Intel® oneAPI Toolkits Installation Guide for Linux* OS

Contents

Intel® oneAPI Toolkits and Components Installation Guide for Linux* OS

1

This guide covers the installation of Intel® oneAPI toolkits and standalone components on Linux* OS.

Before You Begin

Check the Prerequisites information before installing the Intel oneAPI packages.

Installation

Install Intel oneAPI component and toolkit packages with one of the following options:

- Install with GUI
- Install with Command Line
- Install Using Package Managers
- Install Using Docker Container

Next Steps

Get Intel oneAPI code samples and refer to the toolkit Get Started page for detailed usage instructions, examples, and more:

- Get Started with Intel® oneAPI Base Toolkit
- Get Started with Intel® HPC Toolkit
- Get Started with Intel® Rendering Toolkit

Prerequisites

Consider the following important information before installing the Intel oneAPI packages.

Check System Requirements

Refer to one of the following documents specific for your toolkit to learn more about compatibility details:

- Intel® oneAPI Base Toolkit Release Notes | System Requirements
- Intel® HPC Toolkit Release Notes | System Requirements
- Intel® oneAPI System Bring-up Toolkit Release Notes | System Requirements
- Intel® Rendering Toolkit Release Notes | System Requirements

Install Eclipse*

To use third-party IDE, install Eclipse* on your Linux* OS host system before installing oneAPI Toolkits. This allows you to integrate the plugins as part of the Intel oneAPI Base Toolkit installation.

Set Environment Variables (optional)

Environment variables can be set up manually (as described in Get Started Guides and sample README files) or automatically using one of the methods below:

- Use modulefiles
- Use a setvars.sh configuration file

Set Up Your System for GPU

If you are using GPU, complete the following steps before or after Intel oneAPI installation:

- Install GPU drivers
- Check that you have fulfilled the requirements of Intel® Graphics Compute Runtime for oneAPI Level Zero and OpenCL™ Driver. Make sure that you have permissions to access the `/dev/dri/renderD\*` and `/dev/dri/card\*` files. This typically means that your user account is a member of the video (on Ubuntu* 18, Fedora* 30, and SLES* 15 SP1) or render (on Ubuntu* 19 and higher, CentOS* 8, and Fedora* 31) group. Alternatively, an administrator with sudo or root privilege can change the group owner of `/dev/dri/renderD\*` and `/dev/dri/card\*` to a group ID used by your user base.
- For HPC use cases, adjust driver defaults by setting `udev` rules as described in Set Up User Permissions for Using the Device files for Intel GPUs.
- If you plan to use the Intel® Distribution for GDB* on Linux* OS, make sure to configure debugger access.

- Install Intel GPU Drivers
- Install Plugins for non-Intel GPUs

Install Intel GPU Drivers

To develop and run C++ and SYCL* applications for your Intel GPU on Linux, you must first install the latest Intel GPU drivers. Driver defaults are only appropriate for display applications. For other applications, defaults should be adjusted by `udev` rules as described in Set Up User Permissions for Using the Device files for Intel GPUs.

Install Plugins for non-Intel GPUs

- To use an AMD* GPU with the Intel® oneAPI DPC++ Compiler, install the oneAPI for AMD GPUs plugin from Codeplay.
- To use an NVIDIA* GPU with the Intel® oneAPI DPC++ Compiler, install the oneAPI for NVIDIA GPUs plugin from Codeplay.

Installation

Toolkit Installation

> **Important** Some domain-specific toolkits require you to install the Intel oneAPI Base Toolkit first for full functionality.

Use one of the following options to get and install a toolkit package:

Toolkit Name	Binary Installer	Package Manager	Docker Container
Intel® oneAPI Base Toolkit	Online/offline installer	YUM, DNF, Zypper, APT	Docker Hub

Toolkit Name	Binary Installer	Package Manager	Docker Container
Intel® HPC Toolkit (requires installation of the Intel oneAPI Base Toolkit)	Online/offline installer	YUM, DNF, Zypper, APT	Docker Hub
Intel® Rendering Toolkit	Online/offline installer	YUM, DNF, Zypper, APT	N/A

Each of the binary installer types operates in various modes, which are covered later in this section.

Important For FPGA support in the online/offline installer, ensure that you choose to customize your installation and select the **FPGA Support Package for the Intel® oneAPI DCP++/C++ Compiler** component as part of your installation.

NOTE When using the offline installer, you can enable full offline mode by setting the environment variable `INTEL_SUPPRESS_INTERNET_CONNECTION=1`. When this mode is enabled:

- Installer does not send installation statistics
- Download-only mode is disabled
- Upgrade mode is disabled

Component Installation

You can install toolkit components as standalone via binary installer or package managers. Refer to the Single Component Downloads and Runtime Versions resource to locate a component package.

AI Tools Installation

To get the installation commands for the AI Tools, use the AI Tools Selector. For more information, refer to the AI Tools Selector Guide.

FPGA Development Flow

For instructions on installing the Intel® oneAPI Base Toolkit components required for FPGA development, refer to Install Compiler Components for Altera FPGA Development Flows.

The following FPGA development flows are supported:

- SYCL* HLS Flow

  This flow targets an Intel® FPGA device to generate an RTL IP core that you integrate into your FPGA application design using the Quartus® Prime software.
- FPGA Acceleration Flow

  This flow targets FPGA acceleration boards and requires a third party vendor-provided BSP for deployment of your kernel.

  Depending on the type of FPGA compilation you perform, you might need different software.

Both FPGA development flows require Quartus® Prime software when compiling your design for FPGA simulation or generating an FPGA hardware image. Quartus® Prime software is not required when compiling your design for FPGA emulation or to obtain the FPGA Optimization Report.

For an outline of the steps required to set up your FPGA development environment, refer to Installing the FPGA Development Environment in the *Intel® oneAPI DPC++/C++ Compiler Handbook for Altera FPGAs*.

Install with GUI

Download an installation package using the offline installer option for the toolkit you wish to install from here.

After downloading the toolkit installation package, follow the steps below to install it with GUI.

1. Launch the installer with the following command:
   - root: `sudo sh ./l_[Toolkit Name]Kit_[version].sh`
   - user: `sh ./l_[Toolkit Name]Kit_[version].sh`
2. Follow the installer instructions.
3. Once the installation is complete, verify that your toolkit is installed to the correct installation directory:
   - root: `/opt/intel/oneapi`
   - user: `~/intel/oneapi`

NOTE If you are using Intel GPU, you need to install the latest GPU drivers separately.

Install with Command Line

NOTE

- If you have root (administrative and sudo) permissions, you can install Intel® oneAPI toolkits in a shared directory. This allows any Intel® oneAPI Toolkits installed in this manner to be accessible for other users on the same Linux system. Refer to *root* installation commands in this document.
- If you do not have root (administrative or sudo) permissions, you can install the Intel® oneAPI toolkits only in your home directory or a user-specific location. Refer to *user* installation commands in this document.

Important: For FPGA flows, always install Intel oneAPI products with sudo permissions.

1. Download the installation package for your toolkit.
2. Run the installation package in one of the following modes:
   - **Non-interactive (silent) installation**, which allows you to define the installation configuration only once and does not require any user input during installation:
     - root:

```
sudo sh ./l_[Toolkit Name]Kit_[version].sh -a --silent --eula accept
```

     - user:

```
sh ./l_[Toolkit Name]Kit_[version].sh -a --silent --eula accept
```

     For the full list of supported installer options, refer to the Command Line Options section.
   - **Interactive mode**, which prompts you to select or confirm certain options during the installation process:
     - root:

```
sudo sh ./l_[Toolkit Name]Kit_[version].sh -a --cli
```

     - user:

```
sh ./l_[Toolkit Name]Kit_[version].sh -a --cli
```

     For the full list of supported installer options, refer to the Command Line Options section.
3. Once the installation is complete, verify that the toolkit is installed in the default directory:

- root:

```
/opt/intel/oneapi
```

- user:

```
~/intel/oneapi
```

4. If you are using GPU, you need to install the Intel GPU drivers separately.

Command Line Options

Options for Package Extraction Script

`-h, --help`	Show help for the package extraction script.
`-f, --extract-folder`	Point to the folder where the package content will be saved.
`-x, --extract-only`	Unpack the installation package without launching the installer.
`-r, --remove-extracted-files <yes|no>`	Remove extracted files after installation. This action cleans up the temporary package file location.
`-l, --log <log file>`	Log all package extraction actions to the specified file.
`-a <arguments>`	Pass arguments to the installer.

Arguments for Installer

Specify these arguments after the `-a` option.

Arguments for Installer

Option	Supported mode	Default value (if option is not passed)	Description
`-c, --cli`	CLI	N/A	Run the installer in interactive text-based user interface (TUI) mode.
`-s, --silent`	Silent	N/A	Run the installer in non-interactive (silent) mode.
`--eula`	Silent	`decline`	Required. Accept or decline End User License Agreement (EULA), supported values: `accept` or `decline` (default).
`--action`	Silent/CLI	`install`	Specify one of the supported values below when the installer action is needed: • `install` (default) Install the product. Use the `--components` option to specify the list of components to be installed. If not specified, the default set of components is installed. • `remove` Uninstall the product.

Option	Supported mode	Default value (if option is not passed)	Description
			• `modify` Change the current set of components installed. List all the components you need using the `--components` option. Components that are already installed still must be in the list if remain relevant. • `downloadonly` Download an offline installation package without installing it. To customize the list of components to be included into a package, use the `--components` option. • `repair` Repair the currently installed product.
`--instance`	Silent/CLI	`default`	Specify an ID of an installation instance. For example: `sh ./l_[Toolkit Name]Kit_[version].sh -a --instance=<instance ID>`. This option enables side-by-side installation of oneAPI products. Each instance is a separate installation entity with its own isolated environment. Product installed in one instance is not visible in another instance. If omitted, installation is performed in default instance. To get the list of available instances, use the `--list-instances` option.
`--list-instances`	Silent/CLI	N/a	Get the list of available installation instances.
`--config`	Silent/CLI	N/A	Point to the configuration INI file with options. Use this file as an alternative to passing options via the command line; mixed approach is also supported. Sample content of a configuration file: `s=eula=accept`. Use this command to run the installer with the options passed via `config.txt`: `sh ./l_[Toolkit Name]Kit_[version].sh --config config.txt`
`--components`	Silent	`default`	Specify components to perform an action on, supported values: `all`, `default`, custom components split by ':'. If you need the default components and some extra component(s), combine `default` with the name of the extra component(s) separated by ':'. For example: `--components default:<component_name>`.
`--list-products`	N/A	N/A	Get the list of downloaded products, their IDs, versions and statuses (installed/not installed). Use together with the `--instance` option to get the list of available products in a specific instance. For

Option	Supported mode	Default value (if option is not passed)	Description
			example: `sh ./l_[Toolkit Name]Kit_[version].sh -a --list-products --instance=<instance ID>`.
`--product-id`	Silent/CLI	N/A	Specify an ID of a product to perform an action on. Use this option with `--list-components` or `--action {install|remove|modify|repair}`.
`--product-ver`	Silent/CLI	N/A	Specify a product version to perform an action on. Use this option with `--list-components` or `--action {install|remove|modify|repair}`.
`--list-components`	N/A	N/A	Get the list of available components of the current package or of a product specified with `--product-id`. Use together with the `--instance` option to get the list of available components in a specific instance. For example: `sh ./l_[Toolkit Name]Kit_[version].sh -a --list-components --instance=<instance ID>`.
`--package-path`	Silent/CLI	N/A	Specify the directory of the package to install.
`--install-dir`	Silent	default installation directory	Supported in silent mode. Customize the installation directory.
`--log-dir`	Silent/CLI	default log location	Customize the directory to save the log file to.
`--proxy`	Silent/CLI	N/A	Specify proxy settings in the following format: `http://username:password@proxy-server.mycorp.com:3128`.
`--download-cache`	Silent	default download cache location	Point to the directory to store all downloaded and cached files.
`--download-dir`	Silent	default download directory	Customize the download directory, which is used in download-only mode.
`--intel-sw-improvement-program-consent`	Silent	`decline`	Accept or decline participation in Intel Software Improvement Program, supported values: `accept` or `decline` (default). To get the program description, use the `--show-intel-sw-improvement-program-consent` command.
`--show-intel-sw-improvement-program-consent`	N/A	N/A	Show the detailed description of the Intel Software Improvement Program.

Option	Supported mode	Default value (if option is not passed)	Description
`--ignore-errors`	Silent/CLI	N/A	Complete installation even if non-critical errors occur. Check the log file for the list of errors occured and ignored during installation.
`-h, --help`	N/A	N/A	Show the installer help.
`-p, --property`	Silent/CLI	N/A	Pass additional custom options. For example, the string `-p=option1=value -p option2=value` gives two additional options. If a custom option is provided twice with different values, only the latest one will be used. For example, the string `-p=option=a -p option=b` takes `b` as value for `option`.

Command Line Options Usage Examples

- Display the list of already installed products and products included in the downloaded package:
  - root:

```
sudo sh ./l_[Toolkit Name]Kit_[version].sh -a --silent --list-products
```

  - user:

```
sh ./l_[Toolkit Name]Kit_[version].sh -a --list-products
```

  Example of output:

```
ID Version Language Installed Name
================================================================================================
intel.oneapi.lin.tbb.product 2021.1.1-129 false Intel® oneAPI Threading Building Blocks
```

- Display the list of components in product of current package:
  - root:

```
sudo sh ./l_[Toolkit Name]Kit_[version].sh -a --list-components
```

  - user:

```
sh ./l_[Toolkit Name]Kit_[version].sh -a --list-components
```

- Display the list of components of any installed product on the system:
  - root:

```
sudo sh ./l_[Toolkit Name]Kit_[version].sh -a --list-components --product-id
intel.oneapi.lin.tbb.product --product-ver 2021.1.1-129
```

  - user:

```
sh ./l_[Toolkit Name]Kit_[version].sh -a --list-components --product-id
intel.oneapi.lin.tbb.product --product-ver 2021.1.1-129
```

  Example of output:

```
ID Version Language Installed Name
================================================================================================
intel.oneapi.lin.tbb.devel 2021.1.1-129 Intel® oneAPI Threading Building Blocks
```

- Install specific Intel oneAPI Toolkit products and components:

- root:

```
sudo sh ./l_[Toolkit Name]Kit_[version].sh --silent --eula accept --components
intel.oneapi.lin.tbb.devel
```

- user:

```
sh ./l_[Toolkit Name]Kit_[version].sh -a --silent --eula accept --components
intel.oneapi.lin.tbb.devel
```

Install Using Package Managers

You can install Intel oneAPI packages from one of these repositories:

- YUM, DNF, Zypper
- APT
- Cloudera (oneMKL)
- Spack
- Conda
- PIP
- NuGet

YUM/DNF/Zypper

You can install Intel oneAPI packages using either the YUM (DNF) or Zypper package managers, depending on your Linux distribution.

Pre-installation Steps

1. Check the System Requirements for your toolkit to make sure that your OS is supported:
   - Intel® oneAPI Base Toolkit
   - Intel® HPC Toolkit
   - Intel® Rendering Toolkit

   Use this command to identify your OS version:

```
# Redhat, Fedora, CentOS and related
more /etc/redhat-release

# Ubuntu, Debian, others
more /etc/lsb-release
```

2. If you plan to use Intel GPU, install the Intel GPU drivers.
3. If you are on a company intranet or behind a firewall, set the `http_proxy` and `https_proxy` environment variables to allow YUM/DNF/Zypper access the repository servers using HTTPS protocol.
4. Set up the repository:

   If using YUM/DNF:

   a. Create the YUM or DNF repo file in the /temp directory as a normal user:

```
tee > /tmp/oneAPI.repo << EOF
[oneAPI]
name=Intel® oneAPI repository
baseurl=https://yum.repos.intel.com/oneapi
enabled=1
gpgcheck=1
repo_gpgcheck=1
gpgkey=https://yum.repos.intel.com/intel-gpg-keys/GPG-PUB-KEY-INTEL-SW-PRODUCTS.PUB
EOF
```

b. Move the newly created `oneAPI.repo` file to the YUM/DNF configuration directory `/etc/yum.repos.d`:

```
sudo mv /tmp/oneAPI.repo /etc/yum.repos.d
```

If using Zypper:

Add the Intel oneAPI repository public key with the following command:

```
sudo zypper addrepo https://yum.repos.intel.com/oneapi oneAPI
```

By adding this new repository, Zypper automatically imports the public repo key. For some cases rpm might require explicit key import by:

```
rpm --import https://yum.repos.intel.com/intel-gpg-keys/GPG-PUB-KEY-INTEL-SW-PRODUCTS.PUB
```

Install Package

1. Get the name of a toolkit package that you need to install from the list of Intel oneAPI packages. Write down or copy your package name for future reference.
2. Install the needed package with the following command:

```
sudo {yum|dnf|zypper} install <package_name>
```

For example, to install the Intel® oneAPI Base Toolkit package from YUM, use:

```
sudo yum install intel-basekit
```

NOTE If you need to install on a machine with no internet access, or in case of a large distributed installation on a cluster, you can download a package without installing it with the `--download-only` option. For more details, refer to YUM instructions or Zypper man pages.

Installation is complete! For next steps, refer to the Get Started Guide for your toolkit:

- Get Started with Intel® oneAPI Base Toolkit
- Get Started with Intel® HPC Toolkit
- Get Started with Intel® Rendering Toolkit

Upgrade Toolkit/Component

You can upgrade toolkit or component package to the latest version using the following instructions:

- Toolkit: `{yum|dnf|zypper} upgrade <toolkit package>`

  For example, to upgrade the Intel oneAPI Base Toolkit package to the latest version, use the following command:

```
# If using YUM:
sudo yum upgrade intel-basekit


# If using DNF:
sudo dnf upgrade intel-basekit


# If using Zypper:
sudo zypper upgrade intel-basekit
```

- Component: `{yum|dnf|zypper} upgrade <component package>`

For example, to upgrade the Intel Distribution for GDB* package, use the following command:

```
# If using YUM:
sudo yum upgrade intel-oneapi-dpcpp-debugger


# If using DNF:
sudo dnf upgrade intel-oneapi-dpcpp-debugger


# If using Zypper:
sudo zypper upgrade intel-oneapi-dpcpp-debugger
```

List of Intel® oneAPI Packages

Toolkit Packages

The following toolkits and associated versions are available for installation via YUM repositories:

NOTE The repositories always contain the latest released version.

Toolkit Name	64-bit Meta Package Name (default)	32-bit Meta Package Name*
Intel® oneAPI Base Toolkit	`intel-basekit`	`intel-basekit-32bit`
Intel® HPC Toolkit	`intel-hpckit`	`intel-hpckit-32bit`
Intel® Rendering Toolkit	`intel-renderkit`	`intel-renderkit-32bit`

* - only required if you deploy 32-bit applications

Runtime Library Packages

The oneAPI repository provides runtime library packages. Install these packages on systems where you run oneAPI applications but do not do development, compilation, or runtime profiling. The following runtime library packages are available:

- oneAPI runtime libraries package, which is a superset of all runtimes for oneAPI components:
  - 64-bit: `intel-oneapi-runtime-libs`
  - 32-bit: `intel-oneapi-runtime-libs-32bit`
- Component runtime library packages. For instructions on how to get the list of all available standalone runtime packages, see List Standalone Runtime Library Packages.

APT

Pre-installation Steps

NOTE If you have an existing installation of Intel® oneAPI Beta, remove it with the following command:

```
sudo apt autoremove <package_name>
```

1. Check the toolkit-specific System Requirements page to make sure that your OS is supported:
   - Intel® oneAPI Base Toolkit
   - Intel® HPC Toolkit
   - Intel® Rendering Toolkit

   You can get your OS version using the following command depending on your Linux distribution:

```
# Redhat, Fedora, CentOS and related
more /etc/redhat-release


# Ubuntu, Debian, others
more /etc/lsb-release
```

2. If you plan to use Intel GPU, install the Intel GPU drivers.
3. Set up the repository:

```
# download the key to system keyring
wget -O- https://apt.repos.intel.com/intel-gpg-keys/GPG-PUB-KEY-INTEL-SW-PRODUCTS.PUB \
| gpg --dearmor | sudo tee /usr/share/keyrings/oneapi-archive-keyring.gpg > /dev/null

# add signed entry to apt sources and configure the APT client to use Intel repository:
echo "deb [signed-by=/usr/share/keyrings/oneapi-archive-keyring.gpg] https://apt.repos.intel.com/
oneapi all main" | sudo tee /etc/apt/sources.list.d/oneAPI.list
```

4. Update packages list and repository index:

```
sudo apt update
```

Install Packages

NOTE If you are on a company intranet or behind a firewall, set the `http_proxy` and `https_proxy` environment variables to allow APT access the repository servers using HTTPS protocol.

1. Get the name of a toolkit package that you need to install from the list of Intel oneAPI packages. Write down or copy your package name for future reference.
2. Install the needed package with the following command:

```
sudo apt install <package_name>
```

   For example, to install the Intel® oneAPI Base Toolkit package, use:

```
sudo apt install intel-basekit
#repeat 'apt install ...' for each toolkit you need
```

   If you need to install on a machine with no internet access, or in case of a large distributed installation on a cluster, you can download a package without installing it with the `--download-only` option.

NOTE If you want to integrate tools into the Eclipse* IDE, open Eclipse and verify that a menu titled **Intel** is present. If the menu is not present, see Installing Eclipse* Plugins from the IDE.

Installation is complete! For next steps, refer to the Get Started Guide for your toolkit:

- Get Started with Intel® oneAPI Base Toolkit
- Get Started with Intel® HPC Toolkit
- Get Started with Intel® Rendering Toolkit

NOTE If you have applications with long-running GPU compute workloads in native environments, you must disable the hangcheck timeout period to avoid terminating workloads.

List of Intel® oneAPI Packages

Toolkit Packages

The following toolkits and associated versions are available for installation via APT repositories:

NOTE The repositories always contain the latest released version.

Toolkit Name	64-bit Meta Package Name (default)	32-bit Meta Package Name*
Intel® oneAPI Base Toolkit	`intel-basekit`	`intel-basekit-32bit`
Intel® HPC Toolkit	`intel-hpckit`	`intel-hpckit-32bit`
Intel® Rendering Toolkit	`intel-renderkit`	`intel-renderkit-32bit`

* - only required if you deploy and deploy 32-bit applications

Intel® System Bring-Up Toolkit is not distributed via a repository, see details.

Runtime Library Packages

The oneAPI repository provides runtime library packages. Install these packages on systems where you run oneAPI applications but do not do development, compilation, or runtime profiling. The following runtime library packages are available:

- oneAPI runtime libraries package, which is a superset of all runtimes for oneAPI components:
  - 64-bit: `intel-oneapi-runtime-libs`
  - 32-bit: `intel-oneapi-runtime-libs-32bit`
- Component runtime library packages. For instructions on how to get the list of all available standalone runtime packages, refer to the List Standalone Runtime Library Packages section.

Conda

This page provides general instructions on installing the Intel® oneAPI component packages via the Conda* package manager.

For additional installation notes, refer to the Conda documentation.

To install a package, execute the following command:

- To install the latest version available:

```
conda install -c https://software.repos.intel.com/python/conda/ -c conda-forge <package_name>
```

  To get your package name, refer to the list of packages in the table below.
- To install a specific version:

```
conda install -c https://software.repos.intel.com/python/conda/ -c conda-forge
<package_name>==<version>
```

  For example: `conda install -c https://software.repos.intel.com/python/conda/ -c conda-forge mkl==2024.2.1`

List of Available Packages

Component Name	Package Name	Platform	Dependencies
Intel® MPI Library	impi_rt impi-devel	linux-x64	N/A
Intel® Fortran Compiler and Intel® Fortran Compiler Classic	intel-fortran-rt	linux-x64 linux-x86	Intel OpenMP* Runtime Library
Intel® CPU Runtime for OpenCL™ Applications	intel-opencl-rt	linux-x64	oneTBB
Intel® oneAPI DPC++/C++ Compiler	dpcpp-cpp-rt dpcpp_linux-64	linux-x64	Intel® CPU Runtime for OpenCL™ Applications Intel OpenMP* Runtime Library
	intel-sycl-rt	linux-x64	N/A
Intel® oneAPI DPC++ Library	onedpl-devel	linux-x64	N/A
Intel OpenMP* Runtime Library	intel-openmp	linux-x64 linux-x86	N/A
Intel® oneAPI Threading Building Blocks (oneTBB)	tbb tbb-devel	linux-x64 linux-x86	N/A
	tbb4py	linux-x64	N/A
Intel® oneAPI Data Analytics Library (oneDAL)	dal dal-static dal-devel dal-include	linux-x64	oneTBB Intel® CPU Runtime for OpenCL™ Applications
	daal4py	linux-x64	oneDAL
Intel® Integrated Performance Primitives (Intel® IPP)	ipp ipp-static ipp-include	linux-x64 linux-x86	N/A

Component Name	Package Name	Platform	Dependencies
	ipp-devel		
Intel® Integrated Performance Primitives Cryptography	ipp_crypto ipp_crypto-static ipp_crypto-include	linux-x64	N/A
Intel® oneAPI Math Kernel Library (oneMKL)	mkl mkl-devel mkl-static mkl-include	linux-x64 linux-x86	Intel OpenMP* Runtime Library oneTBB
	onemkl-sycl-blas onemkl-sycl-lapack onemkl-sycl-dft onemkl-sycl-sparse onemkl-sycl-vm onemkl-sycl-rng onemkl-sycl-stats onemkl-sycl-datafitting mkl-dpcpp mkl-devel-dpcpp	linux-x64	Intel® oneAPI DPC++/C++ Compiler Runtime Intel® CPU Runtime for OpenCL™ Applications
Intel® oneAPI Deep Neural Network Library (oneDNN)	onednn	linux-x64	Intel® CPU Runtime for OpenCL™ Applications Intel® oneAPI DPC++/C++ Compiler Runtime
	onednn-devel	linux-x64	N/A
Intel® oneAPI Collective Communications Library (oneCCL)	oneccl-devel	linux-x64	N/A
Thread Composability Manager (TCM)	tcm	linux-x64 linux-x86	N/A

PIP

This page provides general instructions on installing the Intel® oneAPI component packages from the Python* Package Index (PyPI).

For additional installation notes, refer to the PyPI documentation.

To install a package, execute the following command:

- To install the latest version available:

```
pip install <package_name>
```

  To get your package name, refer to the list of packages in the table below.
- To install a specific version:

```
pip install -c intel <package_name>==<version>
```

  For example: `pip install mkl==2021.1.1`

> **Important** For Intel® oneAPI Deep Neural Network Library (oneDNN), only packages of identical configuration can be installed into one environment. For example, you can install onednn-devel-cpu-vcomp with onednn-cpu-vcomp, but should avoid installing it with packages of other configurations, like cpu-iomp, cpu-tbb, cpu-dpcpp-gpu-dpcpp.

List of Available Packages

Component Name	Package Name	Platform	Dependencies
Intel® MPI Library	`impi_rt` `impi-devel`	`linux-x64`	N/A
Intel® Fortran Compiler and Intel® Fortran Compiler Classic	`intel-fortran-rt`	`linux-x64` `linux-x86`	Intel® MPI Library Intel OpenMP* Runtime Library
Intel® CPU Runtime for OpenCL™ Applications	`intel-opencl-rt`	`linux-x64`	oneTBB
Intel® oneAPI DPC++/C++ Compiler	`dpcpp-cpp-rt`	`linux-x64`	Intel® CPU Runtime for OpenCL™ Applications Intel OpenMP* Runtime Library
	`intel-sycl-rt`	`linux-x64`	N/A
Intel® oneAPI DPC++ Library	`onedpl-devel`	`linux-x64`	N/A
Intel OpenMP* Runtime Library	`intel-openmp`	`linux-x64` `linux-x86`	N/A
Intel® oneAPI Threading Building Blocks (oneTBB)	`tbb`	`linux-x64`	N/A

Component Name	Package Name	Platform	Dependencies
	tbb-devel	linux-x86	
	tbb4py	linux-x64	N/A
Intel® oneAPI Data Analytics Library (oneDAL)	daal daal-static daal-devel daal-include	linux-x64	oneTBB Intel® CPU Runtime for OpenCL™ Applications
	daal4py	linux-x64	oneDAL
Intel® Extension for Scikit-learn	scikit-learn-intelex	linux-x64	oneDAL
Intel® Integrated Performance Primitives (Intel® IPP)	ipp ipp-static ipp-include ipp-devel	linux-x64 linux-x86	N/A
Intel® Integrated Performance Primitives Cryptography	ipp_crypto ipp_crypto-static ipp_crypto-include	linux-x64 linux-x86	N/A
Intel® oneAPI Math Kernel Library (oneMKL)	mkl mkl-devel mkl-static mkl-include	linux-x64 linux-x86	Intel OpenMP* Runtime Library oneTBB
	onemkl-sycl-blas onemkl-sycl-lapack onemkl-sycl-dft onemkl-sycl-sparse onemkl-sycl-vm onemkl-sycl-rng onemkl-sycl-stats onemkl-sycl-datafitting	linux-x64 linux-x86	Intel® oneAPI DPC++/C++ Compiler Runtime Intel® CPU Runtime for OpenCL™ Applications

Component Name	Package Name	Platform	Dependencies
	mkl-dpcpp mkl-devel-dpcpp		
Intel® oneAPI Deep Neural Network Library (oneDNN)*	onednn onednn-devel	linux-x64	Intel® CPU Runtime for OpenCL™ Applications Intel® oneAPI DPC++/C++ Compiler Runtime
Intel® oneAPI Collective Communications Library (oneCCL)	oneccl-devel	linux-x64	N/A
Thread Composability Manager (TCM)	tcm	linux-x64 linux-x86	N/A

NuGet

This page provides general notes on how to install Intel® oneAPI components distributed via the NuGet channel. NuGet is a Microsoft-supported mechanism for sharing compiled code. It also defines how the packages are created, hosted and consumed, and it provides the tools for each of those roles. For more details on the installation process, please refer to the Microsoft* documentation.

Intel® oneAPI components distributed via NuGet include both development and runtime options.

For your convenience, the components are divided to *devel* and *static* packages corresponding to the different linking types (dynamic and static). Certain component packages are also split into x64 and x86 versions to reduce the overall package size.

Development Packages

The following table provides the full list of available packages:

Component Name	Package Name	Platform	Dependencies
Intel® MPI Library	intelmpi.devel.<platform>	linux-x64	N/A
Intel OpenMP* Runtime Library	intelopenmp.devel.<platform> intelopenmp.static.<platform>	linux	N/A
Intel® oneAPI Data Analytics Library (oneDAL)	inteldal.devel.<platform> inteldal.static.<platform>	linux-x64	oneTBB

Component Name	Package Name	Platform	Dependencies
Intel® oneAPI Threading Building Blocks (oneTBB)	`inteltbb.devel.<platform>`	`linux`	N/A
Intel® Integrated Performance Primitives Cryptography	`intelipp_crypto.devel.<platform>`	`linux-x64`	N/A
	`intelipp_crypto.static.<platform>`	`linux-x86`	
	`intelipp_crypto.nonpic.<platform>`		

All the specified dependencies will be downloaded automatically by the NuGet Package Manager.

Runtime Packages

The runtime packages are runtime redistributable libraries that will automatically load optimizations specific to your Intel hardware (including, but not limited to, vectorization). They can be used by another NuGet package that depends on these runtimes.

Component Name	Package Name	Platform Availability
Intel® oneAPI Data Analytics Library (oneDAL)	`inteldal.redist.<platform>`	`linux-x64`

Cloudera

This page provides general instructions on installing the Intel® oneAPI component parcel packages using the Cloudera* Manager. For additional notes, refer to the Parcels documentation and Cloudera Installation Guide.

Currently, only the Intel® oneAPI Math Kernel Library (oneMKL) component is distributed via Cloudera*. Package name: `mkl-<version>-el7.parcel`, for example, `mkl-2021.2.0.296-el7.parcel`.

The install the oneMKL parcel:

1. In the **Cloudera Manager Admin Console**, access the **Parcels** page by doing one of the following:
   - Click the **Parcels** indicator in the left navigation bar.
   - Click the **Hosts** in the left navigation bar, then click the **Parcels** tab.
2. At the **Parcels** page, click the **Parcel Repositories & Network Settings** button.
3. In the **Remote Parcel Repository URLs** list, click the plus symbol to open an additional row. Enter the path to Intel® MKL Parcel repository: `http://parcels.repos.intel.com/mkl/latest`. Click the **Save & Verify configuration** button.
4. Click the **Check for New Parcels** button. In the **Location** selector, click **Available Remotely**. The latest oneMKL parcel should be available for download.
5. Click the **Download** button for the oneMKL parcel. By downloading the package, you agree with the terms and conditions stated in the End-User License Agreement (EULA).
6. When download is completed, click the **Distribute** button to distribute the parcel on all cluster nodes.
7. When distribution is completed, click the **Activate** button to activate the parcel on all cluster nodes.

Note

The repository URL referenced above installs the latest version of oneMKL parcel. To install a lower version, use the URL based on the following model: `http://parcels.repos.intel.com/mkl/<version>.<update>.<build_number>`.

Spack

This page provides general instructions on installing the Intel® oneAPI component packages via Spack. After installation, you can use the tools directly or use Spack to build packages with the tools.

To install a package, execute the following command:

```
spack install <package_name>
```

Example

The example below demonstrates how to set up Intel oneAPI compilers with Spack.

1. Install the compilers package with the following command:

```
spack install intel-oneapi-compilers
```

2. Add the oneAPI compilers to the set of compilers that Spack can use:

```
spack compiler add `spack location -i intel-oneapi-compilers`/compiler/latest/bin
```

   For 2023.x and earlier versions, use:

```
spack compiler add `spack location -i intel-oneapi-compilers`/compiler/latest/linux/bin/intel64
spack compiler add `spack location -i intel-oneapi-compilers`/compiler/latest/linux/bin
```

   This adds the compilers to your `compilers.yaml`.

3. Verify that the compilers are available:

```
spack compiler add
```

The `intel-oneapi-compilers` package includes two families of compilers:

- Intel: `icc`, `iccpc`, `ifort` - Intel's classic compilers
- oneAPI: `icx`, `icpx`, `ifx` - Intel's new generation of compilers based on LLVM

To build the `patchelf` Spack package with `icc`, use:

```
spack install patchelf%intel
```

To build with `icx`, use:

```
spack install patchelf%oneapi
```

In addition to compilers, oneAPI contains many libraries. The `hdf5` package works with any compatible MPI implementation. To build `hdf5` with Intel oneAPI MPI, use:

```
spack install hdf5 +mpi
^intel-oneapi-mpi
```

For more information, see Spack documentation.

List Available Toolkits, Components, and Runtime Library Packages

Use the commands provided below to find and install specific toolkits, standalone components, standalone runtime library packages, or simply to see all available packages in a corresponding oneAPI repository:

- List toolkit packages
- List standalone components
- List standalone runtime library packages
- List all packages

List Toolkit Packages

To query the repository for available toolkit packages, use the following command:

YUM/DNF:

```
sudo -E {yum|dnf} --disablerepo="*" --enablerepo="oneAPI" list available | grep kit | grep -v
runtime
```

Zypper:

```
sudo -E zypper pa -ir oneAPI | grep kit | grep -v runtime
```

APT:

```
sudo -E apt-cache pkgnames intel | grep kit | grep -v runtime
```

List Standalone Components

The oneAPI repository also contains standalone components, which are packages that provide a specific tool for cases where you do not need an entire toolkit. For these packages, if there is a <component>-runtime package, make sure to get and install both the component package and its runtime package. Not all standalone components need an additional runtime package. If you do not see a runtime package for your standalone component, then you do not need one.

To query the repository for available standalone components and their runtime packages, use the following command:

YUM/DNF:

```
sudo -E {yum|dnf} --disablerepo="*" --enablerepo="oneAPI" list available | grep intel-oneapi |
grep -v intel-oneapi-runtime
```

Zypper:

```
sudo -E zypper pa -ir oneAPI | grep intel-oneapi | grep -v intel-oneapi-runtime
```

APT:

```
sudo -E apt-cache pkgnames intel | grep intel-oneapi | grep -v intel-oneapi-runtime
```

List Standalone Runtime Library Packages

The oneAPI repository provides standalone runtime library packages. Install these packages on systems where you run oneAPI applications but do not do development, compilation, or runtime profiling. In this case, you only need the shared libraries dynamically linked to by executables, provided by these packages.

To query the repository for available component runtime libraries, use the following command:

YUM/DNF:

```
sudo -E {yum|dnf} --disablerepo="*" --enablerepo="oneAPI" list available | grep intel-oneapi-
runtime
```

Zypper:

```
sudo -E zypper pa -ir oneAPI | grep intel-oneapi-runtime
```

APT:

```
sudo -E apt-cache pkgnames intel | grep intel-oneapi-runtime
```

List All Packages

To query all available Intel® oneAPI packages provided in a repository, use the following command:

YUM/DNF:

```
sudo -E {yum|dnf} --disablerepo="*" --enablerepo="oneAPI" list available
```

Zypper:

```
sudo -E zypper pa -ir oneAPI
```

APT:

```
sudo -E apt-cache pkgnames intel
```

Install Packages or Components to Different Directories

Intel oneAPI installer supports side-by-side installation. It means that you can install multiple instances of toolkits/components to different directories on the same machine. Each instance is a separate installation entity with its own isolated environment. Product installed in one instance is not visible in another instance.

With multi-instance installation, you can:

- Install a newer version of a toolkit/component without removing the previous one
- Install toolkits to different directories other than default
- Have multiple instances of the same version of a toolkit/component installed

To install a package into a specific instance, use the following command:

- root:

```
sudo sh ./l_[Toolkit Name]Kit_[version].sh -a -s –eula=accept –-install-dir=<custom-install-dir>
--instance=<instance ID>
```

- user:

```
./l_[Toolkit Name]Kit_[version].sh -a -s –eula=accept –-install-dir=<custom-install-dir> --
instance=<instance ID>
```

where

- `<custom-install-dir>` is the directory where you want to install a specific instance to
- `<instance ID>` is a unique combination of alphanumeric symbols that designate an instance, for example `my-custom-instance-1`

For instructions on how to uninstall product(s) from a specific instance, refer to Uninstall Using Silent CLI.

Configure WSL 2 for GPU Workflows

With Microsoft* Windows Subsystem for Linux 2 (WSL 2), you can use native Linux distribution of Intel® oneAPI tools and libraries on Windows*. Get the latest version of WSL 2 following the process described in The Windows Subsystem for Linux in the Microsoft Store is now generally available on Windows 10 and 11.

To be able to use Intel oneAPI tools on WSL 2 for GPU workflows, install the Intel GPU drivers as described below.

Ubuntu* 20.04 (focal)

Step 1: Add package repository

Install the `repositories.intel.com/graphics` package repository by executing the following command in your WSL 2 console:

```
sudo apt-get install -y gpg-agent wget
wget -qO - https://repositories.intel.com/graphics/intel-graphics.key |
  sudo gpg --dearmor --output /usr/share/keyrings/intel-graphics.gpg
```

```
echo 'deb [arch=amd64 signed-by=/usr/share/keyrings/intel-graphics.gpg] https://
repositories.intel.com/graphics/ubuntu focal-devel main' | \
  sudo tee  /etc/apt/sources.list.d/intel.gpu.focal.list
sudo apt update
```

Tip Before pasting the command to your console, run `sudo ls` and enter your password to prevent the commands from being swallowed by the sudo password prompt.

The code above performs the following:

- Checks that your system has `gpg-agent` and `wget` installed
- Downloads and installs the public key used to verify the integrity of the package repository
- Adds the `repositories.intel.com/graphics` repository to the system

Step 2: Install runtime and development (optional) packages

Install GPU software packages with the following command:

```
sudo apt-get install \
  intel-opencl-icd \
  intel-level-zero-gpu level-zero \
  intel-media-va-driver-non-free libmfx1 libmfxgen1 libvpl2 \
  libegl-mesa0 libegl1-mesa libegl1-mesa-dev libgbm1 libgl1-mesa-dev libgl1-mesa-dri \
  libglapi-mesa libgles2-mesa-dev libglx-mesa0 libigdgmm11 libxatracker2 mesa-va-drivers \
  mesa-vdpau-drivers mesa-vulkan-drivers va-driver-all
```

OPTIONAL: If you plan to perform development tasks, you need to install the following optional development packages to make sure that oneAPI tools function correctly:

```
sudo apt-get install -y \
  libigc-dev \
  intel-igc-cm \
  libigdfcl-dev \
  libigfxcmrt-dev \
  level-zero-dev
```

Reboot WSL 2 by executing `wsl.exe –shutdown` in PowerShell.

Step 3: Verify installation

Verify Computing drivers installation:

```
sudo apt-get install clinfo
clinfo
```

The command should return similar to the following:

```
Number of platforms                               2
Platform Name                                   Intel(R) OpenCL HD Graphics
Platform Vendor                                 Intel(R) Corporation
Platform Version                                OpenCL 3.0
Platform Profile                                FULL_PROFILE
```

Ubuntu* 22.04 (jammy)

Step 1: Add package repository

Install the `repositories.intel.com/graphics` package repository by executing the following command in your WSL 2 console:

```
sudo apt-get install -y gpg-agent wget
wget -qO - https://repositories.intel.com/graphics/intel-graphics.key |
  sudo gpg --dearmor --output /usr/share/keyrings/intel-graphics.gpg
echo 'deb [arch=amd64,i386 signed-by=/usr/share/keyrings/intel-graphics.gpg] https://
repositories.intel.com/graphics/ubuntu jammy arc' | \
  sudo tee  /etc/apt/sources.list.d/intel.gpu.jammy.list
```

> **Tip** Before pasting the command to your console, run `sudo ls` and enter your password to prevent the commands from being swallowed by the sudo password prompt.

The code above performs the following:

- Checks that your system has `gpg-agent` and `wget` installed
- Downloads and installs the public key used to verify the integrity of the package repository
- Adds the `repositories.intel.com/graphics` repository to the system

Step 2: Install runtime and development (optional) packages

Install GPU software packages with the following command:

```
sudo apt-get install -y \
  intel-opencl-icd intel-level-zero-gpu level-zero \
  intel-media-va-driver-non-free libmfx1 libmfxgen1 libvpl2 \
  libegl-mesa0 libegl1-mesa libegl1-mesa-dev libgbm1 libgl1-mesa-dev libgl1-mesa-dri \
  libglapi-mesa libgles2-mesa-dev libglx-mesa0 libigdgmm12 libxatracker2 mesa-va-drivers \
  mesa-vdpau-drivers mesa-vulkan-drivers va-driver-all
```

OPTIONAL: If you plan to perform development tasks, you need to install the following optional development packages to make sure that oneAPI tools function correctly:

```
sudo apt-get install -y \
  libigc-dev \
  intel-igc-cm \
  libigdfcl-dev \
  libigfxcmrt-dev \
  level-zero-dev
```

To support Steam games, install i386 packages:

```
sudo dpkg --add-architecture i386
sudo apt-get update

sudo apt-get install  -y \
    udev mesa-va-drivers:i386 mesa-common-dev:i386 mesa-vulkan-drivers:i386 \
    libd3dadapter9-mesa-dev:i386 libegl1-mesa:i386  libegl1-mesa-dev:i386   \
    libgbm-dev:i386 libgl1-mesa-glx:i386 libgl1-mesa-dev:i386   \
    libgles2-mesa:i386 libgles2-mesa-dev:i386 libosmesa6:i386   \
    libosmesa6-dev:i386 libwayland-egl1-mesa:i386  libxatracker2:i386 \
    libxatracker-dev:i386 mesa-vdpau-drivers:i386  libva-x11-2:i386
```

Reboot WSL 2 by executing `wsl.exe –shutdown` in PowerShell.

Step 3: Verify installation

Verify Computing drivers installation:

```
sudo apt-get install clinfo
clinfo
```

The command should return similar to the following:

```
Number of platforms                               2
Platform Name                                   Intel(R) OpenCL HD Graphics
Platform Vendor                                 Intel(R) Corporation
Platform Version                                OpenCL 3.0
Platform Profile                                FULL_PROFILE
```

Install Compiler Components for Altera FPGA Development Flows

To use the Intel® oneAPI DPC++/C++ Compiler for development flows that target Altera FPGA devices, install the **FPGA Support Package for the Intel® oneAPI DPC++/C++ Compiler** (often referred to as the "FPGA Support Package"). This package adds FPGA-specific support to the Intel® oneAPI DPC++/C++ Compiler to enable FPGA code development.

At a high level, the steps to install the compiler components for FPGA development are as follows:

1. Obtain Compiler Components for FPGA Development.
2. Install compiler components in one of the following ways:
   - Install Compiler Components with GUI.
   - Install Compiler Components with Command Line.
   - Install Compiler Components with Linux Package Managers.
3. (optional) Additional Software Requirements for FPGA Development.

Obtain Compiler Components for FPGA Development

Installation Type	Installation Packages Required
Online installation	• Intel® oneAPI Base Toolkit online installer
Offline installation	• Intel® oneAPI Base Toolkit offline installer • FPGA Support Package for the Intel® oneAPI DCP ++/C++ Compiler* offline installer Download the FPGA Support Package offline installer from the Single Component Downloads and Runtime Versions page.
Linux Package Manager	• `intel-basekit` • `intel-oneapi-compiler-fpga`

Install Compiler Components with GUI

Online Installation

1. Start the GUI installer according to the instructions in Install with GUI.
2. In the online installer GUI, customize your installation and select the optional **FPGA Support Package for the Intel® oneAPI DCP++/C++ Compiler** component as follows:
   a. Click **Customize**.
   b. Expand **Intel® oneAPI DCP++/C++ Compiler**.
   c. Under **Optional**, select **FPGA Support Package for the Intel® oneAPI DCP++/C++ Compiler**.

Offline Installation

1. Run the GUI installer for the Intel® oneAPI Base Toolkit according to the instructions in Install with GUI.
2. Run the GUI installer for the FPGA Support Package for the Intel® oneAPI Base Toolkit according to the instructions in Install with GUI.

Install Compiler Components with Command Line

Online Installation

Include the following option when you run the installation command (refer to Install with Command Line for details):

```
--components default:intel.oneapi.lin.compiler.fpga
```

For example, to silently install the Intel oneAPI Base Toolkit and FPGA support package, run the following command:

- Root directory installation:

```
sudo sh ./l_BaseKit_p_[version].sh -a --silent --eula accept --components
default:intel.oneapi.lin.compiler.fpga
```

- User directory installation:

```
sh ./l_BaseKit_p_[version].sh -a --silent --eula accept --components
default:intel.oneapi.lin.compiler.fpga
```

Offline Installation

For an offline installation, run separate commands to install the Intel® oneAPI Base Toolkit and the FPGA Support Package for the oneAPI DCP++/C++ Compiler.

For example, to silently install the Intel oneAPI Base Toolkit and FPGA Support Package, run the following commands:

- Root directory installation:

```
sudo sh ./l_BaseKit_p_[version].sh -a --silent --eula accept
sudo sh ./l_compiler-fpga-addon_p_[version]_offline.sh -a --silent --eula accept
```

- User directory installation:

```
sh ./l_BaseKit_p_[version].sh -a --silent --eula accept
sh ./l_compiler-fpga-addon_p_[version]_offline.sh -a --silent --eula accept
```

Install Compiler Components with Linux Package Managers

To install the required components with Linux package managers (APT/YUM/DNF/Zypper), follow the instructions in Install Using Package Managers. The FPGA Support Package is not available in Conda, PIP, NuGet, Cloudera, or Spack.

Install the following packages:

- `intel-basekit`
- `intel-oneapi-compiler-fpga`

Additional Software Requirements for FPGA Development

- Install Quartus® Prime Software.

The Quartus® Prime software is required only when compiling your design for FPGA simulation or generating an FPGA hardware image. Quartus® Prime software is **not** required when compiling your design for FPGA emulation or to obtain the FPGA Optimization Report.

- Install FPGA Board Support Packages

  Board support packages are required only when compiling to FPGA hardware as part of the FPGA acceleration flow. BSPs are not required for the SYCL* HLS flow.

Installation Topology Variations

When setting up your system, you can install the required components in a variety of topologies:

- **Set up a single system**

  In this method, you can use a single system acting as both the runtime and development system. Install the Intel oneAPI Base Toolkit, FPGA Support Package for the Intel® oneAPI DCP++/C++ Compiler, Intel® Quartus® Prime Software, and FPGA board support package on the same system.
- **Set up separate development and runtime systems**

  In this method, you install the custom platform/FPGA device on the runtime system and run only the design. On the development system, install the Intel® Quartus® Prime software to compile and generate the FPGA bitstream. Refer to Intel® Quartus® Prime Software, Install Intel® FPGA Board Packages sections for more information.
- **Set up a cloud on-premise**

  A cloud on-premise helps reduce the hardware cost necessary for development. In this workflow, you can set up two development systems, one for the FPGA development and the other for the Intel® Quartus® Prime software compilation. The runtime system can be different. After setting up your development systems, install the physical card on the runtime system. Refer to the Intel® oneAPI DPC++ /C++ Library System Requirements for FPGA requirements.

  - On the first development system with lower configurations (8 GB RAM), iterate over your designs using the emulation and report flow to verify code correctness. Install the Intel® oneAPI Base Toolkit and FPGA Support Package for the Intel® oneAPI DCP++/C++ Compiler packages on this system. For more information about emulation and report flow, refer to Types of SYCL* FPGA Compilation.
  - On the second development system with higher configurations (memory requirements dictated by the FPGA acceleration board or FPGA development kit that you use), install the Intel® Quartus® Prime Software. Perform Intel® Quartus® Prime compilation using either the hardware flow or the device link flow. For more information, refer to Types of SYCL* FPGA Compilation in the *Intel oneAPI DPC++/C++ Compiler Handbook for Altera FPGAs*.

Install Quartus® Prime Software

Quartus® Prime software includes everything you need to design for Intel® FPGAs, from design entry and synthesis to optimization, verification, and simulation. It contains the following features:

- Hybrid Placer & Global Router
- Timing Analyzer
- Physical Synthesis
- Incremental Fitter Optimization
- Interface Planner
- Synthesis Engine
- Platform Designer
- Partial Reconfiguration
- Block-Based (Hierarchical) Design

NOTE The Quartus® Prime software is not required for the FPGA development flow emulation or report generation stages. You can complete those stages with only the Intel® oneAPI DPC++/C++ Compiler (part of the Intel oneAPI Base Toolkit) and the FPGA Compiler Support Package for the Intel® oenAPI DPC++/C++ Compiler.

If you want to use the Quartus® Prime software (required for FPGA hardware and simulation flow) with the Intel® oneAPI DPC++/C++ Compiler, the edition and version of Quartus® Prime software that you need depends on your target device (either standalone device or the device on an acceleration board). Refer to the Intel® oneAPI DPC++ /C++ Library System Requirements for more information.

- If you want to install a version of the Quartus® Prime software, follow the instructions in the following documents:
  - Intel® FPGA Software Installation and Licensing
  - Quartus® Prime Pro Edition User Guide: Getting Started
  - Quartus® Prime Standard Edition User Guide: Getting Started
- If you already have a version of the Quartus® Prime software installed on your system and you want to use that version, then use one of the following methods to set up the environment:
  - Set `QUARTUS_ROOTDIR_OVERRIDE = <path_to_your_quartus_folder>`
  - Add the `bin` directory of the Quartus® Prime software to your `PATH` variable.
- If you have multiple versions of the Quartus® Prime software installed, Altera recommends setting the `QUARTUS_ROOTDIR_OVERRIDE` variable to point to the Quartus software path you want to use. Otherwise, you might end up using a version different than the one you expected. Ensure that you set the `QUARTUS_ROOTDIR_OVERRIDE` variable after running the `setvars` script, which can potentially override your setting.

Install FPGA Board Support Packages

To compile an executable that can run on an FPGA board, install a Board Support Package (BSP) that allows targeting compiles to that board. Intel does not ship BSPs with oneAPI. You must download and install BSPs from a third-party vendor. For more information, refer to the Intel® FPGA development flow page.

To use a third-party vendor-provided BSP with the Intel® oneAPI Base Toolkit, follow these instructions:

1. Follow vendor-specific instructions to download and install the board package.
2. Install a version of the Quartus® Prime software that is compatible with the BSP (as indicated by the vendor).
3. Follow instructions in Install the Quartus® Prime Software to ensure the Intel oneAPI DPC++/C++ Compiler is configured to run with the installed Quartus® Prime software.

Related Links

- Intel® oneAPI DPC++/C++ Compiler System Requirements
- Get Started with the Intel® oneAPI Base Toolkit for Linux
- Intel® oneAPI DPC++/C++ Compiler Handbook for Intel FPGAs
- Intel® oneAPI FPGA Development Flow in the Intel® oneAPI DPC++/C++ Compiler Handbook for Altera FPGAs
- Intel® FPGA SDK for OpenCL™ Pro Edition: Custom Platform Toolkit User Guide
- Quartus® Prime Software User Guides

Uninstall oneAPI Toolkits and Components

Uninstall Intel PAC Card

To uninstall an installed software stack, you can run one of the following commands:

- Intel® PAC for Intel® Arria® 10 GX FPGA

```
aocl uninstall intel_a10gx_pac
```

- Intel® FPGA PAC D5005

```
aocl uninstall intel_s10sx_pac
```

After uninstalling the PAC card, you can delete the BSP files and the PAC directory using `rm -rf <pac_install_dir>`.

Uninstall oneAPI Toolkits

NOTE Manual removal of the installation folder (not recommended) does not uninstall the oneAPI toolkits completely. Incorrect uninstallation may block future installations. If you still need to remove the toolkit manually, make sure to clean up the system correctly and remove the following:

- Installation folder
- Installer cache, located at `/var/intel/installercache/*` (root) or `<user home>/intel/installercache/*` (user), including:
  - packages cache at `/var/intel/installercache/packagescache` (root) or `<user home>/intel/installercache/packagescache` (user)
  - download cache at `/var/intel/installercache/downloadcache/` (root) or `<user home>/intel/installercache/downloadcache` (user)
  - (Optional) package manager database `/var/intel/packagemanager.db` (root) or `<user home>/intel/packagemanager.db` (user)
- Installer and package manager executables, located at `/opt/intel/oneapi/installer/` (root) or `<user home>/intel/oneapi/installer/` (user) and `/opt/intel/packagemanager/` (root) or `<user home>/intel/packagemanager/` (user), respectively.

Uninstall Using GUI

Use the following commands to uninstall oneAPI Toolkits:

```
cd /opt/intel/oneapi/installer
sudo ./installer
```

- Select Remove to remove the toolkit
- Use Installer dashboard dialog with the list of already installed products (toolkits) to Modify, Repair or Remove each toolkit separately

Uninstall Using Silent CLI

Use the following commands to uninstall oneAPI Toolkits:

1. Display the list of already installed products and products included in the downloaded package using the following command:

```
l_[Toolkit Name]Kit_[version].sh -s -a --list-products
```

Example of output:

```
ID Version Language Installed Name
=============================================================================================
=========
intel.oneapi.lin.tbb.product 2021.1.1-129 false Intel® oneAPI Threading Building Blocks
```

2. Uninstall selected product:

```
cd /opt/intel/oneapi/installer
sudo ./installer --action remove --product-id intel.oneapi.lin.tbb.product --product-ver
2021.1.1-129
```

To uninstall a product from a specific installation instance, use the following command:

```
cd /opt/intel/oneapi/installer
sudo ./installer --action remove --product-id intel.oneapi.lin.tbb.product --product-ver
2021.1.1-129 --instance=<instance id>
```

where `<instance id>` is a unique combination of alphanumeric symbols set during multi-instance installation.

Uninstall Using Linux Package Manager

- APT

```
sudo apt autoremove <package_name>
```

- YUM

```
sudo yum autoremove <package_name>
```

Troubleshooting

Diagnose Errors

The Diagnostics Utility for Intel oneAPI toolkits provides more checks to find missing dependencies and permissions errors. Learn more.

Integrity check failed

During installation, you may get an error message about failed integrity check of downloaded files.

Cause: Downloaded files are corrupted because of the hard drive corruption.

Solution: Check hard drive health and restart the installation.

Corrupted terminal screen

After launching the installer in CLI mode, you may see a corrupted terminal screen. Dialogs and other elements are not rendered properly.

Cause: The installer does not support terminal size less than 80x24 characters.

Solution: Resize your terminal size to 80x24 characters or greater before launching the installer.

YUM packages conflict on Amazon Linux 2* OS

When installing toolkits of version 2021.2 or 2021.3 on Amazon Linux 2* OS via YUM, you may get an error similar to the following:

```
Error: intel-oneapi-tbb-2021.2.0 conflicts with intel-oneapi-common-
licensing-2021.1.1-2021.1.1-60.noarch
```

Solution: To work around the issue, install Intel oneAPI toolkits with the following commands:

- Intel® oneAPI Base Toolkit

```
sudo yum install intel-basekit intel-oneapi-common-licensing-<version> intel-oneapi-libdpstd-
devel-<version>
```

- Intel® HPC Toolkit

```
sudo yum install intel-hpckit intel-oneapi-common-licensing-<version> intel-oneapi-libdpstd-devel-<version>
```

- Intel® Rendering Toolkit

```
sudo yum install intel-renderkit intel-oneapi-common-licensing-<version>
```

where `<version>` is 2021.2.0 or 2021.3.0. For example:

```
sudo yum install intel-basekit intel-oneapi-common-licensing-2021.2.0 intel-oneapi-libdpstd-devel-2021.2.0
```

2021.x installation overwrites existing 2022.x installer

If you launch the installer for the 2021.x version of a toolkit on a system with the 2022.x version installed, your existing 2022.x installer will be downgraded to the 2021.x version. The 2021.x installer does not recognize installed 2022.x packages.

Cause: Compatibility issue between 2022.x and 2021.x versions of the installer.

Solution: Restore the latest installer by launching the installer for the 2022.x version of the toolkit. In future, when you need to install 2021.x on a system with 2022.x installed, before launching the installer, back up the following installer directories by renaming them:

- root: `/opt/intel/oneapi/installer` and `/opt/intel/packagemanager/1.0`
- user: `~/intel/oneapi/installer` and `~/intel/packagemanager/1.0`

When you are done with 2021.x, you can change back the directory names to restore the 2022.x installer.

Installation hangs indefinitely when launched in GUI mode using SSH

In an SSH session, installation may hang indefinitely when launched in GUI mode.

Cause: Issue detecting available screen for GUI in an SSH session.

Solution: Interrupt current process using Ctrl + C and launch the installer in CLI mode using the following command:

```
sh ./l_[Toolkit Name]Kit_[version].sh -a --cli
```

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.