インストールしないビルド (BWOI) の例

*このサンプルは Microsoft Game Development Kit (2020 年 6 月) と互換性があります。*

# 説明

個々の開発者は、日常的な作業のために、コンパイラ ツールセットと必要な SDK の両方をマシンにインストールする必要があります。*Microsoft Game Development Kit* (GDK) では、ヘッダーとライブラリに加えて、デバッグ、MSBuild プラットフォーム定義、プロファイリング ツールのためのVisual Studio 統合を提供しています。ただし、毎日のビルドを行うときにヘッダーとライブラリに対して ‘xcopy-style’ 配置を使用できる場合は、ビルド サーバーの保守が大幅に簡素化されます。この例では、Microsoft GDK をインストールせずに、**Gaming.\*.x64** プラットフォームを使用して MSBuild ベースのプロジェクトをビルドする方法を示します。Windows コンテナーを使用して、Visual Studio をホスト コンピューターに直接インストールすることなく、分離されたビルド環境を作成するオプションも提供しています。

# ソフトウェアのセットアップ

通常、ビルド マシンには、定期的に保守されているイメージの一部として、Visual Studio ツールセットと Windows SDK がインストールされます。これは、Azure DevOps "[Microsoft ホステッド](https://docs.microsoft.com/en-us/azure/devops/pipelines/agents/hosted?view=azure-devops)" に当てはまります。これは、[セルフホステッド Windows エージェント](https://docs.microsoft.com/en-us/azure/devops/pipelines/agents/v2-windows?view=azure-devops) またはその他のカスタム ビルド マシンを設定する場合に一般的です。

Microsoft GDK プロジェクトをビルドするには、[Visual Studio 2017](https://walbourn.github.io/vs-2017-15-9-update/) (v141 プラットフォーム ツールセットの VC++ プロジェクトのみをビルドできる)、[Visual Studio 2019](https://walbourn.github.io/visual-studio-2019/) (v141 および v142 プラットフォーム ツールセットの VC++ プロジェクトをビルドできる)、または [Visual Studio 2022](https://walbourn.github.io/visual-studio-2022/) (v141、v142、および v143 プラットフォーム ツールセットの VC++ プロジェクトをビルドできる) をセットアップできます。完全な Visual Studio インストールまたは [Visual Studio Build Tools](https://visualstudio.microsoft.com/downloads/#build-tools-for-visual-studio-2022) を使用することもできます。次のコンポーネントをインストールしてください。

**オプション 1:完全な Visual Studio インストール**

|  |  |
| --- | --- |
| ワークロード | コンポーネント ID ([コマンド ライン インストール](https://docs.microsoft.com/en-us/visualstudio/install/use-command-line-parameters-to-install-visual-studio)の場合) |
| C++ でゲーム開発 | Microsoft.VisualStudio.Workload.NativeGame |
| C++ でデスクトップ開発  *必須コンポーネント、VS 2019 または 2022 のみ:* Windows 10 SDK (10.0.19041.0)  *省略可能なコンポーネント、VS 2019 または 2022 のみ:*MSVC v141 - VS 2017 C++ x64/x86 ビルド ツール (v14.16)  *\* VS 2019/MSBuild 16.0 または VS 2022/MSBuild 17.0 を使用して v141 プラットフォーム ツールセット プロジェクトをビルドする場合にのみ必要*  *省略可能なコンポーネント、VS 2022 のみ:*MSVC v142 - VS 2019 C++ x64/x86 ビルド ツール (v14.29)  *\* VS 2022/MSBuild 17.0 を使用して v142 プラットフォーム ツールセット プロジェクトをビルドする場合にのみ必要*  *省略可能なコンポーネント、VS 2019 または 2022 のみ:* Windows (12.0.0 - x64/x86) 用 C++ Clang ツール  *\* Clang ツールセットを使用してビルドする場合にのみ必要* | Microsoft.VisualStudio.Workload.NativeDesktop  *VS 2019 または 2022 のみ:*Microsoft.VisualStudio.Component.Windows10SDK.19041  *省略可能、VS 2019 または 2022 のみ:*Microsoft.VisualStudio.Component.VC.v141.x86.x64  *省略可能、VS 2022 のみ:*Microsoft.VisualStudio.ComponentGroup.VC.Tools.142.x86.x64  *省略可能、VS 2019 または 2022 のみ:*Microsoft.VisualStudio.ComponentGroup.NativeDesktop.Llvm.Clang |

**オプション 2:Visual Studio Build Tools**

|  |  |
| --- | --- |
| ワークロード | コンポーネント ID ([コマンド ライン インストール](https://docs.microsoft.com/en-us/visualstudio/install/use-command-line-parameters-to-install-visual-studio)の場合) |
| C++ Build Tools  *必須コンポーネント、VS 2019 または 2022 のみ:* Windows 10 SDK (10.0.19041.0)  *必須コンポーネント、VS 2019 または 2022 のみ:*MSVC v142 - VS 2019 C++ x64/x86 ビルド ツール (最新) または MSVC v143 - VS 2022 C++ x64/x86 ビルド ツール (最新)  *\* VS 2017 には同等のコンポーネントが自動で含まれます*  *省略可能なコンポーネント、VS 2019 または 2022 のみ:*MSVC v141 - VS 2017 C++ x64/x86 ビルド ツール (v14.16)  *\* VS 2019/MSBuild 16.0 または VS 2022/MSBuild 17.0 を使用して v141 プラットフォーム ツールセット プロジェクトをビルドする場合にのみ必要*  *省略可能なコンポーネント、VS 2022 のみ:*MSVC v142 - VS 2019 C++ x64/x86 ビルド ツール (v14.29)  *\* VS 2022/MSBuild 17.0 を使用して v142 プラットフォーム ツールセット プロジェクトをビルドする場合にのみ必要*  *省略可能なコンポーネント、VS 2019 または 2022 のみ:* Windows (12.0.0 - x64/x86) 用 C++ Clang ツール  *\* Clang ツールセットを使用してビルドする場合にのみ必要* | Microsoft.VisualStudio.Workload.VCTools  *VS 2019 または 2022 のみ:*Microsoft.VisualStudio.Component.Windows10SDK.19041  *VS 2019 または 2022 のみ:*Microsoft.VisualStudio.Component.VC.Tools.x86.x64  *省略可能、VS 2019 または 2022 のみ:*Microsoft.VisualStudio.Component.VC.v141.x86.x64  *省略可能、VS 2022 のみ:*Microsoft.VisualStudio.ComponentGroup.VC.Tools.142.x86.x64  *省略可能、VS 2019 または 2022 のみ:*Microsoft.VisualStudio.ComponentGroup.NativeDesktop.Llvm.Clang |

|  |
| --- |
| BWOIExample プロジェクトでは、既定で v142 ツールセットが使用されます。つまり、VS 2019 または VS 2022 が必要です。VS 2019 でビルドするには、MSVC v142 - VS 2019 C++ x64/x86 ビルド ツール (最新) コンポーネントが必要です。VS 2022 では、 MSVC v142 - VS 2019 C++ x64/x86 ビルド ツール (v14.29) コンポーネントが必要です。 |

VS 2017 (15.9 更新プログラム) では、Windows 10 SDK (17763) が既定でインストールされます。Windows 10 SDK (19041) を取得するには、[スタンドアロン](https://developer.microsoft.com/en-US/windows/downloads/windows-10-sdk)でインストールする必要があります。

# ビルド環境の設定

ソフトウェア要件をインストールしたら、インストールを必要としない抽出された GDK を設定できます。これを行うには 2 つの方法があります。必要に応じて、Windows 10 SDK を抽出することもできます。

## 方法 1:抽出された GDK をダウンロードする

これは、最も簡単なオプションとして推奨されます。

1. [Xbox デベロッパー向けダウンロード](https://aka.ms/gdkdl)のページに移動します。
2. ファイルの種類として [Game Core] を選択します。
3. [ビルド/バージョン] メニューで、使用する GDK ビルドの [Microsoft GDK Extracted for Build Systems] を選択します。
4. ZIP をダウンロードし、ビルド マシン上のどこかのフォルダーに抽出します。短いパスの場所を選択して、MAX\_PATH の問題を回避します。

## 方法 2:GDK を手動で抽出する

この方法はより複雑ですが、個別のダウンロードは必要ありません。標準の GDK インストーラーのコピーが必要です。

|  |
| --- |
| このメソッドは、公開 GDK で使用できます。これには、抽出されたダウンロードのオプションがありません。 |

1. **コマンド プロンプトを開きます** (VS または GDK 用の*開発者コマンド プロンプト*である必要はありません)。
2. BWOIExample サンプル フォルダーに **cd を実行します**。
3. VS 2022、2019、または 2017 の環境変数を設定し、ターゲット エディション番号を指定します。抽出された GDK のカスタム パスを指定する場合は、短い、引用符で囲まれていない絶対パスを使用して、MAX\_PATH を超えるなどの問題を回避します。  
     
   setenv vs2022 220300 [path-for-extracted-sdks]
4. インストーラー イメージから GDK を抽出します。

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

|  |
| --- |
| MSIEXEC のすべての使用ではグローバル ロックを受け取ります。そのため、抽出操作の場合でも、別の MSIEXEC インスタンス (Windows Update または同じスクリプトの他のインスタンス) が同時に実行されている場合、失敗します。  同じ VM でビルド パイプラインを実行する場合は、Global\\_MSIExecute ミューテックスと独自のグローバル ロックの使用に基づいて、外部のロックまたはロック解除サイクルを指定する必要があります。  一般に、開発者のマシンで MSI を 1 回抽出し、結果をエージェントがアクセスできるフォルダーにコピーする方が簡単です。 |

## 省略可能:Windows 10 SDK を抽出する

必要に応じて、Windows 10 SDK を抽出することもできます。これにより、適切なバージョンがビルド マシンで常に使用できるようになります。通常、Visual Studio インストールで Windows 10 SDK (19041) をインストールする限り、これは不要です。

このプロセスには、Windows 10 SDK インストーラー イメージのコピーが必要です。これを取得する最も簡単な方法は、Windows 10 SDK .ISO を [Windows デベロッパー センター](https://developer.microsoft.com/windows/downloads/windows-10-sdk) からダウンロードすることです (バージョン 10.0.19041.0 が必要です)。

1. **コマンド プロンプトを開き、**BWOIExample フォルダーに対して **cd を実行します**。
2. 環境変数を設定します。抽出した GDK と同じパスを使用します。  
     
   setenv vs2022 220300 [path-for-extracted-sdks]
3. インストーラー イメージから Windows 10 SDK を抽出します。  
     
   extractsdk <path-to-sdk-installer>\Installers

## VS 2019 または 2022 のみ:VCTargets のマージ

GDK のフラット ファイル ディレクトリの設定に加えて、VS 2019 および 2022 BWOI は、標準の Microsoft.Cpp MSBuild ルールと GDK の MSBuild ルールをマージする結合された VCTargets フォルダーがあることに依存しています。VS 2017 の場合、これは内部変数を使用して処理できますが、VS 2019 および 2022 の場合、堅牢なソリューションは、抽出された GDK と共に、マージされたフォルダーを作成することです。

1. **コマンド プロンプトを開き、**BWOIExample フォルダーに対して **cd を実行します**。
2. 環境変数を設定します。ダウンロードされた、または手動で抽出した GDK のパスを指定します。  
     
   setenv vs2022 220300 [path-for-extracted-sdks]
3. マージされた VC++ MSBuild ターゲット ディレクトリをビルドし、抽出された GDK の横に配置します。  
     
   vctargets

これらの手順を実行した後、ExtractedFolder 環境変数は、サンプルのビルド対象となる、抽出された移植可能な GDK (および省略可能な Windows SDK と VCTargets ディレクトリ) を指します。このフォルダーは、他のビルド マシンにもコピーできます。

# サンプルのビルド

ビルドの残りは通常どおりに実行されます。この BWOI の例は、Directory.Build.props ファイルによって実行されます。ターゲット vcxproj 自体は完全に "ストック" であり、Directory.Build.props ファイルを削除すると、GDK がインストールされている通常の開発者システムとまったく同じように動作します。

1. **コマンド プロンプトを開きます** (VS または GDK 用の開発者コマンド プロンプトである必要はありません)。
2. BWOIExample サンプル フォルダーに **cd を実行します**。
3. VS 2022、2019 または VS 2017、および GDK エディション ターゲットに対して、**setenv** を実行します。

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

|  |
| --- |
| setenv を実行しない場合、ビルドは Directory.Build.props で指定された既定値にフォールバックします。必要に応じて、ファイル内でこれらを直接変更できます。setenv を使用していない場合は、MSBuild がパス上にあることを確認する必要もあります。 |

1. 次のコマンド ラインでプロジェクトをビルドします。

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

|  |
| --- |
| VS 2019 で、v142 プラットフォーム ツールセット プロジェクトのみをサポートし、Microsoft.VisualStudio.Component.VC.v141.x86.x64 をインストール*していない*場合は、Directory.Build.Props を編集して VCTargetsPath15 の設定を削除する必要があります。同様に、VS 2022 では、v143 プラットフォーム ツールセットのサポートのみをインストールした場合は、VCTargetsPath15 と VCTargetsPath16 の両方を削除します。 |

# Windows コンテナー内でビルドする

代わりに、Docker で実行する Windows コンテナーを使用して、分離された、再現可能なビルド環境を作成できます。これらをビルド サーバーやローカルの開発者ビルドにも使用することで、ビルド プロセスを一貫して行うことができます。このサンプルには、Visual Studio 2022 ビルド ツールを使用して BWOI ビルド環境を設定する Dockerfile が含まれます。

|  |
| --- |
| ここで説明するプロセスでは、プロジェクトで v142 ツールセット以降を使用する必要があります。  Windows コンテナーの詳細については、[Windows コンテナーのドキュメント](https://docs.microsoft.com/en-us/virtualization/windowscontainers/)を参照してください。 |

Dockerfile を使用するには、抽出された GDK を提供する必要があります。必要に応じて、抽出された Windows SDK を提供することもできます。ただし、Visual Studio をホスト コンピューターにインストールする必要はありません。

1. Docker がインストールされており、Windows コンテナーを使用するように設定されていることを確認します。詳細については、[こちら](https://docs.microsoft.com/en-us/virtualization/windowscontainers/quick-start/set-up-environment)を参照してください。
2. Dockerfile を BWOIExample プロジェクトと抽出された SDK の両方を含む親ディレクトリに移動します。次に例を示します。

<親ディレクトリ>

-> Dockerfile

-> BWOIExample

-> <プロジェクトおよびスクリプト ファイル>

-> SDK

-> Microsoft GDK

-> <抽出された GDK ファイル>

-> <省略可能な抽出された Windows SDK>

|  |
| --- |
| コンテナーのビルド時に、Docker で必要となるのは、setenv.cmd、vctargets.cmd、および抽出された SDK へのアクセスだけです。必要に応じて、実際のプロジェクト ソースを別の場所に配置できます。 |

1. Dockerfile を含むディレクトリに移動し、次を実行します。  
     
   docker build -t gdkbwoi:latest -m 2GB --build-arg ExtractedSDKDir="sdks" --build-arg ScriptDir="BWOIExample" --build-arg GDKVer="220300" .

|  |
| --- |
| コンテナーが追加の CPU コアを使用することを許可するには、--cpus=*N* フラグを使用します。追加メモリを使用するには、-m 2GB フラグの値を変更します。 |

Docker で、コンテナーの作成、VS ビルド ツールのダウンロードとインストール、必要な \*.cmd スクリプトと抽出された SDK のコピー、VCTargets の結合のプロセスが自動化されます。

1. コンテナーがビルドされたら、次を使用して実行します。  
     
   cmd.exe の使用:

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

docker run --rm -v ${pwd}\BWOIExample:c:\Project -w c:\Project gdkbwoi msbuild BWOIExample.vcxproj /p:Configuration=Debug /p:Platform=Gaming.Xbox.Scarlett.x64  
  
このコマンドは、コンテナーを起動し、その中にプロジェクト ディレクトリをマウントし、指定されたパラメーターで msbuild を実行します。必要に応じて、構成とプラットフォームを変更できます。別のプロジェクトをビルドするには、“%cd%\BWOIExample” をそのプロジェクトの場所に変更します。

ビルドが完了すると、コンテナーは終了します。プロジェクト ディレクトリがコンテナーにマウントされたため、ビルド結果がホスト コンピューターのプロジェクト ディレクトリに表示されます。

# 追加情報

Microsoft GDK のドキュメントでは、MSBuild の "BWOI" プロパティについて詳しく説明しています。

Microsoft Game Development Kit のドキュメント

-> 開発とツール

-> インストールせずに Microsoft Game Development Kit (GDK) を使用

<https://aka.ms/GDK_BWOI>

*CMakeExample* サンプルでは、MSBuild ベース以外のビルド システムを使用している場合に、特定のすべてのコンパイラおよびリンカー スイッチに関する詳細を提供しています。このサンプルの extractgdk.cmd スクリプトで作成されたのと同じ BWOI イメージを使用できるようにするためのビルド オプション (既定ではオフ) がサポートされています。CMake の例では、Gaming.\*.x64 MSBuild プラットフォームを使用しないため、vctargets.cmd の結果は必要ありません。

詳細については、**CMakeExample** を参照してください。

# 既知の問題

VS 2019 のいくつかのバージョンで、DisableInstalledVCTargetsUse=true を使用していて、プロジェクトに <MinimumVisualStudioVersion>16.0</MinimumVisualStudioVersion> が含まれている場合、ビルドが次で失敗する可能性があります。

C:\Program Files (x86)\Microsoft Visual Studio\2019\Enterprise\MSBuild\Current\Bin\Microsoft.Common.CurrentVersion.targets(816,5): エラー :プロジェクト 'X.vcxproj' に BaseOutputPath/OutputPath プロパティが設定されていません。 このプロジェクトに、構成とプラットフォームの有効な組み合わせが指定されていることを確認してください。 Configuration='Debug' Platform='Gaming.XBox.Scarlett.x64'. ソリューション ファイルを使用せずにプロジェクトをビルドしようとしていますが、このプロジェクトに存在しない既定以外の構成またはプラットフォームが指定されているため、このメッセージが表示されている可能性があります。

回避策は、**Directory.Build.props** にオーバーライドを追加することです

<PropertyGroup>

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

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

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

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

<!-- VS バグの回避策 -->

<MinimumVisualStudioVersion>15.0</MinimumVisualStudioVersion>

</PropertyGroup>

この問題は、Visual Studio 2019 バージョン 16.11 で[修正されました](https://developercommunity.visualstudio.com/t/1695-or-later-fails-when-using-disableinstalledvct/1435971)。

# バージョン履歴

|  |  |
| --- | --- |
| **2020 年 2 月** | 初期バージョン。 |
| **2020 年 5 月** | 省略可能な抽出された Windows 10 SDK 用に更新されました。 |
| **2020 年 6 月** | 2006 GDK FAL リリース用に更新されました。 |
| **2021 年 4 月** | LargeLogo.png を追加しました。 |
| **2021 年 6 月** | 非推奨の GDK に関する情報を削除し、説明を追加し、DisableInstalledVCTargetsUse の使用を追加しました。  一般的なコードのクリーンアップ。 |
| **2021 年 10 月** | Dockerfile と Windows コンテナー内でのビルドに関する指示を追加しました。 |
| **2022 年 3 月** | Visual Studio 2022 をサポートするように更新しました。  既定で v142 ツールセットを使用するようにプロジェクト ファイルを更新しました。  マウントされたディレクトリを使用するように Dockerfile を変更しました。 |