**List of Intel(R) DPC++ Compatibility Tool specific options:**

--always-use-async-handler - Always create the cl::sycl::queue with an async

exception handler. Default: off.

--assume-nd-range-dim=<value> - Provides a hint to the tool on the dimensionality

of nd\_range to use in generated code. The values are:

=1 - Generate kernel code assuming 1D nd\_range

where possible, and 3D in other cases.

=3 - Generate kernel code assuming 3D nd\_range

(default).

--build-script-file=<file> - Specifies the name of generated makefile for

migrated file(s). Default name: Makefile.dpct.

--check-unicode-security - Enable detection and warnings about Unicode

constructs that can be exploited by using

bi-directional formatting codes and homoglyphs in

identifiers. Default: off. --comments

- Insert comments explaining the generated code.

Default: off.

--cuda-include-path=<dir> - The directory path of the CUDA header files.

--custom-helper-name=<name> - Specifies the helper headers folder name and main

helper header file name. Default: dpct.

--enable-ctad - Use a C++17 class template argument deduction

(CTAD) in your generated code. Default: off.

--extra-arg=<string> - Additional argument to append to the migration

command line, example:

--extra-arg="-I /path/to/header”. The options that

can be passed this way can be found with the

dpct -- -help command.

--format-range=<value> - Sets the range of formatting. The values are:

=migrated - Only formats the migrated code (default).

=all - Formats all code.

=none - Do not format any code.

--format-style=<value> - Sets the formatting style. The values are:

=llvm - Use the LLVM coding style.

=google - Use the Google coding style.

=custom - Use the coding style defined in the .clang-format file

(default).

--gen-build-script - Generates makefile for migrated file(s) in -out-root

directory. Default: off.

--help - Provides list of Intel(R) DPC++ Compatibility Tool

specific options.

--in-root=<dir> - The directory path for the root of the source tree that

needs to be migrated. Only files under this root are

migrated. Default: Current directory, if input source

files are not provided. If input source files are

provided, the directory of the first input source file is

used.

--in-root-exclude=<dir|file> - Excludes the specified directory or file from

processing.

--keep-original-code - Keeps the original code in comments of generated

DPC++ files. Default: off.

--no-cl-namespace-inline - DEPRECATED: Do not use cl:: namespace inline.

Default: off. This option will be ignored if the

replacement option --use-explicit-namespace is used.

--no-dpcpp-extensions=<value> - Comma separated list of DPC++ extensions not to

be used in migrated code. By default, these

extensions will be used in migrated code.

=enqueued\_barriers - Enqueued barriers DPC++ extension.

--no-dry-pattern - Do not use DRY (do not repeat yourself) pattern

when functions from dpct namespace are inserted.

Default: off.

--no-incremental-migration - Tells the tool to not perform an incremental migration.

Default: off (incremental migration happens).

--optimize-migration - Generates DPC++ code applying more aggressive

assumptions that potentially may alter the

semantics of your program. Default: off.

--out-root=<dir> - The directory path for root of generated files. A

directory is created if it does not exist. Default: dpct\_output.

--output-file=<file> - Redirects the stdout/stderr output to <file> in the

output directory specified by the --out-root option.

--output-verbosity=<value> - Sets the output verbosity level:

=silent - Only messages from clang.

=normal - 'silent' and warnings, errors, and notes from dpct.

=detailed - 'normal' and messages about which file is being processed.

=diagnostics - 'detailed' and information about the detected

conflicts and crashes. (default)

-p=<dir> - The directory path for the compilation database

(compile\_commands.json). When no path is

specified, a search for compile\_commands.json is

attempted through all parent directories of the first

input source file.

--process-all - Migrates or copies all files, except hidden, from

the --in-root directory to the --out-root directory.

The --in-root option should be explicitly specified.

Default: off.

--report-file-prefix=<prefix> - Prefix for the report file names. The full file name

will have a suffix derived from the report-type and

an extension derived from the report-format. For

example: <prefix>.apis.csv or <prefix>.stats.log. If

this option is not specified, the report will go to

stdout. The report files are created in the directory,

specified by -out-root.

--report-format=<value> - Format of the reports:

=csv - Output is lines of comma separated values. The

report file name extension will be .csv. (default)

=formatted - Output is formatted to be easier to read for human

eyes. Report file name extension will be log.

--report-only - Only reports are generated. No DPC++ code is

generated. Default: off.

--report-type=<value> - Specifies the type of report. Values are:

=apis - Information about API signatures that need

migration and the number of times they were

encountered. The report file name will have .apis

suffix added.

=stats - High level migration statistics: Lines Of Code (LOC)

that are migrated to DPC++, LOC migrated to

DPC++ with helper functions, LOC not needing

migration, LOC needing migration but are not

migrated. The report file name has the .stats suffix

added (default)

=all - All of the reports.

--rule-file=<file> - Specifies the rule file path that contains rules used

for migration.

--stop-on-parse-err - Stop migration and generation of reports if parsing

errors happened. Default: off.

--suppress-warnings=<value> - Comma separated list of migration warnings to

suppress. Valid warning IDs range from 1000 to

1096. Hyphen separated ranges are also allowed.

For example: --suppress-warnings=1000-1010,1011

--suppress-warnings-all - Suppresses all migration warnings. Default: off.

--sycl-named-lambda - Generates kernels with the kernel name. Default: off.

--use-custom-helper=<value> - Customize the helper header files for migrated code.

The values are:

=none - No customization (default).

=file - Limit helper header files to only the necessary files

for the migrated code and place them in the

--out-root directory.

=api - Limit helper header files to only the necessary APIs

for the migrated code and place them in the

--out-root directory.

=all - Generate a complete set of helper header files and

place them in the --out-root directory.

--use-experimental-features=<value> - Comma separated list of experimental features to

be used in migrated code. By default, experimental

features will not be used in migrated code. **The values are**:

=nd\_range\_barrier - Experimental helper function used to help cross

group synchronization during migration.

=free-function-queries - Experimental extension that allows getting ‘id’, ‘item’,

‘nd\_item’, ‘group’, and ‘sub\_group’ instances globally.

=local-memory-kernel-scope-allocation - Experimental extension that allows allocation of

local memory objects at the kernel functor scope

--use-explicit-namespace=<value> - Defines the namespaces to use explicitly in

generated code. The value is a comma separated

list. Default: dpct, sycl. Possible values are:

=none - Generate code without namespaces. Cannot be

used with other values.

=cl - Generate code with **cl::sycl::namespace**. Cannot be

used with sycl or sycl-math values.

=dpct - Generate code with **dpct::namespace**.

=sycl - Generate code with **sycl::namespace**. Cannot be

used with cl or sycl-math values.

=sycl-math - Generate code with **sycl::namespace**, applied only

for SYCL math functions. Cannot be used with cl or

sycl values.

--usm-level=<value> - Sets the USM level to use in source code generation.

=restricted - Uses API from DPC++ Explicit and Restricted Unified

Shared Memory extension for memory management

migration. (default)

=none - Uses helper functions from DPCT header files for

memory management migration.

--vcxprojfile=<file> - The file path of vcxproj.

--version - Shows the version of the tool.

**<source0> ... Paths of input source files. These paths are looked up in the compilation database.**

**EXAMPLES**:

***Migrate single source file:***

dpct source.cpp

***Migrate single source file with C++11 features:***

dpct --extra-arg="-std=c++11" source.cpp

***Migrate all files available in compilation database:***

dpct -p=<path to location of compilation database file>

***Migrate one file in compilation database:***

dpct -p=<path to location of compilation database file> source.cpp

***Migrate all files available in vcxprojfile:***

dpct --vcxprojfile=path/to/vcxprojfile.vcxproj

See [Diagnostics Reference](https://software.intel.com/content/www/us/en/develop/documentation/intel-dpcpp-compatibility-tool-user-guide/top/diagnostics-reference.html) to resolve warnings and complete the migration:

https://software.intel.com/content/www/us/en/develop/documentation/intel-dpcpp-compatibility-tool-user-guide/top/diagnostics-reference.html