如何：将可扩展项目迁移到 Visual Studio 2017

原文https://docs.microsoft.com/zh-cn/visualstudio/extensibility/how-to-migrate-extensibility-projects-to-visual-studio-2017

本文内容

本文档说明如何将可扩展性项目升级到Visual Studio 2017。除了描述如何更新项目文件，还介绍了如何从版本2（VSIX V2）的扩展清单（extension manifest）升级到新版本3的VSIX清单（manifest）格式（VSIX V3）。

安装含有相应工作负载的 Visual Studio 2017

确保你的安装包括以下工作负载：

1. .NET 桌面开发
2. Visual Studio 扩展开发

在 Visual Studio 2017 中打开 VSIX 解决方案

所有VSIX项目会要求主版本单向升级到Visual Studio 2017。

项目文件（例如 *.csproj）将被更新：

1. MinimumVisualStudioVersion项 - 现设为15.0
2. OldToolsVersion项（如果先前存在） - 现设为14.0

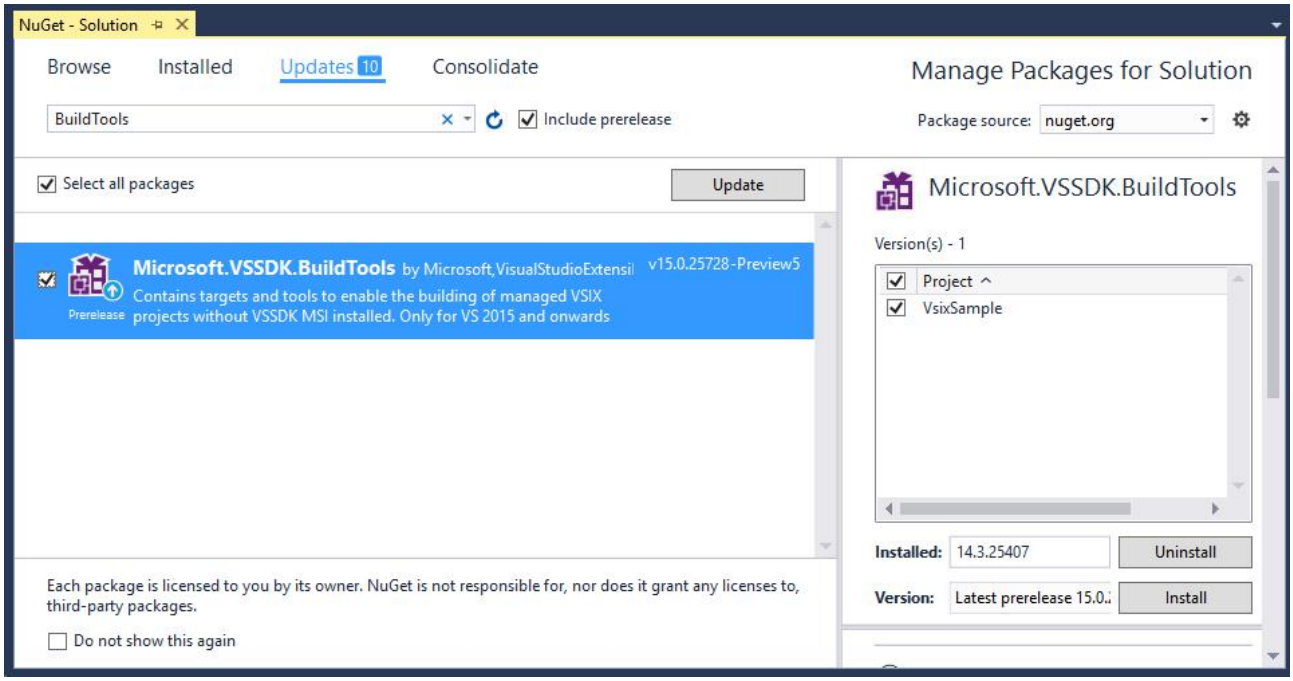
更新 Microsoft.VSSDK.BuildTools NuGet 包

> 注意：如果你的解决方案不引用 Microsoft.VSSDK.BuildTools NuGet 包，你可以跳过这一步。

为了在新的VSIX v3（版本3）创建你的扩展，你的解决方案需要新的VSSDK构建工具（VSSDK Build Tool）。这会和Visual Studio 2017一起安装，但你的VSIX v2扩展可能仍通过NuGet引用旧的版本。如果是这样，你将需要为解决方案手动安装的Microsoft.VSSDK.BuildTools NuGet包的更新。

1. 更新Microsoft.VSSDK.BuildTools NuGet引用：
2. 右击解决方案，选择**管理解决方案的NuGet程序包...**
3. 选择到**更新**选项卡。
4. 选择Microsoft.VSSDK.BuildTools (latest version)。

5.点击更新。



改变 VSIX 扩展清单

为了确保用户安装的Visual Studio具有运行扩展所需的所有程序集，请在扩展清单（extension manifest）文件中指定所有必需的组件或包。用户尝试安装扩展时，VSIXInstaller会检查是否安装了所有的必备条件。如果缺少某些组件，会作为扩展安装过程的一部分来提示用户安装缺少的组件。

> 注意：至少，所有扩展都应该将 Visual Studio 核心编辑器组件作为必备条件。

1. 编辑扩展清单（extension manifest）文件（通常叫source.extension.vsixmanifest）。
2. 确保InstallationTarget项包括15.0。
3. 添加安装所需的必备条件（如下面的示例所示）。
    1. 我们建议你只指定安装必备条件的组件ID。

2. 这部分见本文档结尾部分关于识别组件ID的说明。

示例：

```
<PackageManifest>
 ...
   <Prerequisites>
      <Prerequisite Id="Microsoft.VisualStudio.Component.CoreEditor" Version="[15.0,16.
0)" />
      <Prerequisite Id="Microsoft.VisualStudio.Component.DiagnosticTools" Version="[15.
0.25814.0,16.0)" />
      <Prerequisite Id="Microsoft.VisualStudio.Shell.12.0" Version="[12.0]" />
   </Prerequisites>
 ...
</PackageManifest>
```

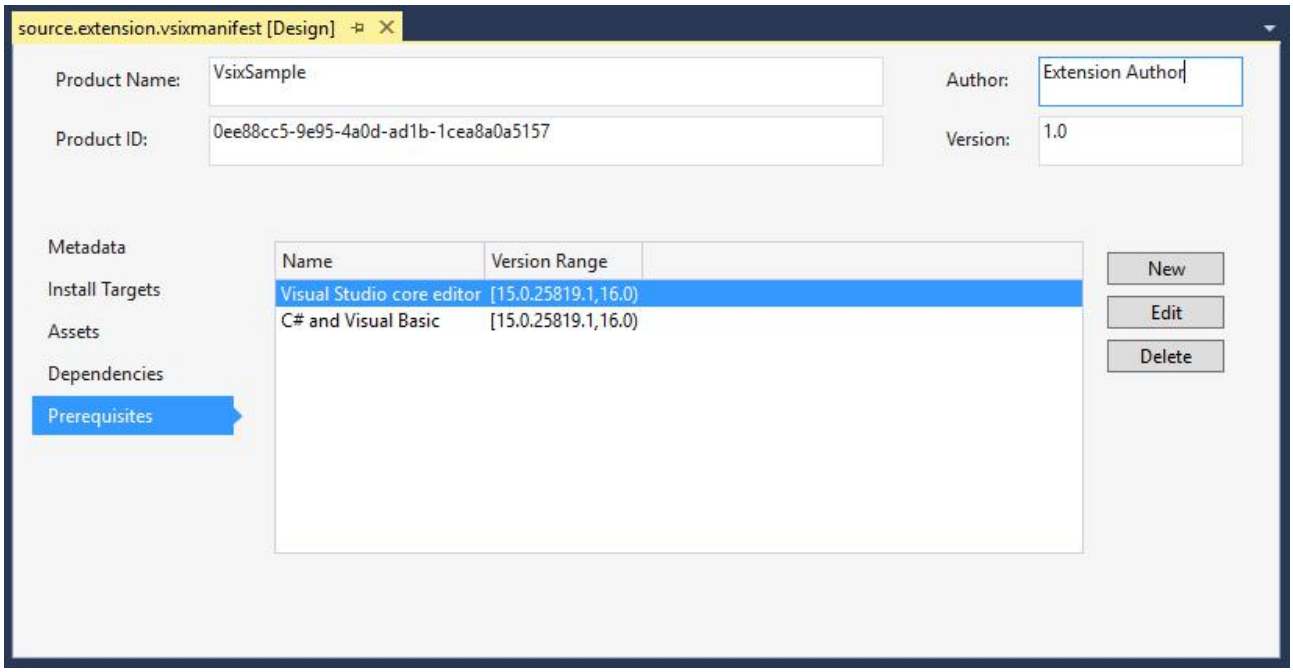
其他方法：使用设计器来修改VSIX扩展清单（extension manifest）

除了直接编辑XML文档，你可以使用清单设计器（Manifest Designer）中的Prerequisites选项卡来选择必备条件，XML文档将会自动更新。

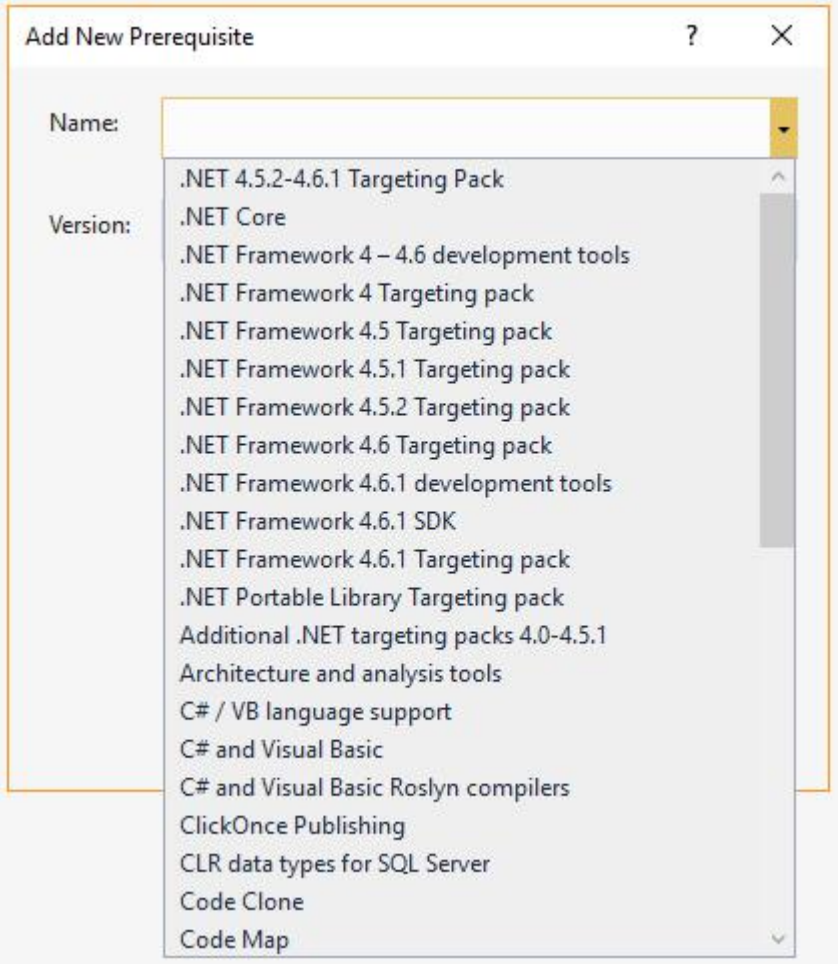
> 注意：清单设计器只允许你选择当前 Visual Studio 实例上安装的组件（而不是工作负载或包）。如果需要添加当前未安装的工作负载、包或组件为先决条件，请直接编辑清单的 XML 文件。

1. 打开source.extension.vsixmanifest [Design]文件。

2. 选择Prerequisites选项卡并点击New按钮。

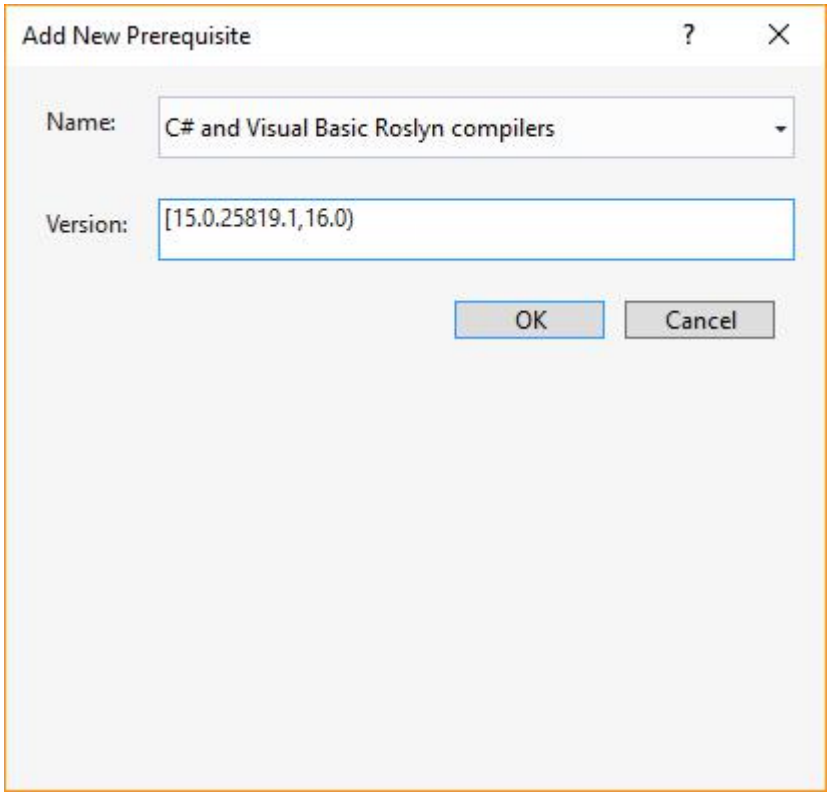


3. 会打开Add New Prerequisite窗口。



4. 单击Name下拉列表并选择所需的必备条件。
5. 如果需要则更新版本。

> 注意：版本字段这个版本字段会以当前安装的组件的版本预先填充，其范围扩展到（但不包括）组件的下一个主要版本。



6. 点击OK。

如果你正在从 Preview 4 或 Preview 5 迁移过来

1. Prerequisites标签替换SetupDependencies标签并且把Installer标签内的元素移出来。现在Prerequisites标签就直接PackageManifest标签里。

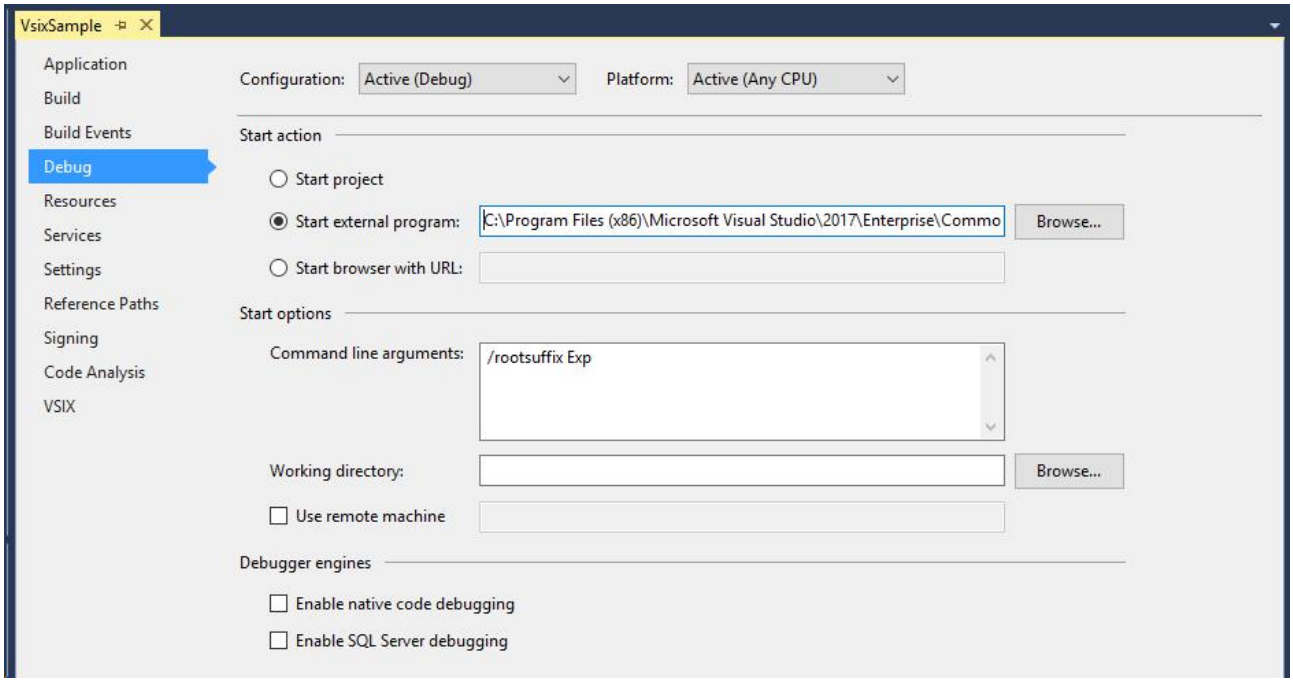
2. [可选] 删除GenerateVsixV3元素。（只在Preview 5要求这样做）不在Preview 5里，GenerateVsixV3元素会被忽略掉。

更新项目的调试设置

如果你希望在Visual Studio的实验实例里调试你的扩展，确保**调试** > **开始调试**在项目设置里为**启动外部程序**：其值是你的Visual Studio 2017的devenv.exe文件。

看起来像这样：

```
C:\Program Files (x86)\Microsoft Visual Studio\2017\Enterprise\Common7\IDE\devenv.exe
```



> 注意：**开始调试**的设置通常存储在.csproj.user 文件里。.gitignore 文件通常会包含这个文件，因此当提交到源代码管理时通常不与其他项目文件一起保存。因此，如果你从源代码管理中拉取新的解决方案，那么该项目很可能不会为**开始调试**设置值。用 Visual Studio 2017 创建新的 VSIX 项目与会生成默认指向当前 Visual Studio 安装目录的.csproj.user 文件。但是如果你正迁移 VSIX v2 扩展，有可能的.csproj.user 文件将包含对以前 Visual Studio 安装目录的引用。当你调试你的扩展时，设置好的**调试** > **开始调试**会允许 Visual Studio 实验实例启动。

检查扩展被正确建立（如 VSIX v3）

1. 创建VSIX项目
2. 解压生成的VSIX。
    1. 默认情况下，VSIX文件在bin/Debug或者bin/Release目录下，名为[你的扩展的名字].vsix。
    2. 把后缀.vsix改为.zip，很容易查看它的内容。
3. 检查以下三个文件是否存在：
    1. extension.vsixmanifest
    2. manifest.
    3. catalog.json

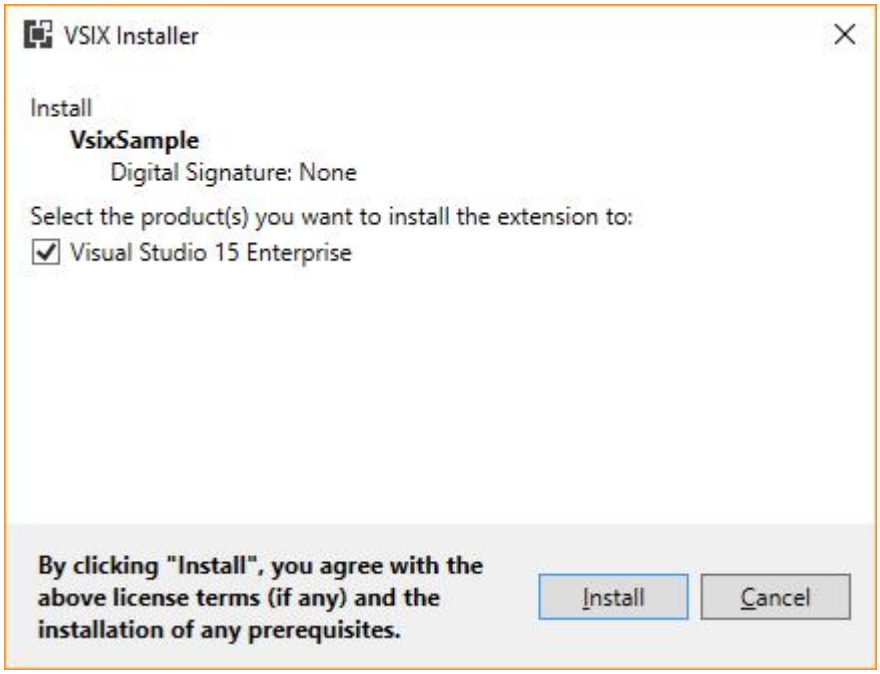
检查是否安装了所有必备条件

试验发现这个VSIX可以成功安装在安装了所有必备条件的机器上。

> 注意：在安装任何扩展之前，请关闭 Visual Studio 的所有实例。

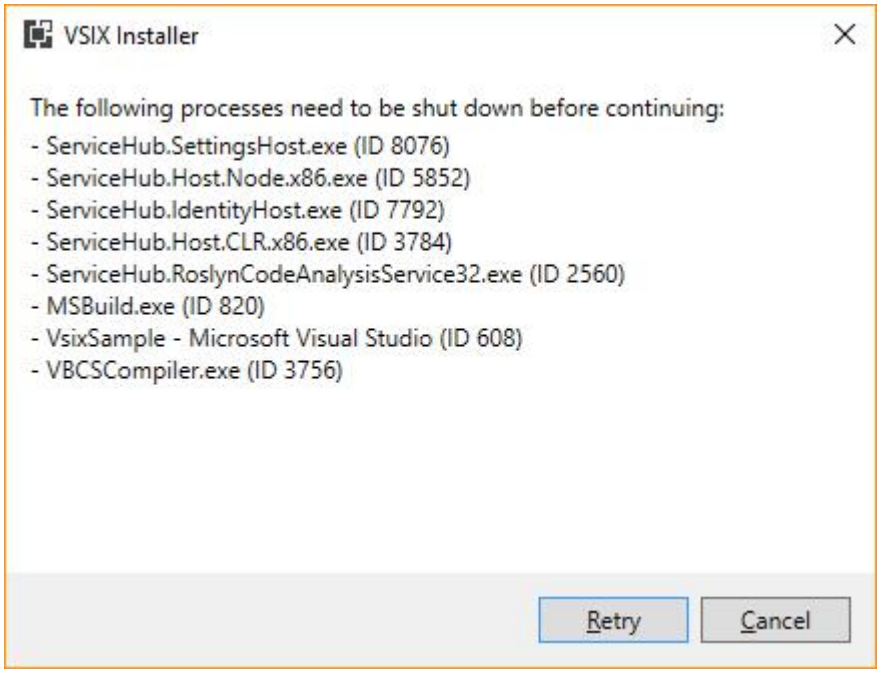
尝试安装扩展：

1. 在Visual Studio 2017上



2. 可选：尝试在以前版本的Visual Studio安装。
    1. 证明向后兼容的能力。
    2. 应该在Visual Studio 2012，Visual Studio 2013和Visual Studio 2015上可以运行。
3. 可选：检查发现VSIX安装版本检查器提供了一些可选择的版本。
    1. 包括Visual Studio以前的版本（如果已安装）。
    2. 包括Visual Studio 2017。

如果打开Visual Studio，你也许会看到这样的对话框：

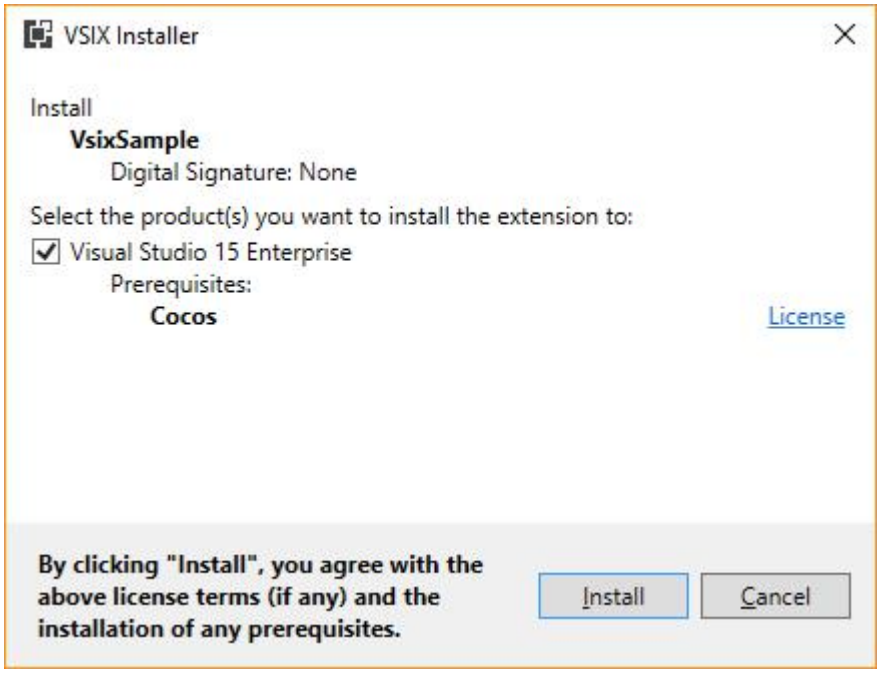


等待进程关闭，或者手动结束任务。你可以通过列出的名称找到进程，也可以使用括号里列出的PID。

> 注意：当 Visual Studio 实例运行时，这些进程不会自动关闭。确保关闭了 Visual Studio 的所有实例——包括来自其他用户的 Visual Studio 实例，然后继续重试。

检查缺少的必备条件

1. 尝试在装有Visual Studio 2017的机器上安装扩展 ,但该机器不拥有上面Prerequisites里定义的所有组件。
2. 检查发现安装器自动识别了缺少的组件并且在VSIXInstaller里作为必备条件列出了这些组件。
3. 注意：如果有必备条件需要和扩展一起安装，会要求进行升级。



确定要使用的组件

在查找依赖项时，你会发现一个依赖项可以映射到多个组件。为了确定应该使用依赖项，我们建议你选择与你的扩展具有类似功能的组件，并考虑用户以及他们最可能或者不介意安装的组件类型。我们还建议这样创建你的扩展：安装扩展所需的必备条件仅为其运行的最小条件，如果没有检测到指定的某些组件，可以关闭额外的特性。

为了提供进一步的指导，我们已经确定了几种常见的扩展类型及其建议的必备条件：

扩展类型	显示名称	Id
编辑器	Visual Studio 核心编辑器	Microsoft.VisualStudio.Component.CoreEditor
Roslyn 编译器	C #和 Visual Basic	Microsoft.VisualStudio.Component.Roslyn.LanguageServices

WPF	Managed Desktop Workload Core	Microsoft.VisualStudio.Component.ManagedDesktop.Core
调试器	Just-In-Time debugger	Microsoft.VisualStudio.Component.Debugger.JustInTime

查找组件 ID

按Visual Studio产品排序的组件列在Visual Studio 2017 Workload and Component IDs。在你的清单中使用这些组件ID作为Prerequisite ID。

如果你不确定哪个组件包含特定的二进制文件，请下载组件 -> 二进制文件映射电子表格。

vs2017-ComponentBinaryMapping.xlsx

Excel表中有四列：**组件名**（Component Name），**组件ID**（ComponentId），**版本**（Version），和**二进制/文件名**（Binary / File Names）。你以使用筛选查找指定的组件和二进制文件。

对你的所有引用，首先确定哪些在核心编辑器（Microsoft.VisualStudio.Component.CoreEditor）组件里。至少我们需要将核心编辑器组件指定为所有扩展的必备条件。对于那些不在核心编辑器中的引用，在**二进制/文件名**列添加筛选来查找具有这些引用子集的组件。

示例：

如果你一个调试器扩展并且知道你的项目引用VSDebugEng.dll和VSDebug.dll，点击**二进制/文件名**标题筛选按钮。搜索"VSDebugEng.dll"并点击OK。下一步点击**二进制文件/文件名**标题再次搜索"VSDebug.dll"。选择复选框"添加当前选项以筛选"并点击OK。现在，通过**组件名**查看与你的扩展最相关的组件。在这个示例里，你会选择Just-In-Time debugger并添加到你的vsixmanifest里。

如果知道你的项目要处理debugger元素，可以在筛选器搜索框中搜索"debugger"，以查看名称包含debugger的组件。