explore SYCL\\* Through Samples](#) to learn more.

## Next Steps

After successfully building a sample application, [Explore SYCL with Samples from Intel](#) and explore the tools in the Intel® oneAPI Base Toolkit.

Tool	Description
------	-------------

## Intel® DPC++ Compatibility Tool

The Intel® DPC++ Compatibility Tool assists in migration of CUDA\* applications to SYCL\* ready code that can use the Intel® oneAPI DPC++/C++ Compiler. [Get started.](#)

## Intel® oneAPI DPC++/C++ Compiler

The Intel® oneAPI DPC++/C++ Compiler targets CPUs and accelerators through single-source code while permitting custom tuning. [Get Started.](#)

## Intel® oneAPI DPC++ Library

This library is a companion to the Intel® oneAPI DPC++/C++ Compiler and provides an alternative for C++ developers who create heterogeneous applications and solutions. Its APIs are based on familiar standards-C++ STL, Parallel STL (PSTL), Boost.Compute, and SYCL\*-to maximize productivity and performance across CPUs, GPUs, and FPGAs.

## Intel® Distribution for GDB\*

GDB, the GNU Project debugger, allows you to see what is going on 'inside' another program while it executes -- or what another program was doing at the moment it crashed. [Get Started.](#)

[Learn more.](#)

## Intel® oneAPI Math Kernel Library (oneMKL)

The oneMKL helps you achieve maximum performance with a math computing library of highly optimized, extensively parallelized routines for CPU and GPU. The library has C and Fortran interfaces for most routines on CPU, and SYCL\* interfaces for some routines on both CPU and GPU. [Get started.](#)

## Intel® oneAPI Threading Building Blocks (oneTBB)

oneTBB is a flexible performance library that simplifies the work of adding parallelism to complex applications, even if you're not a threading expert. [Learn more.](#)

## Intel® Integrated Performance Primitives

Intel® Integrated Performance Primitives (Intel® IPP) is an extensive library of ready-to-use, domain-specific functions that are highly optimized for diverse Intel® architectures. [Get Started with Intel® IPP](#) or [Get Started with Intel® IPP Cryptography.](#)

## Intel® oneAPI Data Analytics Library

The Intel® oneAPI Data Analytics Library (oneDAL) is a library that helps speed up big data analysis by providing highly optimized algorithmic building blocks for all stages of data analytics (preprocessing, transformation, analysis, modeling, validation, and decision making) in batch, online, and distributed processing modes of computation. The current version of oneDAL provides SYCL\* API extensions to the traditional C++ interface. [Get started.](#)

## Intel® oneAPI Deep Neural Network Library

The Intel® oneAPI Deep Neural Network Library (oneDNN) is an open-source performance library for deep learning applications. The library includes basic building blocks for neural networks optimized for Intel® Architecture Processors and Intel® Processor Graphics. oneDNN is intended for deep learning applications and framework developers interested in improving application performance on Intel CPUs and GPUs. [Get started.](#)

## Intel® VTune™ Profiler

Intel® VTune™ Profiler is a performance analysis tool targeted for users developing serial and multithreaded applications. The tool is delivered as a Performance Profiler with Intel Performance Snapshots and supports local and remote target analysis on the Windows\* and Linux\* platforms. [Get started.](#)

## Intel® Advisor

Intel® Advisor gives software architects and developers the data and analysis tools they need to build well-threaded and vectorized code that exploits modern hardware capabilities. [Get started.](#)

Intel® FPGA Add-on for oneAPI Base Toolkit

Use reconfigurable hardware to accelerate data-centric workloads. [Learn more.](#)

Learn more about SYCL and targeting other accelerators using the following resources:

Resource	Description
<a href="#">Intel® oneAPI Programming Guide</a>	Provides details on the oneAPI programming model, including details about SYCL, programming for various target accelerators, and introductions to the oneAPI libraries.
<a href="#">FPGA Optimization Guide for Intel® oneAPI Toolkits</a>	The oneAPI FPGA Optimization Guide provides guidance on leveraging the functionalities of SYCL to optimize your design.
<a href="#">Explore SYCL Through Intel® FPGA Code Samples</a>	Provides guidance on how to target and develop your design on an FPGA using the oneAPI programming model. It also provides guidance on how to optimize a design to achieve performance and latency targets for an application targeting the FPGA.
<a href="#">Intel® FPGA SDK for OpenCL™ Pro Edition: Custom Platform Toolkit User Guide</a>	Outlines the procedure for creating an Intel® FPGA Software Development Kit (SDK) for OpenCL™ Pro Edition Custom Platform. The Intel® FPGA SDK for OpenCL™ Pro Edition Custom Platform Toolkit provides the necessary tools for implementing a fully functional Custom Platform.
<a href="#">oneAPI GPU Optimization Guide</a>	The oneAPI GPU Optimization Guide demonstrates how to improve the behavior of your software by partitioning it across the host and accelerator to specialize portions of the computation that run best on the accelerator. Specialization includes restructuring and tuning the code to create the best mapping of the application to the hardware. The value of oneAPI is that it allows each of these variations to be expressed in a common language with device-specific variants launched on the appropriate accelerator.

For more information about this toolkit, see the [Intel® oneAPI Base Toolkit](#) page.

## Using Containers

### Intel® oneAPI Base Toolkit

oneAPI containers are not officially supported or tested on Windows\* host systems. However, the instructions in the [Using Containers](#) section of the [Get Started with the Intel® oneAPI Base Toolkit for Linux\\*](#) should apply to a Windows host. There are known issues regarding access to the GPU on a Windows host.

## Using Cloud CI Systems

Cloud CI systems allow you to build and test your software automatically. See the [repo in github](#) for examples of configuration files that use oneAPI for the popular cloud CI systems.

## Troubleshooting

*Potential errors and how to avoid or fix them.*

Issue	How to fix
Errors that occur during installation or directly after installation.	See the <a href="#">Troubleshooting page of the Intel® oneAPI Toolkits Installation Guide for Windows* OS.</a>

Issue	How to fix
<p>Undefined error appears when building a sample.</p>	<p>If the sample was built by using <code>cmake</code> and then <code>make</code>, more details to diagnose the reason for your error are available by rebuilding with the <code>VERBOSE</code> option:</p> <pre data-bbox="824 390 1442 420">make VERBOSE=1</pre> <p>For more comprehensive troubleshooting, use the Diagnostics Utility for Intel® oneAPI Toolkits, which provides system checks to find missing dependencies and permissions errors. <a href="#">Learn more.</a></p>
<p>Developing with Offline Systems: I need to develop on a system that is not connected to the internet.</p>	<p>If you are using an offline system, download the samples from a system that is internet connected and transfer the sample files to your offline system.</p> <p>After you have downloaded the samples, follow the instructions in the <code>README.md</code> file. The readme file is located in the folder of the sample you are interested in.</p> <p>The samples can be downloaded from the <a href="#">code samples repository</a>.</p> <p>The full set of documentation can be downloaded from <a href="#">Downloadable Documentation</a>.</p>
<p>Errors due to missing dependencies, missing environment variables or missing machine capabilities.</p>	<p>The Diagnostics Utility for Intel oneAPI Toolkits provides the ability to find missing dependencies and permissions errors and is already installed with this toolkit. <a href="#">Learn more.</a></p>
<p>Unable to install toolkits or access libraries.</p>	<p>Verify that you meet the System Requirements listed on the <a href="#">Intel® oneAPI Base Toolkit product page</a></p>
<p>When building a sample on Windows, error MSB8020: compiler not found appears.</p>	<p>When compiling, this error appears:</p> <p>error MSB8020: The build tools for Intel® oneAPI DPC++ Compiler (Platform Toolset = 'Intel® oneAPI DPC++ Compiler') cannot be found. To build using the Intel® oneAPI DPC++ Compiler build tools, please install Intel® oneAPI DPC++ Compiler build tools. Alternatively, you may upgrade to the current Visual Studio tools by selecting the Project menu or right-click the solution, and then select "Retarget solution."</p> <p>This issue can occur if Visual Studio is installed after a oneAPI Toolkit, resulting in the toolkit plug-ins being absent.</p> <p>To fix this problem, uninstall and reinstall your oneAPI toolkits:</p> <ol style="list-style-type: none"> <li>1. In Windows settings, open Add or Remove Programs.</li> <li>2. Click on Intel® oneAPI toolkits.</li> <li>3. Click <b>Uninstall</b>.</li> <li>4. After the Uninstall process is complete, verify that Visual Studio and/or MSBuild are installed.</li> <li>5. Install your oneAPI toolkits following the same procedure you used the first time. To download the software, go to the <a href="#">Intel® oneAPI Toolkits page</a>.</li> </ol>

Issue	How to fix
<p>When running a sample on Windows, SPIRV internal error appears.</p>	<p>The SPIRV error could be occurring if the default processor is a GPU but you are running a sample for a CPU. To force the sample to compile on the CPU:</p> <ol style="list-style-type: none"> <li>1. In the src file for the sample, open the .cpp file. For example, C:\samples\vector-add\src\vector-add-buffers.cpp.</li> <li>2. Find the #else statement that starts with //The default device selector will select the most performant device.</li> <li>3. Replace the entire #else #endif statement with: <pre data-bbox="824 638 1442 814">#else // The default device selector will select the most performant device. // default_selector d_selector; cpu_selector d_selector; #endif</pre> </li> <li>4. Compile the sample again, then run the sample.</li> </ol>
<p>In Visual Studio, the Intel menu does not appear and/or the samples from the installed toolkit are missing.</p>	<p>If Visual Studio is installed after a oneAPI toolkit, the plug-ins that come as part of the toolkit are not installed.</p> <p>To fix this problem, uninstall and reinstall your oneAPI toolkits:</p> <ol style="list-style-type: none"> <li>1. In Windows settings, open Add or Remove Programs.</li> <li>2. Select on Intel® oneAPI toolkits.</li> <li>3. Click <b>Uninstall</b>.</li> <li>4. After the Uninstall process is complete, verify that Visual Studio 2017 and/or Visual Studio 2019 is installed.</li> <li>5. Install your oneAPI toolkits following the same procedure you used the first time. The software is available for download at the <a href="#">Intel® oneAPI Toolkits page</a>.</li> </ol>
<p>Problems connecting from behind a proxy</p>	<p>The oneAPI CLI Samples Browser does not work with system proxy settings and does not support WPAD proxy.</p> <p>If you are unable to access the samples from the oneAPI CLI Samples Browser, set the value of https_proxy and http_proxy, then run oneapi-cli.exe in the same command line window. To set the proxy enter this command, where your.proxy is the name of your proxy server and 8080 is your default port.</p> <pre data-bbox="824 1701 1442 1759">set http_proxy=http://your.proxy:8080 set https_proxy=https://your.proxy:8080</pre> <p>After you set the proxy, return to the instructions for Running a Sample Using the Command Line and try again.</p>

Issue	How to fix
Long-running applications are terminated.	<p>If your application has long-running GPU compute workloads in native environments, those workloads may be terminated by the hardware. This functionality is intended to prevent process hangs to run indefinitely; however, this behavior is not recommended for virtualizations or other standard usages of GPU, such as gaming.</p> <p>A workload that takes more than four seconds for GPU hardware to execute is considered a long-running workload. By default, individual threads that qualify as long-running workloads are considered hung and are terminated.</p> <p>To modify the termination process, see <a href="#">Timeout Detection and Recovery (TDR) Registry Keys</a> from Microsoft.</p>

## Next Steps

After successfully building a sample application, [Explore SYCL with Samples from Intel](#) and explore the tools in the Intel® oneAPI Base Toolkit.

Tool	Description
Intel® DPC++ Compatibility Tool	The Intel® DPC++ Compatibility Tool assists in migration of CUDA* applications to SYCL* ready code that can use the Intel® oneAPI DPC++/C++ Compiler. <a href="#">Get started.</a>
Intel® oneAPI DPC++/C++ Compiler	The Intel® oneAPI DPC++/C++ Compiler targets CPUs and accelerators through single-source code while permitting custom tuning. <a href="#">Get Started.</a>
Intel® oneAPI DPC++ Library	This library is a companion to the Intel® oneAPI DPC++/C++ Compiler and provides an alternative for C++ developers who create heterogeneous applications and solutions. Its APIs are based on familiar standards-C++ STL, Parallel STL (PSTL), Boost.Compute, and SYCL*-to maximize productivity and performance across CPUs, GPUs, and FPGAs.
Intel® Distribution for GDB*	GDB, the GNU Project debugger, allows you to see what is going on 'inside' another program while it executes -- or what another program was doing at the moment it crashed. <a href="#">Get Started.</a>  <a href="#">Learn more.</a>
Intel® oneAPI Math Kernel Library (oneMKL)	The oneMKL helps you achieve maximum performance with a math computing library of highly optimized, extensively parallelized routines for CPU and GPU. The library has C and Fortran interfaces for most routines on CPU, and SYCL* interfaces for some routines on both CPU and GPU. <a href="#">Get started.</a>
Intel® oneAPI Threading Building Blocks (oneTBB)	oneTBB is a flexible performance library that simplifies the work of adding parallelism to complex applications, even if you're not a threading expert. <a href="#">Learn more.</a>



Intel® Integrated Performance Primitives	Intel® Integrated Performance Primitives (Intel® IPP) is an extensive library of ready-to-use, domain-specific functions that are highly optimized for diverse Intel® architectures. <a href="#">Get Started with Intel® IPP</a> or <a href="#">Get Started with Intel® IPP Cryptography</a> .
Intel® oneAPI Data Analytics Library	The Intel® oneAPI Data Analytics Library (oneDAL) is a library that helps speed up big data analysis by providing highly optimized algorithmic building blocks for all stages of data analytics (preprocessing, transformation, analysis, modeling, validation, and decision making) in batch, online, and distributed processing modes of computation. The current version of oneDAL provides SYCL* API extensions to the traditional C++ interface. <a href="#">Get started</a> .
Intel® oneAPI Deep Neural Network Library	The Intel® oneAPI Deep Neural Network Library (oneDNN) is an open-source performance library for deep learning applications. The library includes basic building blocks for neural networks optimized for Intel® Architecture Processors and Intel® Processor Graphics. oneDNN is intended for deep learning applications and framework developers interested in improving application performance on Intel CPUs and GPUs. <a href="#">Get started</a> .
Intel® VTune™ Profiler	Intel® VTune™ Profiler is a performance analysis tool targeted for users developing serial and multithreaded applications. The tool is delivered as a Performance Profiler with Intel Performance Snapshots and supports local and remote target analysis on the Windows* and Linux* platforms. <a href="#">Get started</a> .
Intel® Advisor	Intel® Advisor gives software architects and developers the data and analysis tools they need to build well-threaded and vectorized code that exploits modern hardware capabilities. <a href="#">Get started</a> .
Intel® FPGA Add-on for oneAPI Base Toolkit	Use reconfigurable hardware to accelerate data-centric workloads. <a href="#">Learn more</a> .

Learn more about SYCL and targeting other accelerators using the following resources:

Resource	Description
<a href="#">Intel® oneAPI Programming Guide</a>	Provides details on the oneAPI programming model, including details about SYCL, programming for various target accelerators, and introductions to the oneAPI libraries.
<a href="#">FPGA Optimization Guide for Intel® oneAPI Toolkits</a>	The oneAPI FPGA Optimization Guide provides guidance on leveraging the functionalities of SYCL to optimize your design.
<a href="#">Explore SYCL Through Intel® FPGA Code Samples</a>	Provides guidance on how to target and develop your design on an FPGA using the oneAPI programming model. It also provides guidance on how to optimize a design to achieve performance and latency targets for an application targeting the FPGA.
<a href="#">Intel® FPGA SDK for OpenCL™ Pro Edition: Custom Platform Toolkit User Guide</a>	Outlines the procedure for creating an Intel® FPGA Software Development Kit (SDK) for OpenCL™ Pro Edition Custom Platform. The Intel® FPGA SDK for OpenCL™ Pro Edition Custom Platform Toolkit provides the necessary tools for implementing a fully functional Custom Platform.
<a href="#">oneAPI GPU Optimization Guide</a>	The oneAPI GPU Optimization Guide demonstrates how to improve the behavior of your software by partitioning it across the host and accelerator to specialize portions of the computation that run best on the accelerator. Specialization includes restructuring and tuning the code to create the best mapping of the application to the hardware. The value of oneAPI is that it allows each of these variations to be expressed in a common language with device-specific variants launched on the appropriate accelerator.

For more information about this toolkit, see the [Intel® oneAPI Base Toolkit](#) page.

## 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.

© Intel Corporation. Intel, the Intel logo, and other Intel marks are trademarks of Intel Corporation or its subsidiaries. Other names and brands may be claimed as the property of others.

Product and Performance Information
Performance varies by use, configuration and other factors. Learn more at <a href="http://www.Intel.com/PerformanceIndex">www.Intel.com/PerformanceIndex</a> . Notice revision #20201201

No license (express or implied, by estoppel or otherwise) to any intellectual property rights is granted by this document.

The products described 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.

Intel disclaims all express and implied warranties, including without limitation, the implied warranties of merchantability, fitness for a particular purpose, and non-infringement, as well as any warranty arising from course of performance, course of dealing, or usage in trade.