Intel® oneAPI Toolkits Installation Guide for Windows*

Contents

Chapter 1: Intel® oneAPI Toolkits and Components Installation Guide for Windows*

Prerequisites.......3
    Install Intel GPU Drivers.......4
Installation.......4
    Install with GUI.......5
    Install with Command Line.......5
    Install Using Package Managers.......9
        Conda.......9
        PIP.......12
        NuGet.......15
        Maven.......17
Uninstall oneAPI Toolkits and Components.......17
Troubleshooting.......18
Notices and Disclaimers.......19

Intel® oneAPI Toolkits and Components Installation Guide for Windows* 1

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

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

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® oneAPI HPC Toolkit
- Get Started with Intel® oneAPI IoT Toolkit
- Get Started with Intel® oneAPI AI Analytics Toolkit
- Get Started with Intel® oneAPI System Bring-up Toolkit
- Get Started with Intel® oneAPI Rendering Toolkit

Prerequisites

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

System Requirements

Refer to the toolkit-specific Release Notes and System Requirements documents to learn more about compatibility details:

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

Microsoft* Visual Studio* Integration

Before enabling integration into Microsoft* Visual Studio* via the oneAPI toolkit installer, install the **Desktop development with C++** workload into each Visual Studio instance.

For the list of the supported Visual Studio versions, refer to the toolkit System Requirements.

Set Up Your System for Intel GPU

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

- Install Intel GPU Drivers

Install Intel GPU Drivers

If you use Intel GPU, you need to install the OpenCL and Level 0 graphics drivers. Follow the instructions applicable for your device:

- Intel® Arc™ Graphics, 11th-13th Gen Intel® Core™ Processor Graphics.
- Xe Dedicated, 6th-10th Gen Intel® Core™ Processor Graphics, and related Intel Atom®, Pentium®, and Celeron® processors. Driver version varies depending on the Intel Graphics in the system.
- Intel® Data Center GPU Flex Series (ATS-M). Contact your original equipment manufacturer (OEM) representative for access to the Intel Registration Center.

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 install a toolkit package:

Toolkit Name	Binary Installer	Package Manager
Intel® oneAPI Base Toolkit	Online/offline installer	N/A
Intel® oneAPI HPC Toolkit (requires installation of the Intel oneAPI Base Toolkit)	Online/offline installer	N/A
Intel® oneAPI IoT Toolkit (requires installation of the Intel oneAPI Base Toolkit)	Online/offline installer	N/A
Intel® oneAPI AI Analytics Toolkit	N/A	Conda
Intel® oneAPI Rendering Toolkit	Online/offline installer	N/A

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

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.

Get 32-bit Libraries

To get access to 32-bit binaries of the Intel® Integrated Performance Primitives (Intel® IPP) and Intel® oneAPI Math Kernel Library (Intel® oneMKL), install the Intel® oneAPI Base Toolkit (32-bit) add-on package from the Intel oneAPI Base Toolkit download page using the process described at Install with GUI.

To use 32-bit libraries from the Intel oneAPI Base Toolkit (32-bit) package, you need to have the same version of the Intel® oneAPI Base Toolkit or standalone libraries installed on your system.

Intel® FPGA Developmental Flow

For emulation and FPGA optimization report flow, you can just use the Intel oneAPI Base Toolkit that includes the Intel oneAPI DPC++/C++ Compiler. For more information about the flows, see FPGA Flow in the *Intel oneAPI Programming Guide*.

For the Intel FPGA hardware or simulation flow, you can target Intel® FPGA devices or a third-party vendor-provided BSP. The vendor specifies which version of Intel® Quartus® Prime software is necessary to compile the BSP. You must install the Intel® Quartus Prime software and BSP packages separately.

NOTE

- FPGA IP Authoring flow is now supported. Developing IP components/hardware with oneAPI requires the Intel oneAPI Base Toolkit and Intel® Quartus® Prime Pro Software. For details about getting started with the IP component development flow, refer to Getting Started with Intel® oneAPI Toolkits and Intel® Quartus® Prime Software and FPGA Flow in the *Intel® oneAPI Programming Guide*.
- Intel® Quartus® Prime software is required only for simulation and hardware generation flows, and integrating your IP component into your design.

Install with GUI

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

1. Double-click the `*.exe` file to launch the GUI installer: `w_[Toolkit Name]Kit_[version].exe`
2. Follow the installer instructions.
3. Choose the required Microsoft* Visual Studio* version to enable integration into IDE (if supported) from the corresponding dialog. You need to have the **Desktop development with C++** workload installed into each Visual Studio instance beforehand.
4. Once the installation is complete, verify that your toolkit is installed to the correct installation directory `C:\Program Files (x86)\Intel\oneAPI`.

NOTE By default, the installer GUI works via the OpenGL library. Alternatively, you can force running the installer GUI with the ANGLE library. To do this, set the `INTEL_USE_ANGLE` environment variable to 1.

Install with Command Line

General Instructions

Launch the installation script using the following command:

```
w_[Toolkit Name]Kit_[version]_offline.exe [options] -a [arguments]
```

where `[options]` contain parameters for the package extraction script, and `[arguments]` are options for the installer.

The package extraction script supports the following options:

`-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`	This option unpacks the installation package only. It does not launch 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.

The values after `-a` are passed as command line arguments to the installer. The following installer options are supported:

Option	**Default value (if option is not passed)**	**Description**
`-s, --silent`	N/A	Run the installer in non-interactive (silent) mode.
`--eula`	`decline`	Required. Accept or decline End User License Agreement (EULA), supported values: `accept` or `decline` (default).
`--action`	`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. • `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.

Option	Default value (if option is not passed)	Description
`--instance`	`default`	Specify an ID of an installation instance. For example: `w_[Toolkit Name]Kit_[version]_offline.exe -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.
`--config`	N/A	Point to the configuration INI file with options. You can 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`.
`--components`	`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	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 example: `w_[Toolkit Name]Kit_[version]_offline.exe -a --list-products --instance=<instance ID>`.
`--product-id`	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`	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	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: `w_[Toolkit Name]Kit_[version]_offline.exe -a --list-components --instance=<instance ID>`.
`--package-path`	N/A	Specify the directory of the package to install.

Option	Default value (if option is not passed)	Description
`--install-dir`	default installation directory	Customize the installation directory.
`--log-dir`	default log directory	Customize the directory to save the log file to.
`--proxy`	N/A	Specify proxy settings in the following format: `http://username:password@proxy-server.mycorp.com:3128`.
`--download-cache`	default download cache location	Point to the directory to store all downloaded and cached files.
`--download-dir`	default download directory	Customize the download directory, which is used in download-only mode.
`--intel-sw-improvement-program-consent`	`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	Show the detailed description of the Intel Software Improvement Program.
`--ignore-errors`	N/A	Complete installation even if non-critical errors occur (like errors in pre-/post-install scripts).
`-h, --help`	N/A	Show the installer help.
`-p, --property`	N/A	Pass additional custom options. For example, the string `-p=option1=value -p option2=value` gives two additional options.

For example, to show the installer help, use the following command:

```
w_[Toolkit Name]Kit_[version]_offline.exe -a -h
```

Non-interactive (Silent) Installation

1. Go to the directory where the installer is located.
2. Run the following command:

```
w_[Toolkit Name]Kit_[version]_offline.exe -s -a --silent --eula accept
```

In silent mode, integration into Visual Studio* is installed by default (if supported). You need to have the **Desktop development with C++** workload installed into each Visual Studio instance beforehand. To skip Visual Studio* integration, pass the following arguments to the installation command: `–p=NEED_VSXXXX_INTEGRATION=0`, where `XXXX` is the Visual Studio version. For example, to install a toolkit and skip integration into Visual Studio* 2019, use the following installation command:

```
w_[Toolkit Name]Kit_[version]_offline.exe -s -a --silent --eula accept -
p=NEED_VS2019_INTEGRATION=0
```

3. Once the installation is complete, verify that the toolkit is installed in the default directory: `C:\Program Files(x86)\Intel\oneAPI`.

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

Examples

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

```
w_[Toolkit Name]Kit_[version]_offline.exe -s -a --list-products
```

Example of output:

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

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

```
w_[Toolkit Name]Kit_[version]_offline.exe -s -a --list-components
```

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

```
w_[Toolkit Name]Kit_[version]_offline.exe -s -a --list-components --product-id
intel.oneapi.win.tbb.product --product-ver 2021.1.1-129
```

Example of output:

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

- Install specific Intel oneAPI Toolkit products and components:

```
w_[Toolkit Name]Kit_[version]_offline.exe -s -a --silent --eula accept --components
intel.oneapi.win.tbb.devel
```

Install Using Package Managers

You can install Intel oneAPI packages from one of the following repositories:

- Conda
- PIP
- NuGet
- Maven (oneDAL)

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 intel <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 intel <package_name>==<version>
```

For example: `conda install -c intel mkl==2021.1.1`

List of Available Packages

Component Name	Package Name	Platform	Dependencies
Intel® MPI Library	`impi_rt` `impi-devel`	win-x64	N/A
Intel® Fortran Compiler (Beta) and Intel® Fortran Compiler Classic	`intel-fortran-rt`	win-x64 win-x86	Intel® MPI Library Intel OpenMP* Runtime Library
Intel® CPU Runtime for OpenCL™ Applications	`intel-opencl-rt`	win-x64	oneTBB
Intel® oneAPI DPC++/C++ Compiler	`dpcpp-cpp-rt`	win-x64	Intel® CPU Runtime for OpenCL™ Applications Intel OpenMP* Runtime Library
Intel OpenMP* Runtime Library	`intel-openmp`	win-x64 win-x86	N/A
Intel® oneAPI DPC++ Library	`onedpl-devel`	win-x64	N/A
Intel® oneAPI Threading Building Blocks (oneTBB)	`tbb` `tbb-devel`	win-x64 win-x86	N/A
	`tbb4py`	win-x64	N/A

Intel® oneAPI Data Analytics Library (oneDAL)	`dal`
	`dal-static`
	`dal-devel`
	`dal-include`
		win-x64
		win-x86
			oneTBB

	`daal4py`	win-x64	oneDAL
Intel® Extension for Scikit-learn	`scikit-learn-intelex`	win-x64	oneDAL

Component Name	Package Name	Platform	Dependencies

Intel® Integrated Performance Primitives (Intel® IPP)	`ipp`
	`ipp-static`
	`ipp-include`
	`ipp-devel`
		`win-x64`
		`win-x86`
			N/A

Intel® Integrated Performance Primitives Cryptography	`ipp_crypto`
	`ipp_crypto_static`
	`ipp_crypto-include`
	`ipp_crypto-devel`
		`win-x64`
		`win-x86`
			N/A

Intel® oneAPI Math Kernel Library (oneMKL)	`mkl`
	`mkl-devel`
	`mkl-static`
	`mkl-include`
		`win-x64`
		`win-x86`
			Intel OpenMP* Runtime Library
			oneTBB

	`mkl-dpcpp` `mkl-devel-dpcpp`	`win-x64` `win-x86`	Intel OpenMP* Runtime Library oneTBB Intel® oneAPI DPC++/C++ Compiler Runtime Intel® CPU Runtime for OpenCL™ Applications
Intel® oneAPI Deep Neural Network Library (oneDNN)*	`onednn-cpu-vcomp` `onednn-devel-cpu-vcomp`	`win-x64`	N/A
	`onednn-cpu-iomp` `onednn-devel-cpu-iomp`	`win-x64`	Intel OpenMP* Runtime Library
	`onednn-cpu-dpcpp-gpu-dpcpp` `onednn-devel-cpu-dpcpp-gpu-dpcpp`	`win-x64`	Intel® CPU Runtime for OpenCL™ Applications Intel oneAPI DPC++/C++ Compiler Runtime
	`onednn-cpu-tbb` `onednn-devel-cpu-tbb`	`win-x64`	oneTBB

* - 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`.

- Install Intel® AI Analytics Toolkit via Conda*
  - List of Available Packages

Install Intel® AI Analytics Toolkit via Conda*

Intel provides access to the AI Kit components through a public Anaconda repository. If you do not have an existing Conda-based python environment, install Conda or Miniconda*. To get more details on the AI Analytics Toolkit, visit the Intel AI Analytics toolkit home page.

1. Activate the conda base environment via opening Anaconda prompt or executing the following command in command prompt:

```
C:\<conda-install-dir>\anaconda3\Scripts\activate.bat
```

2. Create and activate your conda environment:

```
conda create -n <conda environment name>
conda activate <conda environment name>
```

3. To install the packages from the Anaconda channel via Conda, execute the following command:

```
conda install <package_name>
```

For example, to install the TensorFlow* package, execute the following command:

```
conda install tensorflow
```

4. To install the packages from other channels, specify a channel name as in the example below:

```
conda install <package_name> -c <channel>
```

For example, to install the PyTorch* package from the pytorch channel, use the following command:

```
conda install pytorch -c pytorch
```

You can find the list of available packages and their channels in the section below.

List of Available Packages

Component Name	Package Name	Platform	Channel
Intel® Distribution for Python*	`intelpython3_full`	`win`	intel
Modin*	`modin-ray`	`win`	anaconda
Intel® Neural Compressor*	`neural-compressor`	`win`	intel
PyTorch*	`pytorch`	`win`	pytorch
TensorFlow*	`tensorflow`	`win`	anaconda

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`	win-x64	N/A
Intel® Fortran Compiler (Beta) and Intel® Fortran Compiler Classic	`intel-fortran-rt`	win-x64 win-x86	Intel® MPI Library; Intel OpenMP* Runtime Library
Intel® CPU Runtime for OpenCL™ Applications	`intel-opencl-rt`	win-x64	oneTBB
Intel® oneAPI DPC++/C++ Compiler	`dpcpp-cpp-rt`	win-x64	Intel® CPU Runtime for OpenCL™ Applications; Intel OpenMP* Runtime Library
Intel OpenMP* Runtime Library	`intel-openmp`	win-x64 win-x86	N/A
Intel® oneAPI Threading Building Blocks (oneTBB)	`tbb` `tbb-devel`	win-x64 win-x86	N/A
	`tbb4py`	win-x64	N/A
Intel® oneAPI Data Analytics Library (oneDAL)	`daal` `daal-static` `daal-devel`	win-x64	oneTBB

Component Name	Package Name	Platform	Dependencies
	daal-include	win-x86	
	daal4py	win-x64	oneDAL
Intel® Extension for Scikit-learn	scikit-learn-intelex	win-x64	oneDAL
Intel® Integrated Performance Primitives (Intel® IPP)	ipp ipp-static ipp-include ipp-devel	win-x64 win-x86	N/A
Intel® Integrated Performance Primitives Cryptography	ipp_crypto ipp_crypto_static ipp_crypto-include ipp_crypto-devel	win-x64 win-x86	N/A
Intel® oneAPI Math Kernel Library (oneMKL)	mkl mkl-devel mkl-static mkl-include	win-x64 win-x86	Intel OpenMP* Runtime Library oneTBB
	mkl-dpcpp mkl-devel-dpcpp	win-x64 win-x86	Intel OpenMP* Runtime Library oneTBB Intel® oneAPI DPC++/C++ Compiler Runtime Intel® CPU Runtime for OpenCL™ Applications
Intel® oneAPI Deep Neural Network Library (oneDNN)	onednn-cpu-vcomp onednn-devel-cpu-vcomp	win-x64	N/A
	onednn-cpu-iomp onednn-devel-cpu-iomp	win-x64	Intel OpenMP* Runtime Library
	onednn-cpu-dpcpp-gpu-dpcpp onednn-devel-cpu-dpcpp-gpu-dpcpp	win-x64	Intel® CPU Runtime for OpenCL™ Applications Intel oneAPI DPC++/C++ Compiler Runtime

Component Name	Package Name	Platform	Dependencies
	onednn-cpu-tbb onednn-devel-cpu-tbb	win-x64	oneTBB

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>	win-x64	N/A
Intel OpenMP* Runtime Library	intelopenmp.devel.<platform>	win	N/A
Intel® oneAPI Threading Building Blocks (oneTBB)	inteltbb.devel.<platform>	win	N/A
Intel® oneAPI Data Analytics Library (oneDAL)	inteldal.devel.<platform>	win-x64 win-x86	oneTBB
Intel® Integrated Performance Primitives (Intel® IPP)	intelipp.devel.<platform> intelipp.static.<platform>	win-x64 win-x86	N/A
Intel® Integrated Performance Primitives Cryptography	intelipp_crypto.devel.<platform> intelipp_crypto.static.<platform>	win-x64 win-x86	N/A
Intel® oneAPI Math Kernel Library (oneMKL)	intelmkl.devel.<platform>	win-x64	Intel® MPI Library

Component Name	Package Name	Platform	Dependencies
		win-x86	
Intel® oneAPI Math Kernel Library (Cluster Components)	intelmkl.devel.cluster.<platform> intelmkl.static.cluster.<platform>	win-x64	Intel OpenMP* Runtime Library oneMKL
Intel® oneAPI Deep Neural Network Library (oneDNN)	onednn.cpu_vcomp.devel.<platform> onednn.cpu_iomp.devel.<platform> onednn.cpu_tbb.devel.<platform>	win-x64	Intel OpenMP* Runtime Library oneTBB
Intel® oneAPI Video Processing Library (oneVPL)	onevpl.devel.<platform>	win-x64	N/A

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

The devel packages on Windows* also have a dependency on the runtime redist packages (see the next section).

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® MPI Library	intelmpi.redist.<platform>	win-x64
Intel OpenMP* Runtime Library	intelopenmp.redist.<platform>	win
Intel® oneAPI Threading Building Blocks (oneTBB)	inteltbb.redist.<platform>	win
Intel® oneAPI Data Analytics Library (oneDAL)	inteldal.redist.<platform>	win-x64 win-x86
Intel® Integrated Performance Primitives (Intel® IPP)	intelipp.redist.<platform>	win-x64 win-x86
Intel® Integrated Performance Primitives Cryptography	intelipp_crypto.redist.<platform>	win-x64 win-x86

Component Name	Package Name	Platform Availability
Intel® oneAPI Math Kernel Library (oneMKL)	`intelmkl.redist.<platform>`	`win-x64` `win-x86`
Intel® oneAPI Math Kernel Library (Cluster Components)	`intelmkl.redist.cluster.<platform>`	`win-x64`
Intel® oneAPI Deep Neural Network Library (oneDNN)	`onednn.cpu_vcomp.redist.<platform>` `onednn.cpu_iomp.redist.<platform>` `onednn.cpu_tbb.redist.<platform>`	`win-x64`
Intel® oneAPI Video Processing Library (oneVPL)	`onevpl.runtime.<platform>`	`win-x64`

Maven

This page provides general instructions on how to include Intel® oneAPI Data Analytics Library (oneDAL) packages from the Maven repository into your Java project.

To enable oneDAL in your project, specify the following artifacts in your build automation tool:

```
Group ID: com.intel.dal
Version: <version>
Artifact ID: dal
```

where `<version>` is a valid component version, for example, 2021.2.0.123.

For more information on the Maven dependency mechanism, refer to the Maven documentation.

Uninstall oneAPI Toolkits and Components

Uninstall Using GUI

Use the **Add/Remove Programs** option from the Control Panel to uninstall oneAPI Toolkits.

Uninstall with Command Line

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

```
w_[Toolkit Name]Kit_[version].exe -s -a --list-products
```

2. Uninstall the selected product:

```
cd C:\Program Files\Intel\oneAPI\Installer
installer.exe -s --action remove --product-id intel.oneapi.win.tbb.product --product-ver
2021.1.1-129
```

Troubleshooting

Integration into Visual Studio* fails for oneAPI 2022.1 and lower versions of toolkits

During installation of 2022.1 and lower versions of toolkits, integration into Visual Studio* fails on systems with Visual Studio* 2022 installed.

Cause: Incompatibility between oneAPI installers and recent changes in the `Microsoft.VisualStudio.Setup.Configuration.Native.dll` library provided by Microsoft as part of the Visual Studio 2022 installation.

Solution: Upgrade to the latest version of the oneAPI toolkit. Alternatively, before installing 2022.1 and lower toolkit packages, remove Visual Studio* 2022 from your system. For more information about the available workarounds, refer to Known Microsoft* Visual Studio 2022 and oneAPI Toolkits Installation Issue.

OpenCL CPU Runtime Library fails to load Intel oneTBB

Intel OpenCL CPU Runtime Library fails to load the Intel oneTBB library that was installed via Conda.

Cause: No oneTBB registry key installed when Intel oneTBB is installed via Conda package manager.

Solution: Before completing the workaround steps below, make sure that you already configured the targeted device, otherwise, OpenCL Runtime will load two devices, and use the first one by default.

1. Locate the corresponding device configuration file in the same folder where the OpenCL Runtime Library file `intelocl64.dll` resides:
   - `cl.cfg` for CPU device
   - `cl.fpga_emu.cfg` for FPGA emulator
2. Open the configuration file for editing and add your oneTBB DLL path to the `CL_CONFIG_TBB_DLL_PATH` field:

```
CL_CONFIG_TBB_DLL_PATH = <tbb-install-dir>
```

NOTE

- If you configure both `cl.cfg` and `cl.fpga_emu.cfg`, the oneTBB DLL path in them should be the same value
- Add the value of the `CL_CONFIG_TBB_DLL_PATH` field without quotation marks

Integrity check fails

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.

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:

- 32-bit system: `C:\Program Files\Intel\oneAPI\Installer` and `C:\Program Files\Intel\PackageManager\1.0`
- 64-bit system: `C:\Program Files (x86)\Intel\oneAPI\Installer` and `C:\Program Files (x86)\Intel\PackageManager\1.0`

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

Notices and Disclaimers