

Using the Windows Binary Distributions

This page provides important information for users of the Ice binary distributions on Windows platforms.

On this page:

- [Overview of the Ice Binary Distributions for Windows](#)
- [NuGet Package Installation](#)
 - [Adding a NuGet Package to a Visual Studio Project](#)
 - [ZeroC Symbol Server](#)
 - [Debug Symbols and Stack Traces for C++ Applications](#)
- [Using the NuGet Packages](#)
 - [Information for C++ Developers](#)
 - [Information for C++/CX UWP Developers](#)
 - [Information for C# and .NET Developers](#)
- [Using the Ice MSI Installation](#)
 - [Information for PHP Developers](#)
 - [Configuration Files for IceGrid and Glacier2 Services](#)
 - [Starting IceGrid GUI on Windows](#)
 - [Unattended Installation](#)
 - [Registry Key](#)
- [Using the Sample Programs on Windows](#)

Overview of the Ice Binary Distributions for Windows

Ice 3.7 provides the following binary distributions for Windows:

- A traditional Windows installer file, ice-3.7.0.msi
- Several [NuGet](#) packages: zeroc.ice.v100, zeroc.ice.v120, zeroc.ice.v140, zeroc.ice.v141, zeroc.ice.uwp.v140, zeroc.ice.uwp.v141 and zeroc.ice.net

The table below shows which distribution(s) you should install depending on your needs.

Activity	Compiler or Environment	Distribution to Install	Files Installed
Develop C++ desktop applications	Visual Studio 2010	zeroc.ice.v100	The complete Ice C++ SDK for the selected compiler, with x86, x64, Debug and Release binaries
	Visual Studio 2013	zeroc.ice.v120	All Slice compilers (slice2cpp, slice2cs, slice2java, etc.)
	Visual Studio 2015	zeroc.ice.v140	All Ice services (Glacier2, IceBridge, IceGrid, IcePatch2, IceStorm), and the associated command-line utility tools (icegridadmin, icestormadmin, etc.)
	Visual Studio 2017	zeroc.ice.v141	
Develop C++/CX UWP applications	Visual Studio 2015	zeroc.ice.uwp.v140	The complete Ice C++/CX SDK for the selected compiler, with x86, x64, Debug and Release binaries
	Visual Studio 2017	zeroc.ice.uwp.v141	slice2cpp, slice2html
Develop C# or other .NET applications	Visual Studio 2013, 2015 or 2017	zeroc.ice.net	The complete Ice C#/.NET SDK slice2cs, slice2html
Develop PHP applications	PHP 7.1	Ice MSI	The IceGrid GUI admin tool
Develop Java applications	JDK 8	Ice MSI	All the Slice compilers (slice2java, slice2php, slice2html, etc.)
Administer IceGrid deployments		Ice MSI	All the Ice Slice files
Deploy Ice services (Glacier2, IceBridge, IceGrid, IcePatch2, IceStorm)		Ice MSI	The Ice for PHP extension (for PHP 7.1) All Ice services (Glacier2, IceBridge, IceGrid, IcePatch2, IceStorm), and the associated command-line utility tools (icegridadmin, icestormadmin, etc.) (x64 Release only)



Visual Studio

If you are developing applications with Visual Studio, you should also install the [Ice Builder for Visual Studio](#).

With Visual Studio 2010 and Visual Studio 2017, you can only build with MSBuild from the command-line. The integration with the Visual Studio 2017 IDE is not yet functional.



C++ run time

Ice MSI

All the C++ binaries installed by the Ice MSI are x64 Release, built with Visual Studio 2015. The Ice MSI installs the corresponding Visual C++ run time on your computer: you don't need to install Visual Studio or any Visual Studio Redistributable to use this distribution.

Ice NuGet packages

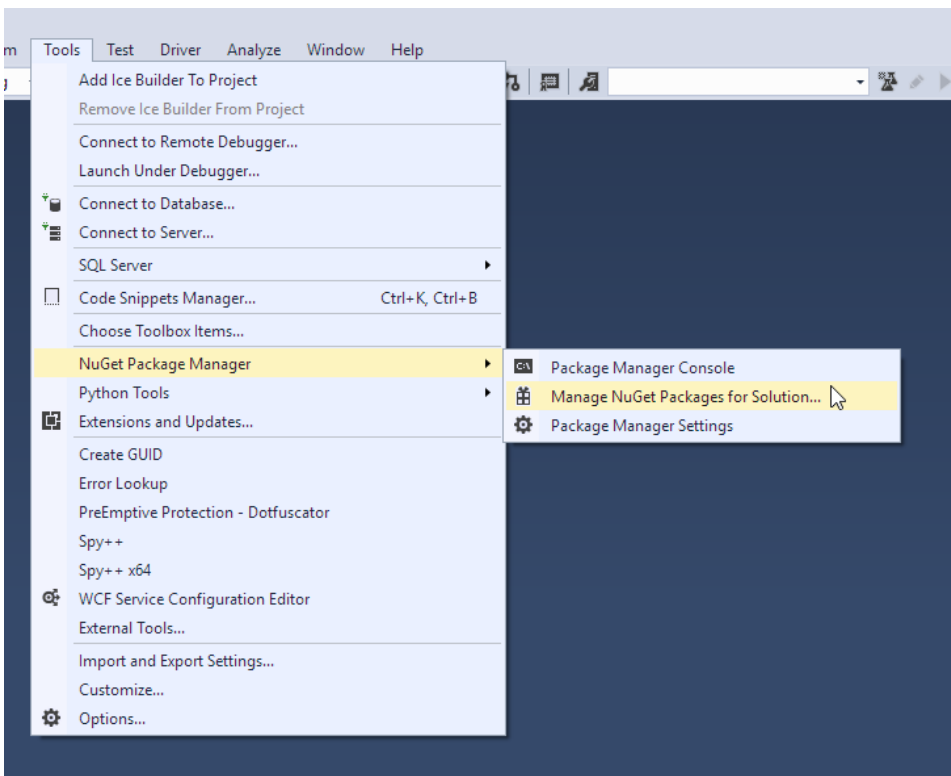
The Ice NuGet packages do not install the Visual C++ run time on your computer.

NuGet Package Installation

Adding a NuGet Package to a Visual Studio Project

Follow these steps to install Ice NuGet packages on your computer.

1. Open the Visual Studio Solution that contains the project you want to work on.
2. Open the NuGet Package Manager from the Tools menu:



3. On the next screen, select the zeroc.ice package you want to install, then select the project in which you want to install it, and finally click the Install button.
NuGet will install this package in the `packages` folder next to your Solution file and configure the selected project to use it.

NuGet - Solution

Browse Installed Updates Consolidate

zeroc.ice. Include prerelease Package source: nuget.org

zeroc.ice.uwp.v140 by ZeroC v3.7.0

Ice C++/CX SDK for Visual Studio 2015 (v140). Provides binaries for Universal Windows (UWP) x86 and x64. Ice is a comprehensive RPC framework that hel...

zeroc.ice.v100 by ZeroC v3.7.0

Ice C++ SDK for Visual Studio 2010 (v100). Ice is a comprehensive RPC framework that helps you network your software with minimal effort.

zeroc.ice.net by ZeroC v3.7.0

Ice C#/.NET SDK. Ice is a comprehensive RPC framework that helps you network your software with minimal effort.

zeroc.ice.v120 by ZeroC v3.7.0

Ice C++ SDK for Visual Studio 2013 (v120). Ice is a comprehensive RPC framework that helps you network your software with minimal effort.

zeroc.ice.v140 by ZeroC v3.7.0

Ice C++ SDK for Visual Studio 2015 (v140). Ice is a comprehensive RPC framework that helps you network your software with minimal effort.

zeroc.ice.v141 by ZeroC v3.7.0

Ice C++ SDK for Visual Studio 2017 (v141). Ice is a comprehensive RPC framework that helps you network your software with minimal effort.

zeroc.ice.uwp.v141 by ZeroC v3.7.0

Ice C++/CX SDK for Visual Studio 2017 (v141). Provides binaries for Universal

zeroc.ice.v141

Version(s) - 1

Project ^	Version
<input type="checkbox"/> MyFirstIceProject	3.7.0

Installed: 3.7.0 Uninstall

Version: Latest stable 3.7.0 Install

Options

Description

Ice C++ SDK for Visual Studio 2017 (v141). Ice is a comprehensive RPC framework that helps you network your software with minimal effort.

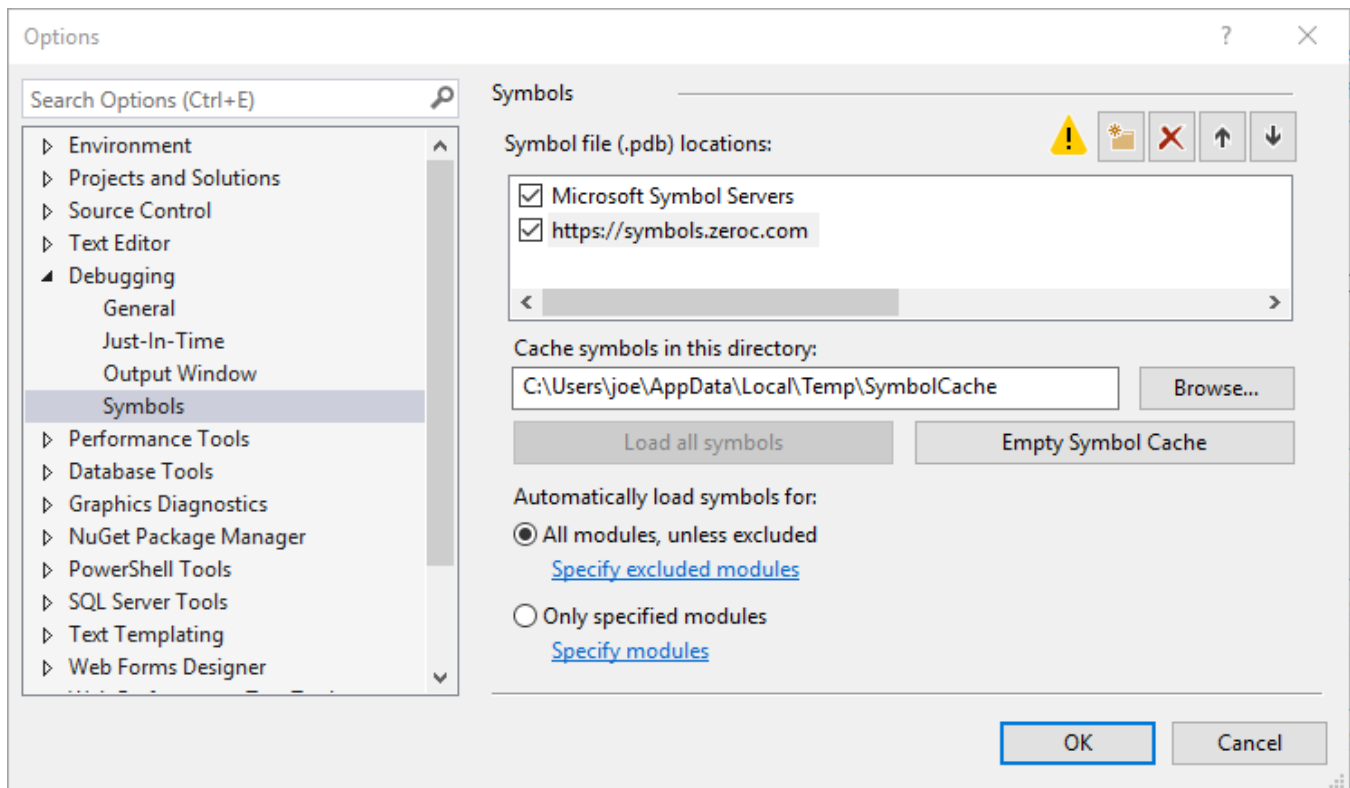
The screenshot above shows a solution with a C++ project. The process is the same with UWP and .NET projects.

[Back to Top ^](#)

ZeroC Symbol Server

The ZeroC symbol server, <https://symbols.zeroc.com>, provides debug symbols for the C++ binaries included in the NuGet packages published by ZeroC.

To use this symbol server, add its URL on Visual Studio's Symbols page:



Check "Enable source server support" in Visual Studio's debugging options to allow Visual Studio to fetch the corresponding source code.



The `zeroc.ice.net` and `zeroc.ice.uwp` NuGet packages include debug symbols (pdb files) and don't need this Symbol Server.

[Back to Top ^](#)

Debug Symbols and Stack Traces for C++ Applications

Ice C++ applications can print stack traces for Ice exceptions, which can be very helpful for debugging. In order to get usable stack traces, you need to install the corresponding Ice debug symbols in a local folder, and configure your system to look for debug symbols in this folder. We recommend you do the following:

1. Download and install the [Debugging Tools for Windows](#).
2. Add these tools to your PATH, for example:

```
set PATH=C:\Program Files (x86)\Windows Kits\10\Debuggers\x64;%PATH%
```

3. Download the debug symbols of your Ice NuGet package(s) to your local `SymbolCache` folder, with the `symchk` tool included in the Debugging Tools for Windows. For example:

```
symchk /v /r packages\zeroc.ice.v140.3.7.0\build\native\bin\* /s SRV*%TEMP%\SymbolCache*https://symbols.zeroc.com
```

(remove the `/v` option for a quieter output)

This copies debug symbols for all the binaries in the selected NuGet packages to `%TEMP%\SymbolCache`. `%TEMP%\SymbolCache` is also the default symbol cache folder for Visual Studio. This folder can cache symbols for any number of NuGet packages, in particular the debug symbols of different versions of the same package.

4. Set the environment variable `_NT_SYMBOL_PATH` to point to this local `SymbolCache` folder, for example:

```
set _NT_SYMBOL_PATH=%TEMP%\SymbolCache
```

[Back to Top ^](#)

Using the NuGet Packages

Information for C++ Developers

Once you've installed the Ice NuGet package into a C++ project as shown earlier, this project will find automatically all Ice C++ header files and import libraries. If you also enable the Ice Builder for Visual Studio in this C++ project, the Ice Builder will take care of compiling the Slice files in this project with `slice2cpp` (it uses the `slice2cpp` installed from the NuGet package).

Moreover, the Debugger Path is set and you can run your application directly from Visual Studio - there is no need to set any additional environment variables.

Occasionally, you may want to run your application outside Visual Studio, or use one of the Ice services or Ice tools included installed by the NuGet package. To do so, add the NuGet package's `bin` folder to your `PATH`.



The `zeroc.ice.v<version>` package is installed in the `packages` folder next to your Visual Studio solution file. For example, a Visual Studio 2017 solution would have a structure similar to the following:

```
C:\MyApplication\MyApplication.sln
C:\MyApplication\MyApplication\MyApplication.vcxproj
C:\MyApplication\packages\zeroc.ice.v141.3.7.0
```

With this example, you could set your `PATH` to:

```
set PATH=C:\MyApplication\packages\zeroc.ice.v141.3.7.0\build\native\x64\Release;%PATH%
```

[Back to Top ^](#)

Compiler Settings

Your application must be compiled with the same flags as the Ice libraries:

- Release: `/MD /EHsc`
- Debug: `/MDd /EHsc`

Add `ICE_CPP11_MAPPING` to the project preprocessor definitions if you want to use the [Ice C++11 mapping](#). Without this definition, you will use the [Ice C++98 mapping](#).

You don't need to list Ice import libraries such as `iceD.lib` when linking with Ice libraries. See [Linking with C++ Libraries on Windows](#) for additional details.

[Back to Top ^](#)

NuGet Package Details

The following table shows the Ice C++ NuGet package layout:

Folder	Description
<code>build\native\include</code>	C++ header files
<code>build\native\lib\<Platform>\<Configuration></code>	C++ import libraries
<code>build\native\bin\<Platform>\<Configuration></code>	C++ binaries (excluding Slice compilers)
<code>tools</code>	<code>slice2xxx</code> compilers
<code>build\native</code>	Visual Studio property and target files

slice	Slice files
-------	-------------

Installing the NuGet package imports the property and target files from the `build\native` folder into the project. The property file defines properties used by the Ice Builder for Visual Studio; you can also use them in custom build steps.

The table below presents these properties:

Name	Value	Description
IceVersion	3.7.0	Ice string version
IceIntVersion	30700	Ice version as a numeric value
IceVersionMM	3.7	Major Minor version
IceSoVersion	37	Version used in dynamic libraries
IceNugetPackageVersion	3.7.0	NuGet package version
IceHome	\$(MSBuildThisFileDirectory)..\..	Full path to the package root folder
IceToolsPath	\$(IceHome)\tools	Full path to the folder of the Slice compilers

The targets file configures the C++ Additional Include Directories and Additional Library Directories to locate C++ headers and import libraries in the package's include and lib folders.



The NuGet package automatically adds its `build\native\lib\<Platform>\<Configuration>` folder to the Additional Library Directories, where `<Platform>` is the selected platform and `<Configuration>` is Debug when the MSBuild property `UseDebugLibraries` is true. Otherwise, `<Configuration>` is Release. Projects created with recent versions of Visual Studio set `UseDebugLibraries` to true automatically for debug projects (meaning projects that link with the Visual C++ debug run-time libraries); if you created your project with an older version of Visual Studio, you need to edit the project file and set `UseDebugLibraries` to true for your debug configurations, for example:

```
<PropertyGroup Condition="'$(Configuration)|$(Platform)'=='Debug|x64'" Label="Configuration">
  ...
  <UseDebugLibraries>true</UseDebugLibraries>
</PropertyGroup>
```

[Back to Top ^](#)

Information for C++/CX UWP Developers

Once you've installed the Ice NuGet package into a C++/CX project as shown earlier, this project will find automatically all Ice C++ header files and static libraries. If you also enable the Ice Builder for Visual Studio in this C++ project, the Ice Builder will take care of compiling the Slice files in this project with `slice2cpp` (it uses the `slice2cpp` installed from the NuGet package).

Compiler Settings

Ice for C++/CX UWP supports only the new [C++11 mapping](#) and requires that you add `ICE_CPP11_MAPPING` to your project's preprocessor definitions.

The Release versions of the Ice libraries are compiled with `/MD` to select the multi-threaded Visual C++ run-time library, while the Debug versions use `/MDd` to select the debug multi-threaded run-time library. Both versions of the Ice libraries are compiled with `/EHsc` to select an [exception handling model](#).

Your application must be compiled with the same flags as the Ice libraries:

- Release: `/MD /EHsc /DICE_CPP11_MAPPING`
- Debug: `/MDd /EHsc /DICE_CPP11_MAPPING`

You don't need to list Ice import libraries such as `iceD.lib` when linking with Ice libraries. See [Linking with C++ Libraries on Windows](#) for additional details.

[Back to Top ^](#)

C++/CX SDK Nuget Package Details

The following table shows the Ice C++/CX NuGet package layout:

Folder	Description
build\native\include	C++ header files
build\native\lib\<Platform>\<Configuration>	C++ static libraries
tools	slice2cpp and slice2html
build\native	Visual Studio property and target files
slice	Slice files

Installing the NuGet package imports the property and target files from the `build\native` folder into the project. The property file defines properties used by the Ice Builder for Visual Studio; you can also use them in custom build steps.

The table below presents these properties:

Name	Value	Description
IceVersion	3.7.0	Ice string version
IceIntVersion	30700	Ice version as numeric value
IceVersionMM	3.7	Major Minor version
IceSoVersion	37	Version used in dynamic libraries
IceNugetPackageVersion	3.7.0	NuGet package numeric version
IceHome	\$(MSBuildThisFileDirectory)..\..	Full path to the package root folder
IceToolsPath	\$(IceHome)\tools	Full path to the folder containing the Slice compilers.

The targets file configures the C++ Additional Include Directories and Additional Library Directories to locate C++ headers and import libraries in the package's `include` and `lib` folders.

[Back to Top ^](#)

Information for C# and .NET Developers

Once you've installed the Ice NuGet package into a C# or .NET project as shown earlier, this project will gain a reference to the Ice assembly. If you also enable the Ice Builder for Visual Studio in this C# or .NET project, the Ice Builder will take care of compiling the Slice files in this project with `slice2cs` (it uses the `slice2cs` compiler installed from the NuGet package). The Ice Builder also simplifies the management of references to Ice assemblies included in the NuGet package.

You can develop Ice applications with any .NET CLR language, however, `slice2cs` only generates C# code. If you are developing in another CLR language, you can put the generated code in a library and reference this library from your project.

[Back to Top ^](#)

.NET SDK NuGet Package Details

The following table shows the Ice for .NET (zeroc.ice.net) NuGet package layout:


Folder	Description
lib	.NET assemblies
tools	slice2cs, slice2html, iceboxnet and bzip2 dll
build	Visual Studio property file
slice	Slice files

Installing the NuGet package imports the property file from the `build` folder into the project. The property file defines properties used by the Ice Builder for Visual Studio; you can also use them in custom build steps.

The table below presents these properties:

Name	Value	Description
IceVersion	3.7	Ice string version
IceIntVersion	30700	Ice version as a numeric value

IceVersionMM	3.7	Major Minor version
IceSoVersion	37	Version used in dynamic libraries
IceNugetPackageVersion	3.7.0	Nuget package numeric version
IceHome	\${MSBuildThisFileDirectory}..\..	Full path to the package root folder
IceToolsPath	\${IceHome}\tools	Full path to the folder containing the Slice compilers.

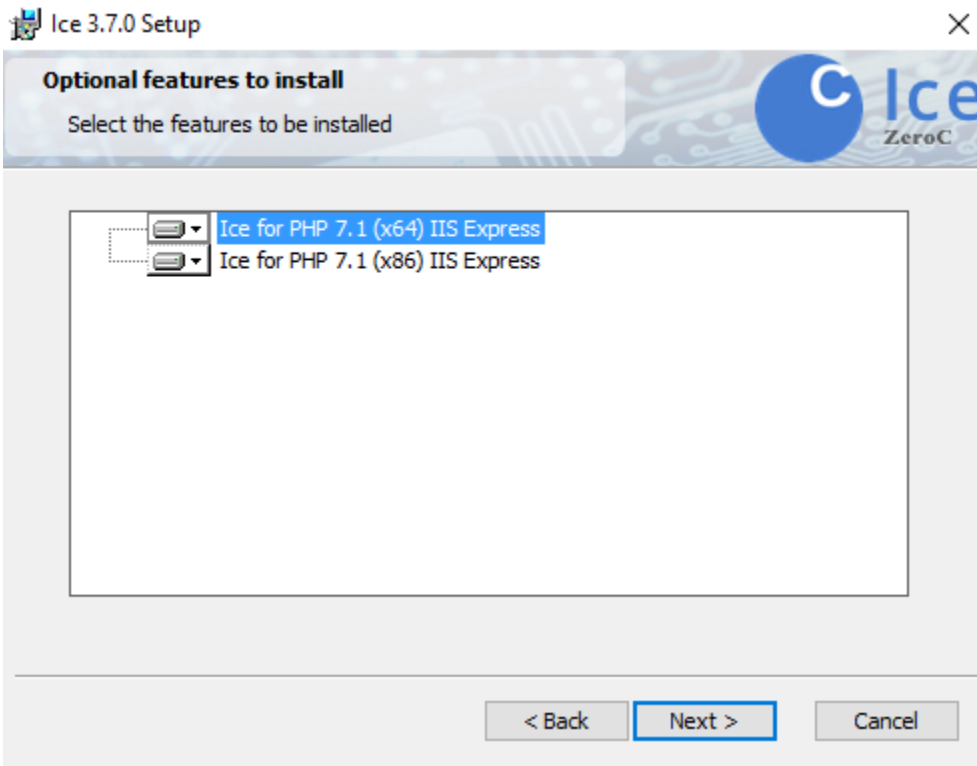
 The main Ice for .NET assembly (ice.dll) included in zeroc.ice.net uses unmanaged code. If you require only managed code, you can clone the [ice repository](#) and build Ice for .NET in a purely managed version.

[Back to Top ^](#)

Using the Ice MSI Installation

Information for PHP Developers

The MSI distribution includes all the components required to develop Ice for PHP applications on Windows, including PHP and Slice source files, the slice2php compiler and the Ice for PHP extension. The MSI installer detects the PHP installations on your computer and allows you to select where to install the Ice for PHP extension:



For each PHP version you select, the installer copies the Ice for PHP extension to the `extensions` folder and updates the `php.ini` configuration file to load the extension; it also modifies `include_path` to include the Ice for PHP folder. The following lines are added to `php.ini`:

```
[PHP_ZEROC_ICE]
extension=php_ice_nts.dll
include_path=${include_path}";C:\Program Files\ZeroC\Ice-3.7.0\php"
```

The Ice MSI includes both x86 and x64 versions of the Ice for PHP extension built with the PHP 7.1 NTS libraries.

[Back to Top ^](#)

Configuration Files for IceGrid and Glacier2 Services

The `config` subdirectory of the Ice MSI installation includes sample configuration files for the Glacier2 router, IceGrid node, and IceGrid registry. These files provide a good starting point on which to base your own configurations, and they contain comments that describe the settings in detail.

The [Ice manual](#) provides more information on installing and running the IceGrid registry, IceGrid node, and Glacier2 router as Windows services.

[Back to Top ^](#)

Starting IceGrid GUI on Windows

You can launch IceGrid GUI using the shortcut that the Ice MSI installer created in your Start menu as **IceGrid GUI**. IceGrid GUI is a Java 8-based application.

[Back to Top ^](#)

Unattended Installation

The Ice MSI installer supports unattended installation. For example, in an administrative command window you can run:

```
start /wait Ice-3.7.0.msi /qn /l*v install.log
```

Windows may prompt you to confirm the installation, otherwise the installer runs using its default configuration (i.e., default installation folder, adds the installation's `bin` folder to the system PATH, and installs the PHP extension for the PHP installations it detects) but without any user interface. The installer will create a log of its activities in the file `install.log`.

[Back to Top ^](#)

Registry Key

The Ice MSI installer adds information to the Windows registry to indicate where it was installed. Developers can use this information to locate the Ice files in their applications.

The registration key used by this installer is:

```
HKEY_LOCAL_MACHINE\SOFTWARE\ZeroC\Ice 3.7.0
```

The install location is stored as a string value named `InstallDir`.

[Back to Top ^](#)

Using the Sample Programs on Windows

The Ice sample programs are provided in [ice-demos GitHub repository](#). You can browse this repository to see build and usage instructions for all supported programming languages.

Clone the ice-demos repository as follows:

```
git clone -b 3.7 https://github.com/zeroc-ice/ice-demos.git
cd ice-demos
```

[Back to Top ^](#)