# Dependent Files

## Build\AssemblyInfoCommon.cs

This file is automatically included in all C# projects. It adds assembly attributes that are normally included in a project’s AssemblyInfo.cs file.

## Build\AssemblyVersion.cs

This file is updated by our BuildRelease scripts and automatically included in all C# projects. It adds the AssemblyVersion and AssemblyFileVersion attributes to all C# binaries, as well as a constant AssemblyVersionInfo.Version string that is accessible to code within the project.

Until our version numbers exceed 1.8.40818, there is special handling to ensure that all VS 2010 builds use this assembly version. PTVS 1.0 incorrectly shipped with this version number. Builds for VS 2012 are unaffected by this.

## Build\CodeCoverage.proj

This file provides settings for building PTVS with code coverage analysis enabled.

At present, this file is used but has no effect.

## Build\Common.Build.settings

This file is included early in all projects. It ensures that cross-language variables are correct.

It defines the output directory (OutputPath and OutDir), defaulting to $(BuildRoot)\Binaries\$(Configuration)\. This value may be overridden by providing OutDir at the command line. OutputPath may also be provided, but must be guaranteed to include a terminating backslash.

RunFXCop [false], RunStyleCop [false] and TreatWarningsAsErrors [true] are Boolean variables that may be provided on the command line.

This script also imports one of Common.Build.Cpp.settings or Common.Build.CSharp.settings based on the extension of the main project. Extensions other than “.csproj”, “.tmp\_proj” and “.vcxproj” will not break, but do not receive any language-specific shared configuration.

Finally, the CustomDictionary.xml and TCWCS.ruleset files are specified for use by code analysis tools.

## Build\Common.Build.Cpp.settings

This file is included for projects with file extension “.vcxproj”. It maps the cross-language variables to those used by Microsoft.Cpp.targets.

The variable Language is set to “C++” identically to Microsoft.Cpp.targets.

C++ projects should have Platform set to either Win32 or x64.

The DEV11, \_DEBUG, NDEBUG, WIN32, WIN64 and \_WINDOWS preprocessor variables are defined as required based on the values of VSTarget, Configuration and Platform. There is no need to redefine these variables in individual projects.

Optimization settings are defined for release builds.

Due to a bug in Microsoft.Cpp.Win32.props, each C++ project must include <DebugInformationFormat>ProgramDatabase</DebugInformationFormat> for both debug and release builds.

## Build\Common.Build.CSharp.settings

This file is included for projects with file extension “.csproj” and auto-generated projects “.tmp\_proj”. It maps the cross-language variables to those used by Microsoft.CSharp.targets.

The variable Language is set to “C#” identically to Microsoft.CSharp.targets.

C# projects should have Platform set to Any CPU, x86 or x64.

The DEV11, DEBUG and TRACE constants are defined as required based on the values of VSTarget and Configuration. There is no need to redefine these variables in individual projects.

Optimization settings are defined for release builds.

## Build\Common.Build.Wix.settings

This file is included for projects with file extension “.wixproj”. It maps the cross-language variables to those used by Wix.targets.

The variable Language is set to “WiX”.

## Build\Common.Build.targets

This file is included at the end of projects.

When Language is set to “C#”, AssemblyInfoCommon.cs, AssemblyVersion.cs and Microsoft.CSharp.targets are included. If CreateVsixContainer is set to any value, Microsoft.VsSDK.targets is also imported.

When Language is set to “C++”, Microsoft.Cpp.targets is included.

If Language is not either of these values, the targets file should be included in the individual project.

## Build\Common.Build.Traversal.targets

This file is included in dir.proj files.

## Build\CustomDictionary.xml

This file contains words that should not be considered spelling errors by the code analysis tools.

## Build\FinalPublicKey.snk

This file contains the strong naming key to be used for delay signing assemblies.

## Build\TCWCS.ruleset, TCWCS.Tests.ruleset

These files contain the active code analysis rules for product and test projects, respectively.

## Release\Product\Python\ProjectBefore.settings

This file is imported into all product projects before defining any properties.

It defines BuildRoot and imports Common.Build.settings.

## Release\Product\Python\ProjectAfter.settings

This file is imported at the end of all product projects. It imports Common.Build.targets.

## Release\Product\Setup\BuildRelease.ps1

This script creates an unsigned release, including all installers for all configurations.

## Release\Product\Setup\BuildPublicRelease.ps1

This script creates a signed release, including all installers for all configurations.

## Release\Product\Setup\SetupProjectBefore.settings

This file is imported into all setup (.wixproj) projects before defining any properties.

It defines BuildRoot, imports Common.Build.settings and defines paths to WiX tools.

## Release\Product\Setup\SetupProjectAfter.settings

This file is imported at the end of all setup projects.

## Release\Tests\TestProjectBefore.settings

This file is imported into all product projects before defining any properties.

It defines BuildRoot and imports Common.Build.settings.

At present, this file is identical to ProjectBefore.settings.

## Release\Tests\TestProjectAfter.settings

This file is imported at the end of all product projects, after including files and before importing any targets files.

It imports Common.Build.targets.

At present, this file is identical to ProjectAfter.settings.

# Creating a new C# Project

Basic C# projects (those that do not use the VS SDK) can be created by making a normal project and immediately applying the following modifications with a text editor. Green text has been inserted, red text has been removed.

<?xml version="1.0" encoding="utf-8"?>

<Project ToolsVersion="4.0" DefaultTargets="Build" xmlns="http://schemas.microsoft.com/developer/msbuild/2003">

<Choose>

<When Condition=" '$(VisualStudioVersion)'=='11.0' Or '$(TargetVisualStudioVersion)'=='VS110' ">

<PropertyGroup>

<MinimumVisualStudioVersion>11.0</MinimumVisualStudioVersion>

<FileUpgradeFlags>

</FileUpgradeFlags>

<UpgradeBackupLocation>

</UpgradeBackupLocation>

<OldToolsVersion>4.0</OldToolsVersion>

</PropertyGroup>

</When>

</Choose>

<Import Project="..\ProjectBefore.settings" />

<PropertyGroup>

<Configuration Condition=" '$(Configuration)' == '' ">Debug</Configuration>

<Platform Condition=" '$(Platform)' == '' ">x86</Platform>

<ProductVersion>8.0.30703</ProductVersion>

<SchemaVersion>2.0</SchemaVersion>

<ProjectGuid>…</ProjectGuid>

<OutputType>WinExe</OutputType>

<AppDesignerFolder>Properties</AppDesignerFolder>

<RootNamespace>WindowsFormsApplication1</RootNamespace>

<AssemblyName>WindowsFormsApplication1</AssemblyName>

<TargetFrameworkVersion>v4.0</TargetFrameworkVersion>

<TargetFrameworkProfile>Client</TargetFrameworkProfile>

<FileAlignment>512</FileAlignment>

</PropertyGroup>

<PropertyGroup Condition=" '$(Configuration)|$(Platform)' == 'Debug|x86' ">

<PlatformTarget>x86</PlatformTarget>

<DebugSymbols>true</DebugSymbols>

<DebugType>full</DebugType>

<Optimize>false</Optimize>

<OutputPath>bin\Debug\</OutputPath>

<DefineConstants>DEBUG;TRACE</DefineConstants>

<ErrorReport>prompt</ErrorReport>

<WarningLevel>4</WarningLevel>

</PropertyGroup>

<PropertyGroup Condition=" '$(Configuration)|$(Platform)' == 'Release|x86' ">

<PlatformTarget>x86</PlatformTarget>

<DebugType>pdbonly</DebugType>

<Optimize>true</Optimize>

<OutputPath>bin\Release\</OutputPath>

<DefineConstants>TRACE</DefineConstants>

<ErrorReport>prompt</ErrorReport>

<WarningLevel>4</WarningLevel>

</PropertyGroup>

<PropertyGroup Condition=" '$(Platform)' == 'x86' ">

<PlatformTarget>x86</PlatformTarget>

</PropertyGroup>

<ItemGroup>

<!-- Reference/ProjectReference elements -->

</ItemGroup>

<ItemGroup>

<!-- Compile/Content elements -->

</ItemGroup>

<Import Project="$(MSBuildToolsPath)\Microsoft.CSharp.targets" />

<Import Project="..\ProjectAfter.settings" />

</Project>

The inserted <Choose> element allows the project to be loaded in both VS 2010 and VS 2012. Apart from <PlatformTarget>, all other platform or configuration specific settings should be inherited from ProjectBefore.settings. Replacing the <Import> of Microsoft.CSharp.targets with ProjectAfter.settings allows

# Creating a new C++ Project

All C++ projects require the following changes to work consistently within our build system. Project files should be for only a single platform, which should be hardcoded before defining configurations in order to support correct builds without the solution file (for example, when referenced from a WiX project).

By convention, all C++ projects target x64. Those that must target Win32 should have “X86” appended to the name of the project and output binaries. All binaries are stored in $(OutputPath), regardless of platform.

<?xml version="1.0" encoding="utf-8"?>

<Project DefaultTargets="Build" ToolsVersion="4.0" xmlns="http://schemas.microsoft.com/developer/msbuild/2003">

<Import Project="..\ProjectBefore.settings" />

<PropertyGroup>

<Platform>Win32</Platform>

</PropertyGroup>

<ItemGroup Label="ProjectConfigurations">

<ProjectConfiguration Include="Debug|Win32">

<Configuration>Debug</Configuration>

<Platform>Win32</Platform>

</ProjectConfiguration>

<ProjectConfiguration Include="Release|Win32">

<Configuration>Release</Configuration>

<Platform>Win32</Platform>

</ProjectConfiguration>

</ItemGroup>

<PropertyGroup Label="Globals">

<ProjectGuid>{6EC02348-9285-41B6-A87D-57EB17E71F8E}</ProjectGuid>

<Keyword>Win32Proj</Keyword>

<RootNamespace>Win32Proj</RootNamespace>

</PropertyGroup>

<Choose>

<When Condition=" '$(VisualStudioVersion)'=='11.0' Or '$(TargetVisualStudioVersion)'=='VS110' ">

<PropertyGroup>

<VCTargetsPath Condition="'$(VCTargetsPath11)' != '' and '$(VSVersion)' == '' and $(VisualStudioVersion) == ''">$(VCTargetsPath11)</VCTargetsPath>

</PropertyGroup>

</When>

</Choose>

<Import Project="$(VCTargetsPath)\Microsoft.Cpp.Default.props" />

<PropertyGroup Condition="'$(Configuration)|$(Platform)'=='Debug|Win32'" Label="Configuration">

<PropertyGroup Condition="'$(Configuration) '=='Debug'" Label="Configuration">

<ConfigurationType>Application</ConfigurationType>

<UseDebugLibraries>true</UseDebugLibraries>

<CharacterSet>Unicode</CharacterSet>

</PropertyGroup>

<PropertyGroup Condition="'$(Configuration)|$(Platform)'=='Release|Win32'" Label="Configuration">

<PropertyGroup Condition="'$(Configuration) '=='Release'" Label="Configuration">

<ConfigurationType>Application</ConfigurationType>

<UseDebugLibraries>false</UseDebugLibraries>

<WholeProgramOptimization>true</WholeProgramOptimization>

<CharacterSet>Unicode</CharacterSet>

</PropertyGroup>

<PropertyGroup Label="Configuration" Condition=" '$(VisualStudioVersion)'=='11.0' Or '$(TargetVisualStudioVersion)'=='VS110' ">

<PlatformToolset>v110</PlatformToolset>

</PropertyGroup>

<Import Project="$(VCTargetsPath)\Microsoft.Cpp.props" />

<ImportGroup Label="ExtensionSettings">

</ImportGroup>

<ImportGroup Label="PropertySheets" Condition="'$(Configuration)|$(Platform)'=='Debug|Win32'">

<Import Project="$(UserRootDir)\Microsoft.Cpp.$(Platform).user.props" Condition="exists('$(UserRootDir)\Microsoft.Cpp.$(Platform).user.props')" Label="LocalAppDataPlatform" />

</ImportGroup>

<ImportGroup Label="PropertySheets" Condition="'$(Configuration)|$(Platform)'=='Release|Win32'">

<Import Project="$(UserRootDir)\Microsoft.Cpp.$(Platform).user.props" Condition="exists('$(UserRootDir)\Microsoft.Cpp.$(Platform).user.props')" Label="LocalAppDataPlatform" />

</ImportGroup>

<ImportGroup Label="PropertySheets">

<Import Project="$(UserRootDir)\Microsoft.Cpp.$(Platform).user.props" Condition="exists('$(UserRootDir)\Microsoft.Cpp.$(Platform).user.props')" Label="LocalAppDataPlatform" />

</ImportGroup>

<PropertyGroup Label="UserMacros" />

<PropertyGroup Condition="'$(Configuration)|$(Platform)'=='Debug|Win32'">

<LinkIncremental>true</LinkIncremental>

</PropertyGroup>

<PropertyGroup Condition="'$(Configuration)|$(Platform)'=='Release|Win32'">

<LinkIncremental>false</LinkIncremental>

</PropertyGroup>

<PropertyGroup>

<IntDir>obj\$(Configuration)\$(Platform)\</IntDir>

<LinkIncremental>false</LinkIncremental>

</PropertyGroup>

<ItemDefinitionGroup Condition="'$(Configuration)|$(Platform)'=='Debug|Win32'">

<ItemDefinitionGroup Condition="'$(Configuration)'=='Debug'">

<ClCompile>

<PrecompiledHeader>Use</PrecompiledHeader>

<WarningLevel>Level3</WarningLevel>

<Optimization>Disabled</Optimization>

<PreprocessorDefinitions>WIN32;\_DEBUG;\_WINDOWS;%(PreprocessorDefinitions)</PreprocessorDefinitions>

<DebugInformationFormat>ProgramDatabase</DebugInformationFormat>

</ClCompile>

<Link>

<SubSystem>Windows</SubSystem>

<GenerateDebugInformation>true</GenerateDebugInformation>

</Link>

</ItemDefinitionGroup>

<ItemDefinitionGroup Condition="'$(Configuration)|$(Platform)'=='Release|Win32'">

<ClCompile>

<WarningLevel>Level3</WarningLevel>

<PrecompiledHeader>Use</PrecompiledHeader>

<Optimization>MaxSpeed</Optimization>

<FunctionLevelLinking>true</FunctionLevelLinking>

<IntrinsicFunctions>true</IntrinsicFunctions>

<PreprocessorDefinitions>WIN32;NDEBUG;\_WINDOWS;%(PreprocessorDefinitions)</PreprocessorDefinitions>

<DebugInformationFormat>ProgramDatabase</DebugInformationFormat>

</ClCompile>

<Link>

<SubSystem>Windows</SubSystem>

<GenerateDebugInformation>true</GenerateDebugInformation>

<EnableCOMDATFolding>true</EnableCOMDATFolding>

<OptimizeReferences>true</OptimizeReferences>

</Link>

</ItemDefinitionGroup>

<ItemGroup>

<!-- ClInclude elements -->

</ItemGroup>

<ItemGroup>

<ClCompile Include="stdafx.cpp">

<PrecompiledHeader Condition="'$(Configuration)|$(Platform)'=='Debug|Win32'">Create</PrecompiledHeader>

<PrecompiledHeader Condition="'$(Configuration)|$(Platform)'=='Release|Win32'">Create</PrecompiledHeader>

<PrecompiledHeader>Create</PrecompiledHeader>

</ClCompile>

<!-- ClCompile elements -->

</ItemGroup>

<Import Project="$(VCTargetsPath)\Microsoft.Cpp.targets" />

<Import Project="..\ProjectAfter.settings" />

<ImportGroup Label="ExtensionTargets">

</ImportGroup>

</Project>

# Creating a new VSPackage Project

Follow the instructions as for [creating a C# project](#_Creating_a_new), and add the following lines to the unconditional PropertyGroup element:

<TargetFrameworkVersion>v4.0</TargetFrameworkVersion>

<TargetFrameworkProfile>Client</TargetFrameworkProfile>

<FileAlignment>512</FileAlignment>

<DeployExtension>True</DeployExtension>

<CreateVsixContainer>True</CreateVsixContainer>

</PropertyGroup>

Common.Build.targets uses CreateVsixContainer to determine whether to import Microsoft.VsSDK.targets: if it has been set, regardless of its value, the VS SDK is included. This may produce an error if no .vsixmanifest file has been included.

## Including a C# project reference in VSIX

Managed assemblies are automatically included in the VSIX when they are known dependencies. VS can be used to add a reference to the project or they may be added manually.

Private references are the default. The output assembly of these references will be included in the VSIX and copied to the installation directory for that extension.

<ProjectReference Include="..\Analysis\Analysis.csproj">

<Project>{A85D479D-67A9-4BDB-904A-7D86DAF68A6F}</Project>

<Name>Microsoft.PythonTools.Analysis</Name>

</ProjectReference>

Public references are created by changing the Private property to false, or manually adding a <Private> element. These references do not copy the output assembly and rely on being able to load types by name at runtime.

<ProjectReference Include="..\PythonTools\PythonTools.csproj">

<Project>{FA7BE5F5-E04F-4613-B7AC-70CE10D1BB68}</Project>

<Name>Microsoft.PythonTools</Name>

<Private>False</Private>

</ProjectReference>

## Including a C++ project reference in VSIX

References to C++ projects require two elements. The first is a ProjectReference following this pattern:

<ProjectReference Include="..\VsPyProf\VsPyProf.vcxproj">

<Project>{6EC75EDA-17AC-453E-BA34-CB6796C4319B}</Project>

<Name>VsPyProf</Name>

<ReferenceOutputAssembly>false</ReferenceOutputAssembly>

<LinkLibraryDependencies>false</LinkLibraryDependencies>

<IncludeOutputGroupsInVSIX>GetCopyToOutputDirectoryItems</IncludeOutputGroupsInVSIX>

</ProjectReference>

The ReferenceOutputAssembly, LinkLibraryDependencies and IncludeOutputGroupsInVSIX are required to ensure that the output is not included in the VSIX but the project is built as a dependency. The reason for this is that the VSIX generator does not use the correct output path for C++ projects and attempts to copy the binary from the wrong location. Visual Studio raises some warnings about these references but will build correctly. MSBuild produces no warnings.

To correctly include the binary, a Content element is required:

<Content Include="$(OutputPath)VsPyProf.dll">

<Link>VsPyProf.dll</Link>

<IncludeInVSIX>true</IncludeInVSIX>

</Content>

As long as the referenced project is correctly using our build system, the binary will always be found at $(OutputPath). Visual Studio may warn about the target not being found but will build correctly. MSBuild produces no warnings.

# Updating WiX projects