内部版本 W/O 安装 (BWOI) 示例

*此示例兼容于 Microsoft 游戏开发工具包(2020 年 6 月)*

# 说明

单个开发人员应该同时在计算机中安装有编译器工具集和所需的 SDK，以完成日常工作。除标题和库以外，Microsoft *游戏开发工具包* (GDK) 还提供了用于调试的 Visual Studio 集成、MSBuild 平台定义和分析工具。但是，如果在进行日常生成工作时对标题和库可以使用“xcopy-style”部署，则将大大简化维护内部版本服务器的工作。此示例演示了一种在未安装 Microsoft GDK 的情况下，使用 **Gaming.\*.x64** 平台生成基于 MSBuild 项目的方法。它还提供了使用 Windows 容器创建独立生成环境的选项，无需直接在主机上安装 Visual Studio。

# 软件设置

生成计算机通常安装有 Visual Studio 工具集和 Windows SDK，将其作为定期维护映像的一部分。Azure DevOps“[Microsoft 托管](https://docs.microsoft.com/en-us/azure/devops/pipelines/agents/hosted?view=azure-devops)”就是如此，也是设置 [自托管 Windows 代理](https://docs.microsoft.com/en-us/azure/devops/pipelines/agents/v2-windows?view=azure-devops) 或其他自定义生成计算机 的特点。

对于生成 Microsoft GDK 项目，可以设置 [Visual Studio 2017](https://walbourn.github.io/vs-2017-15-9-update/)（只能生成 v141 平台工具集 VC++ 项目）或 [Visual Studio 2019](https://walbourn.github.io/visual-studio-2019/)（可以生成 v141 和 v142 平台工具集 VC++ 项目）。还可使用完整的 Visual Studio 安装或 [Visual Studio 生成工具](https://visualstudio.microsoft.com/downloads/#build-tools-for-visual-studio-2019)。请确保安装以下组件：

**选项 1：完整 Visual Studio 安装**

|  |  |
| --- | --- |
| 工作负载 | 组件 ID（用于 [命令行安装](https://docs.microsoft.com/en-us/visualstudio/install/use-command-line-parameters-to-install-visual-studio)） |
| 使用 C++ 进行游戏开发 | Microsoft.VisualStudio.Workload.NativeGame |
| 使用 C ++ 进行桌面开发  *必需组件，仅 VS 2019：* Windows 10 SDK (10.0.19041.0)  *可选组件，仅限 VS 2019：*适用于 x64/x86 的 MSVC v141 C++ 生成工具  *\*仅在使用 VS 2019/MSBuild 16.0 生成 v141 平台工具集项目时必需*  *可选组件，仅限 VS 2019：* 适用于 Windows (12.0.0 - x64/x86) 的 C++ Clang 工具  *\* 仅当在生成使用 Clang 工具集时才需要* | Microsoft.VisualStudio.Workload.NativeDesktop  *仅 VS 2019：*Microsoft.VisualStudio.Component.Windows10SDK.19041  *可选，仅 VS 2019：*Microsoft.VisualStudio.Component.VC.v141.x86.x64  *可选，仅 VS 2019：*Microsoft.VisualStudio.ComponentGroup.NativeDesktop.Llvm.Clang |

**选项 2：Visual Studio 生成工具**

|  |  |
| --- | --- |
| 工作负载 | 组件 ID（用于 [命令行安装](https://docs.microsoft.com/en-us/visualstudio/install/use-command-line-parameters-to-install-visual-studio)） |
| C++ 生成工具  *必需组件，仅 VS 2019：* Windows 10 SDK (10.0.19041.0)  *必需组件，仅 VS 2019：*MSVC v142 - VS 2019 C++ x64/x86 生成工具（最新）  *\* VS 2017 自动包含等效组件*  *可选组件，仅限 VS 2019：*适用于 x64/x86 的 MSVC v141 C++ 生成工具  *\*仅在使用 VS 2019/MSBuild 16.0 生成 v141 平台工具集项目时必需*  *可选组件，仅限 VS 2019：* 适用于 Windows (12.0.0 - x64/x86) 的 C++ Clang 工具  *\* 仅当在生成使用 Clang 工具集时才需要* | Microsoft.VisualStudio.Workload.VCTools  *仅 VS 2019：*Microsoft.VisualStudio.Component.Windows10SDK.19041  *仅 VS 2019：*Microsoft.VisualStudio.Component.VC.Tools.x86.x64  *可选，仅 VS 2019：*Microsoft.VisualStudio.Component.VC.v141.x86.x64  *可选，仅 VS 2019：*Microsoft.VisualStudio.ComponentGroup.NativeDesktop.Llvm.Clang |

|  |
| --- |
| 默认情况下，BWOIExample 项目使用 v141 工具集。如果在 VS 2019 上生成时没有修改，则需要安装适用于 x64/x86 的 MSVC v141 C++ 生成工具。 |

对于 VS 2017（15.9 更新），将默认安装 Windows 10 SDK（17763）。要获取 Windows 10 SDK（19041），需要将其 [独立安装](https://developer.microsoft.com/en-US/windows/downloads/windows-10-sdk)。

# 设置生成环境

安装软件要求后，可以设置无需安装的提取 GDK。可通过两种方法执行此操作。如果需要，也可以提取 Windows 10 SDK。

## 方法 1：下载提取的 GDK

这是建议的最简单的选项。

1. 请转到 [Xbox 开发人员下载](https://aka.ms/gdkdl)。
2. 选择“Game Core”作为文件类型。
3. 在“内部版本/版本”菜单中，为希望使用的 GDK 版本选择“为生成系统提取的 Microsoft GDK”。
4. 下载 ZIP，然后将其提取到生成计算机中某个位置的文件夹中。选择带有短路径的位置，以避免 MAX\_PATH 问题。

## 方法 2：手动提取 GDK

此方法较复杂，但不需要单独下载。你将需要标准 GDK 安装程序的副本。

|  |
| --- |
| 此方法没有提取的下载选项，可与公共 GDK 一起使用。 |

1. **打开命令提示符**（这不一定是 VS 或 GDK 的*开发人员命令提示符*）。
2. **cd** 到 BWOIExample 示例文件夹。
3. 为 VS 2019 或 VS 2017 设置环境变量并提供目标版本编号。如果为提取的 GDK 指定自定义路径，请使用简短、绝对、且不带引号的路径，以避免如超出 MAX\_PATH 等问题。  
     
   setenv vs2019 210600 [path-for-extracted-sdks]
4. 从安装程序映像中提取 GDK：

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

|  |
| --- |
| 所有 MSIEXEC 的使用均采用全局锁定，因此即使只是进行提取操作，如果同时运行另一个 MSIEXEC 实例（Windows 更新或同一脚本的其他实例），也将会失败。  对于在同一虚拟机上运行的生成管道，需要根据 Global\\_MSIExecute 互斥的使用和你自己的全局锁定提供一些外部锁定/解除锁定循环。  通常，只在开发人员计算机上提取 MSI 一次，并将结果复制到代理可访问的文件夹会比较容易。 |

## 可选方案：提取 Windows 10 SDK

如果愿意，也可提取 Windows 10 SDK，这将确保在生成计算机上始终提供正确的版本。只要在 Visual Studio 安装时安装了 Windows 10 SDK (19041)，通常没有必要采用这一方案。

此过程需要 Windows 10 SDK 安装程序映像的副本。获取此副本的最简单方法是从 [Windows 开发人员中心](https://developer.microsoft.com/windows/downloads/windows-10-sdk)（需要版本 10.0.19041.0）下载 Windows 10 SDK .ISO。

1. **打开命令提示符**并 **cd** 到 BWOIExample 文件夹。
2. 设置环境变量。使用与提取的 GDK 相同的路径。  
     
   setenv vs2019 210600 [path-for-extracted-sdks]
3. 从安装程序映像中提取 Windows 10 SDK：  
     
   extractsdk <path-to-sdk-installer>\Installers

## 仅 VS 2019：合并 VCTargets

除设置 GDK 的平面文件目录外，VS 2019 BWOI 还依赖于拥有一个组合的 VCTargets 文件夹，该文件夹合并了标准的 Microsoft.Cpp MSBuild 规则与 GDK 的 MSBuild 规则。对于 VS 2017，这一点可通过内部变量处理，但对于 VS 2019，强大的解决方案是在提取的 GDK 旁边创建一个合并文件夹。

1. **打开命令提示符**并 **cd** 到 BWOIExample 文件夹。
2. 设置环境变量。提供已下载或手动提取 GDK 的路径。  
     
   setenv vs2019 210600 [path-for-extracted-sdks]
3. 生成合并的 VC++ MSBuild 目标目录，并将其放在提取的 GDK 旁边：  
     
   vctargets

运行这些步骤之后，ExtractedFolder 环境变量指向提取的可移植 GDK（以及可选的 Windows SDK 和 VCTargets 目录），该样本将基于此 GDK 生成。也可将此文件夹复制到任何其他生成计算机。

# 生成示例

生成的其余部分照常进行。此 BWOI 示例由 Directory.build.props 驱动。目标 vcxproj 本身是完全“库存”的，如果删除 Directory.Build.props 文件，目标将在安装了该 GDK 的普通开发人员系统上，完全如预期一样工作。

1. **打开命令提示符**（这不一定是 VS 或 GDK 的开发人员命令提示符）。
2. **cd** 到 BWOIExample 示例文件夹。
3. 为 VS 2019 或 VS 2017 和 GDK 版本目标运行 **setenv**:

setenv vs2019 200600 [path-for-extracted-sdks]

|  |
| --- |
| 如果不运行 setenv，则生成将回退到 Directory.Build.props 中指定的默认值。如果愿意，可直接在文件中进行修改。如果不使用 setenv，还将需要确保 MSBuild 在路径上。 |

1. 在命令行上生成项目：

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

|  |
| --- |
| 对于 VS 2019，如果仅希望支持 v142 平台工具集项目，但并未安装 Microsoft.VisualStudio.Component.VC.v141.x86.x64，则应编辑 Directory.Build.Props，以删除 VCTargetsPath15 的设置*。* |

# 在 Windows 容器中生成

作为替代方法，使用 Docker 运行的 Windows 容器可用于创建独立、可重现的生成环境。这些可在生成服务器上使用，甚至可用于本地开发人员生成，以确保生成过程保持一致。此示例包括使用 Visual Studio 2019 生成工具设置 BWOI 生成环境的 Dockerfile。

|  |
| --- |
| 若要了解有关 Windows 容器的详细信息，请参阅 [Windows 文档上的容器](https://docs.microsoft.com/en-us/virtualization/windowscontainers/)。 |

若要使用 Dockerfile，仍需提供提取的 GDK 和提取的 Windows SDK（可选）。然而，无需在主机上安装 Visual Studio。

1. 如[此处](https://docs.microsoft.com/en-us/virtualization/windowscontainers/quick-start/set-up-environment)所述，确保安装 Docker 并将其设置为使用 Windows 容器。
2. 将 Dockerfile 移动到同时包含 BWOIExample 项目和提取的 SDK 的父目录，例如：

<parent dir>

-> Dockerfile

-> BWOIExample

-> <project files>

-> sdks

-> Microsoft GDK

-> <extracted GDK files>

-> <optional extracted Windows SDK>

1. 导航到包含 Dockerfile 的目录并运行：  
     
   docker build -t gdkbwoi:latest -m 2GB --build-arg ExtractedSDKDir="sdks" --build-arg ProjectDir="BWOIExample" --build-arg GDKVer="211000"。

|  |
| --- |
| 若要允许容器使用其他 CPU 核心，请使用 --cpus=*N* 标志。若要使用其他内存，请更改 -m 2GB 标志中的值。 |

Docker 自动执行创建容器、下载和安装 VS Build Tools、复制项目和提取的 SDK 以及合并 VCTargets 的流程。

1. 生成容器后，使用以下命令运行它：  
     
   docker run -it gdkbwoi  
     
   这此命令会在容器的上下文中打开一个命令提示符，其中的环境已配置为生成。
2. 使用命令行在容器中生成项目：  
     
   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

|  |
| --- |
| Dockerfile 中的注释还演示了如何在容器运行时自动调用生成。 |

可以使用 [docker cp](https://docs.docker.com/engine/reference/commandline/cp/) 命令从容器中复制文件。在实际的生成环境中，还可以扩展容器的行为，以自动将内部版本上传到服务器，甚至托管文件本身。

# 其他信息

Microsoft GDK 文档详细介绍了 MSBuild“BWOI”属性：

Microsoft 游戏开发工具包

-> Microsoft Game Core 游戏开发：环境和工具

无需安装即可使用 Microsoft 游戏开发工具包。

<https://aka.ms/GDK_BWOI>

*CMakeExample* 示例提供了有关使用基于 MSBuild 的生成系统时，所有特定编译器和链接器开关的详细信息。它支持生成选项（默认为关闭）以启用使用由此示例的 extractgdk.cmd 脚本创建的同一 BWOI 映像。CMake 示例不需要 vctargets.cmd 的结果，因为它不使用 Gaming.\*.x64 MSBuild 平台。

有关详细信息，请参阅 **CMakeExample**。

# 已知问题

对于 VS 2019，如果使用 DisableInstalledVCTargetsUse=true，且项目包含 <MinimumVisualStudioVersion>16.0</MinimumVisualStudioVersion>，则在以下情况下可能无法生成：

C:\Program Files (x86)\Microsoft Visual Studio\2019\Enterprise\MSBuild\Current\Bin\Microsoft.Common.CurrentVersion.targets(816,5): 错误 :没有为项目“X.vcxproj”设置 BaseOutputPath/OutputPath 属性。 请检查以确保为此项目指定了有效的配置和平台组合。 Configuration='Debug' Platform='Gaming.XBox.Scarlett.x64'。 可能会看到此消息，因为你尝试生成没有解决方案的项目，并且指定了一个对此项目来说并不存在的非默认配置或平台。

解决方法是向 **Directory.Build.props** 添加替代

<PropertyGroup>

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

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

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

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

<!- VS bug 的解决方法 ->

<MinimumVisualStudioVersion>15.0</MinimumVisualStudioVersion>

</PropertyGroup>

有关此解决方法的当前状态，请参阅 [VS 开发人员反馈问题](https://developercommunity.visualstudio.com/t/1695-or-later-fails-when-using-disableinstalledvct/1435971)。

# 版本历史记录

|  |  |
| --- | --- |
| **2020 年 2 月** | 初始版本。 |
| **2020 年 5 月** | 针对可选的提取 Windows 10 SDK 的更新。 |
| **2020 年 6 月** | 针对 2006 GDK FAL 版本的更新。 |
| **2021 年 4 月** | Add LargeLogo.png。 |
| **2021 年 6 月** | 删除了弃用的 GDK 的信息，添加了额外的说明，并增加了DisableInstalledVCTargetsUse 的使用。  常规代码清理。 |
| **2021 年 10 月** | 添加了 Dockerfile 和有关在 Windows 容器中生成的说明。 |