Build W/O Installing (BWOI) Example

*This sample is compatible with the Microsoft Game Development Kit (June 2020)*

# Description

Individual developers are expected to have both the compiler toolset & required SDKs installed on their machines for day-to-day work. The *Microsoft Game Development Kit* (GDK) provides Visual Studio integration for debugging, MSBuild platforms definitions, and profiling tools in addition to headers & libraries. Maintaining a build server, however, is greatly simplified if you can use a ‘xcopy-style’ deployment for the headers & libraries when doing daily builds. This example demonstrates a method for building MSBuild-based projects using the **Gaming.\*.x64** platforms without having the Microsoft GDK installed. It also provides an option to use Windows containers to create an isolated build environment, with no need to install Visual Studio directly on the host machine.

# Software Setup

Build machines typically have the Visual Studio toolset and Windows SDKs installed on them as part of a regularly maintained image. This is true of Azure DevOps “[Microsoft-Hosted](https://docs.microsoft.com/en-us/azure/devops/pipelines/agents/hosted?view=azure-devops)” and is typical of setting up a [self-hosted Windows agent](https://docs.microsoft.com/en-us/azure/devops/pipelines/agents/v2-windows?view=azure-devops) or other custom build machine.

For building Microsoft GDK projects, you can set up [Visual Studio 2017](https://walbourn.github.io/vs-2017-15-9-update/) (which can only build v141 platform toolset VC++ projects), [Visual Studio 2019](https://walbourn.github.io/visual-studio-2019/) (which can build v141 and v142 platform toolset VC++ projects), or [Visual Studio 2022](https://walbourn.github.io/visual-studio-2022/) (which can build v141, v142, and v143 platform toolset VC++ projects). You can also use either a full Visual Studio install or the [Visual Studio Build Tools](https://visualstudio.microsoft.com/downloads/#build-tools-for-visual-studio-2022). Be sure to install the following components:

**Option 1: Full Visual Studio Install**

|  |  |
| --- | --- |
| Workload | Component ID (for [command line install](https://docs.microsoft.com/en-us/visualstudio/install/use-command-line-parameters-to-install-visual-studio)) |
| Game Development with C++ | Microsoft.VisualStudio.Workload.NativeGame |
| Desktop development with C++  *Required component, VS 2019/2022 only:* Windows 10 SDK (10.0.19041.0)  *Optional component, VS 2019/2022 only:* MSVC v141 - VS 2017 C++ x64/x86 build tools (v14.16)  *\* Only required if building v141 platform toolset projects using VS 2019/MSBuild 16.0 or VS 2022/MSBuild 17.0*  *Optional component, VS 2022 only:* MSVC v142 - VS 2019 C++ x64/x86 build tools (v14.29)  *\* Only required if building v142 platform toolset projects using VS 2022/MSBuild 17.0*  *Optional component, VS 2019/2022 only:* C++ Clang tools for Windows (12.0.0 - x64/x86)  *\* Only required if building using the Clang toolset* | Microsoft.VisualStudio.Workload.NativeDesktop  *VS 2019/2022 only:* Microsoft.VisualStudio.Component.Windows10SDK.19041  *Optional, VS 2019/2022 only:* Microsoft.VisualStudio.Component.VC.v141.x86.x64  *Optional, VS 2022 only:* Microsoft.VisualStudio.ComponentGroup.VC.Tools.142.x86.x64  *Optional, VS 2019/2022 only:* Microsoft.VisualStudio.ComponentGroup.NativeDesktop.Llvm.Clang |

**Option 2: Visual Studio Build Tools**

|  |  |
| --- | --- |
| Workload | Component ID (for [command line install](https://docs.microsoft.com/en-us/visualstudio/install/use-command-line-parameters-to-install-visual-studio)) |
| C++ Build Tools  *Required component, VS 2019/2022 only:* Windows 10 SDK (10.0.19041.0)  *Required component, VS 2019/2022 only:* MSVC v142 - VS 2019 C++ x64/x86 build tools (Latest) or MSVC v143 - VS 2022 C++ x64/x86 build tools (Latest)  *\* VS 2017 includes the equivalent component automatically*  *Optional component, VS 2019/2022 only:* MSVC v141 - VS 2017 C++ x64/x86 build tools (v14.16)  *\* Only required if building v141 platform toolset projects using VS 2019/MSBuild 16.0 or VS 2022/MSBuild 17.0*  *Optional component, VS 2022 only:* MSVC v142 - VS 2019 C++ x64/x86 build tools (v14.29)  *\* Only required if building v142 platform toolset projects using VS 2022/MSBuild 17.0*  *Optional component, VS 2019/2022 only:* C++ Clang tools for Windows (12.0.0 - x64/x86)  *\* Only required if building using the Clang toolset* | Microsoft.VisualStudio.Workload.VCTools  *VS 2019/2022 only:* Microsoft.VisualStudio.Component.Windows10SDK.19041  *VS 2019/2022 only:* Microsoft.VisualStudio.Component.VC.Tools.x86.x64  *Optional, VS 2019/2022 only:* Microsoft.VisualStudio.Component.VC.v141.x86.x64  *Optional, VS 2022 only:* Microsoft.VisualStudio.ComponentGroup.VC.Tools.142.x86.x64  *Optional, VS 2019/2022 only:* Microsoft.VisualStudio.ComponentGroup.NativeDesktop.Llvm.Clang |

|  |
| --- |
| The BWOIExample project uses the v142 toolset by default, which means it requires VS 2019 or VS 2022. Building with VS 2019 requires the MSVC v142 - VS 2019 C++ x64/x86 build tools (Latest) component, and VS 2022 requires the MSVC v142 - VS 2019 C++ x64/x86 build tools (v14.29) component. |

For VS 2017 (15.9 update), the Windows 10 SDK (17763) will be installed by default. To get the Windows 10 SDK (19041) you need to install it [stand-alone](https://developer.microsoft.com/en-US/windows/downloads/windows-10-sdk).

# Setting up the build environment

Once the software requirements have been installed, you can set up an extracted GDK that does not require installation. There are two ways to do this. It’s also possible to extract the Windows 10 SDK if desired.

## Method 1: Download the extracted GDK

This is recommended as the simplest option.

1. Go to [Xbox Developer Downloads](https://aka.ms/gdkdl).
2. Select “Game Core” as the file type.
3. In the build/version menu, select “Microsoft GDK Extracted for Build Systems” for the GDK build you wish to use.
4. Download the ZIP and extract it to a folder somewhere on your build machine. Choose a location with a short path to avoid MAX\_PATH issues.

## Method 2: Extract the GDK manually

This method is more complex, but does not require a separate download. You will need a copy of the standard GDK installer.

|  |
| --- |
| This method can be used with the public GDK, which does not have an extracted download option. |

1. **Open a Command Prompt** (this does not have to be a *Developer Command Prompt* for VS or the GDK).
2. **cd** to the BWOIExample sample folder.
3. Set up environment variables for VS 2022, 2019 or 2017 and provide your target edition number. If you specify a custom path for the extracted GDK, use a short, absolute, unquoted path to avoid issues such as exceeding MAX\_PATH.  
     
   setenv vs2022 220300 [path-for-extracted-sdks]
4. Extract the GDK from the installer image:

extractgdk <path-to-gdk-installer>\Installers

|  |
| --- |
| All use of MSIEXEC takes a global lock, so even for just extract operations it will fail if another MSIEXEC instance is running at the same time (Windows update or other instance of the same script).  For build pipelines run on the same VM, you need to provide some external lock/unlock cycle based on the use of the Global\\_MSIExecute mutex and your own global lock.  Generally, it’s easier to just extract the MSI once on a developer’s machine and copy the results to an agent-accessible folder. |

## Optional: Extract the Windows 10 SDK

If you wish, you can also extract the Windows 10 SDK, which will ensure the right version is always available on the build machine. This is generally unnecessary, as long as you install the Windows 10 SDK (19041) with your Visual Studio install.

This process requires a copy of the Windows 10 SDK installer image. The easiest way to get this is to download the Windows 10 SDK .ISO from [Windows Dev Center](https://developer.microsoft.com/windows/downloads/windows-10-sdk) (Version 10.0.19041.0 is required).

1. **Open a Command Prompt** and **cd** to the BWOIExample folder.
2. Set up the environment variables. Use the same path as your extracted GDK.  
     
   setenv vs2022 220300 [path-for-extracted-sdks]
3. Extract the Windows 10 SDK from the installer image:  
     
   extractsdk <path-to-sdk-installer>\Installers

## VS 2019/2022 only: Merge VCTargets

In addition to setting up the flat file directory of the GDK, VS 2019 and 2022 BWOI rely on having a combined VCTargets folder that merges the standard Microsoft.Cpp MSBuild rules with the GDK’s MSBuild rules. For VS 2017, this can be handled through an internal variable, but for VS 2019 and 2022 the robust solution is to create a merged folder alongside the extracted GDK.

1. **Open a Command Prompt** and **cd** to the BWOIExample folder.
2. Set up the environment variables. Provide the path for your downloaded or manually extracted GDK.  
     
   setenv vs2022 220300 [path-for-extracted-sdks]
3. Build the merged VC++ MSBuild targets directories and place them next to the extracted GDK:  
     
   vctargets

After running these steps, the ExtractedFolder environment variable points to the extracted, portable GDK (and optional Windows SDK and VCTargets directories) that the sample will build against. This folder can be copied to any other build machine as well.

# Building the sample

The rest of the building is done as normal. This BWOI example is driven by the Directory.Build.props file. The target vcxproj itself is completely “stock” and if you remove the Directory.Build.props file will work exactly as you’d expect on a normal developer system with the GDK installed.

1. **Open a Command Prompt** (this does not have to be a Developer Command Prompt for VS or the GDK).
2. **cd** to the BWOIExample sample folder.
3. Run **setenv** for VS 2022, 2019 or 2017 and your GDK edition target:

setenv vs2022 220300 [path-for-extracted-sdks]

|  |
| --- |
| If you don’t run setenv, the build will fall back to default values specified in Directory.Build.props. You can modify these directly in the file if you prefer. You will also need to make sure MSBuild is on the path if not using setenv. |

1. Build the project on the command-line:

msbuild BWOIExample.vcxproj /p:Configuration=Debug /p:Platform=Gaming.Desktop.x64

msbuild BWOIExample.vcxproj /p:Configuration=Debug /p:Platform=Gaming.Xbox.XboxOne.x64

msbuild BWOIExample.vcxproj /p:Configuration=Debug /p:Platform=Gaming.Xbox.Scarlett.x64

|  |
| --- |
| For VS 2019, if you only wish to support v142 platform toolset projects and did *not* install Microsoft.VisualStudio.Component.VC.v141.x86.x64, then you should edit the Directory.Build.Props to remove the setting of VCTargetsPath15. Similarly, for VS 2022, remove both VCTargetsPath15 and VCTargetsPath16 if you only installed support for the v143 platform toolset. |

# Building in a Windows container

As an alternative, Windows containers, run with Docker, can be used to create an isolated, reproducible build environment. These can be used on build servers, or even for local developer builds, to ensure a consistent build process. This sample includes a Dockerfile that sets up a BWOI build environment using the Visual Studio 2022 Build Tools.

|  |
| --- |
| The process described here requires that the project use the v142 toolset or later.  To learn more about Windows containers, see the [Containers on Windows documentation](https://docs.microsoft.com/en-us/virtualization/windowscontainers/). |

To use the Dockerfile, you still need to provide an extracted GDK, and optionally an extracted Windows SDK. However, you don’t need to install Visual Studio on the host machine.

1. Ensure Docker is installed and set to use Windows containers, as described [here](https://docs.microsoft.com/en-us/virtualization/windowscontainers/quick-start/set-up-environment).
2. Move the Dockerfile to a parent directory that contains both the BWOIExample project and the extracted SDKs, for example:

<parent dir>

-> Dockerfile

-> BWOIExample

-> <project and script files>

-> sdks

-> Microsoft GDK

-> <extracted GDK files>

-> <optional extracted Windows SDK>

|  |
| --- |
| Docker only needs access to setenv.cmd, vctargets.cmd, and the extracted SDKs when building the container. You can place the actual project source elsewhere if you wish. |

1. Navigate to the directory containing the Dockerfile and run:  
     
   docker build -t gdkbwoi:latest -m 2GB --build-arg ExtractedSDKDir="sdks" --build-arg ScriptDir="BWOIExample" --build-arg GDKVer="220300" .

|  |
| --- |
| To allow your container to use additional CPU cores, use the --cpus=*N* flag. To use additional memory, change the value in the -m 2GB flag. |

Docker automates the process of creating the container, downloading and installing the VS Build Tools, copying the necessary \*.cmd scripts and extracted SDKs, and merging the VCTargets.

1. Once the container is built, run it with:  
     
   Using cmd.exe:

docker run --rm -v %cd%\BWOIExample:c:\Project -w c:\Project gdkbwoi msbuild BWOIExample.vcxproj /p:Configuration=Debug /p:Platform=Gaming.Xbox.Scarlett.x64  
  
Using PowerShell:

docker run --rm -v ${pwd}\BWOIExample:c:\Project -w c:\Project gdkbwoi msbuild BWOIExample.vcxproj /p:Configuration=Debug /p:Platform=Gaming.Xbox.Scarlett.x64  
  
This command launches the container, mounts the project directory inside it, and runs msbuild with the specified parameters. You can change the configuration and platform as needed. To build a different project, change “%cd%\BWOIExample” to the project’s location.

The container will exit when the build completes. Because the project directory was mounted in the container, the build results will appear in the project directory on the host machine.

# Additional Information

The Microsoft GDK documentation covers the MSBuild “BWOI” properties in detail:

Microsoft Game Development Kit documentation

-> Development and tools

-> Using the Microsoft Game Development Kit (GDK) without installation

<https://aka.ms/GDK_BWOI>

The *CMakeExample* sample provides details on all the specific complier & linker switches if using a non-MSBuild based build system. It supports a build option (off by default) to enable using the same BWOI image created by this sample’s extractgdk.cmd script. The CMake example does not need the results of vctargets.cmd because it doesn’t use the Gaming.\*.x64 MSBuild platforms.

See the **CMakeExample** for more details.

# Known Issues

With some versions of VS 2019, if using DisableInstalledVCTargetsUse=true and the project contains <MinimumVisualStudioVersion>16.0</MinimumVisualStudioVersion>, then it can fail to build with:

C:\Program Files (x86)\Microsoft Visual Studio\2019\Enterprise\MSBuild\Current\Bin\Microsoft.Common.CurrentVersion.targets(816,5): error : The BaseOutputPath/OutputPath property is not set for project 'X.vcxproj'. Please check to make sure that you have specified a valid combination of Configuration and Platform for this project. Configuration='Debug' Platform='Gaming.XBox.Scarlett.x64'. You may be seeing this message because you are trying to build a project without a solution file, and have specified a non-default Configuration or Platform that doesn't exist for this project.

The workaround is to add an override to **Directory.Build.props**

<PropertyGroup>

<ExtractedFolder Condition="'$(ExtractedFolder)'==''">C:\xtracted\</ExtractedFolder>

<ExtractedFolder Condition="!HasTrailingSlash('$(ExtractedFolder)')">$(ExtractedFolder)\</ExtractedFolder>

<\_AlternativeVCTargetsPath160>$(ExtractedFolder)VCTargets160\</\_AlternativeVCTargetsPath160>

<\_AlternativeVCTargetsPath150>$(ExtractedFolder)VCTargets150\</\_AlternativeVCTargetsPath150>

<!-- Workaround for VS bug -->

<MinimumVisualStudioVersion>15.0</MinimumVisualStudioVersion>

</PropertyGroup>

This issue [was fixed](https://developercommunity.visualstudio.com/t/1695-or-later-fails-when-using-disableinstalledvct/1435971) in Visual Studio 2019 version 16.11.

# Version History

|  |  |
| --- | --- |
| **February 2020** | Initial version. |
| **May 2020** | Updated for the optional extracted Windows 10 SDK. |
| **June 2020** | Updated for 2006 GDK FAL release. |
| **April 2021** | Add LargeLogo.png. |
| **June 2021** | Removed information on deprecated GDKs, added additional clarifications, and added use of DisableInstalledVCTargetsUse.  General code cleanup. |
| **October 2021** | Added Dockerfile and instructions on building in a Windows container. |
| **March 2022** | Updated to support Visual Studio 2022.  Updated the project file to use the v142 toolset by default.  Changed the Dockerfile to use a mounted directory. |