Visual Studio Add-in

The Ice Visual Studio Add-in manages all aspects of code generation for C++ and .NET projects, including automatically recompiling Slice files that have changed, removing obsolete generated files, and tracking dependencies.

On this page:

Activating the Add-in for a Visual Studio Project

Right-click on the project in the Solution Explorer and choose *Ice Configuration...* or select *Ice Configuration...* in the *Tools* menu. This opens a dialog that allows you to configure the project settings for Ice.

> ⓘ Note that, after adding new configurations or platforms to your project, it may be necessary to disable and then re-enable the add-in for the new configuration or platform to get the correct Ice settings.

Configuring Project Settings for Visual Studio

The following settings are available:

- **Enable Ice Builder**
  This box must be checked to enable the Ice Builder in your project.

- **Trace Level**
  You can change the verbosity of messages printed to the *Output* panel by selecting a different trace level, where **Errors Only** is less verbose and **Debug** is more verbose.

- **Output directory for generated files**
  Set the base directory where generated files will be placed.

- **Slice compiler options**
  Tick the corresponding check boxes to pass `--tie` (.NET only), `--ice`, `--checksum`, or `--stream` options to the Slice compiler.

- **Additional Slice compiler options**
  Enter extra Slice compiler options not supported by **Slice compiler options**. These options are entered the same as they would be on the command line. For example, you can define preprocessor macros by entering `-DFoo -DBAR`.

- **Macro for exporting symbols from DLL** (C++ only)
  The macro for annotating generated code to mark symbols that should be exported from a shared library or DLL, equivalent to the compiler option `--dll-export`.

- **Slice include directories**
  The list of directories to search for included Slice files (`-I` option). The checkbox for each path controls whether the path is passed to the `-I` option as an absolute path or as a path relative to the project directory.

  > ⓘ The add-in automatically adds the main `slice` directory of your Ice installation to this list.

- **Link project with these additional libraries** (C++ only)
  Every Ice project is linked with the `Ice` and `IceUtil` libraries. You can link with additional Ice libraries by checking the appropriate boxes.

- **Add references to the following assemblies** (.NET only)
  Every Ice project references the `Ice` assembly. You can reference additional Ice assemblies by checking the appropriate boxes.

Using Environment Variables in Visual Studio Settings

You can include references to environment variables in the settings for **Output directory for generated files**, **Additional Slice compiler options**, and **Slice include directories**. To do this, the environment variable must be entered using the format $(VAR). For example, for **Slice include directories** you could use $(MY_PATH).

You cannot use environment variables in the --header-ext and --source-ext options in **Additional Slice compiler options**.

Adding Slice Files to a Visual Studio Project

Use *Add -> New Item...* and select *Slice File (.ice)* as the template to create and add a Slice file to a project. To add an existing Slice file to a project, use *Add -> Existing Item...*

Generating Code for a Visual Studio Project

The add-in compiles a Slice file whenever you save the file. The add-in tracks dependencies among Slice files in the project and, after a change, recompiles only the affected files.

Generated files are automatically added to the project. For example, for Demo.ice, the add-in for C++ adds Demo.cpp and Demo.h to the project, whereas the add-in for .NET adds Demo.cs to the project.

Slice compilation errors are displayed in the Visual Studio *Output* and *Error List* panels.

How the Add-in Locates your Ice Installation

The add-in derives the location of your Ice installation from the location of its own DLL. For example, if the add-in's DLL is installed in C:\Ice\vsaddin then the add-in uses C:\Ice as the top-level Ice installation directory. The add-in uses this top-level directory to compose the path names of other subdirectories, such as C:\Ice\slice as the location of the Slice files included in your Ice distribution, C:\Ice\bin as the location of the Slice compilers, and so on.

$(IceHome) Macro

The add-in makes extensive use of the $(IceHome) macro in C++ projects. $(IceHome) provides the full path to the home directory of the Ice installation on the local computer. This macro may be used in user settings, for example to locate slice2freeze.exe (as $(IceHome)\bin\slice2freeze.exe) or the Ice Slice directory (as $(IceHome)\slice).

This macro is set through a Visual Studio C++ property sheet installed as part of the Ice installation on Windows, %AllUsersProfile%\ZeroC\ice.vsprops for Visual Studio 2008 and %AllUsersProfile%\ZeroC\ice.props for Visual Studio 2010.

.NET DEVPATH support

The add-in detects if a .NET project is configured for development mode by inspecting the *<application-name>*.exe.config file. If a project is in development mode, the Ice bin directory is automatically added to the DEVPATH environment variable when the demo is run. References to Ice components are also set with *Copy Local* to false to avoid copying Ice assemblies to the project's output directory. Note that the *Copy Local* setting is not changed for references that are already added to the project.

Command-line Builds using Visual Studio

The add-in supports command-line builds using devenv. For example:

```
devenv MyProject.sln /build
```

Note that for this to work, command-line builds must be enabled for the add-in in the IDE; see *Tools -> Add-in Manager* and check *Command Line* for Ice.

See Also

- slice2cpp Command-Line Options
- slice2cs Command-Line Options