# Code Snippet Studio 2016 – User Guide

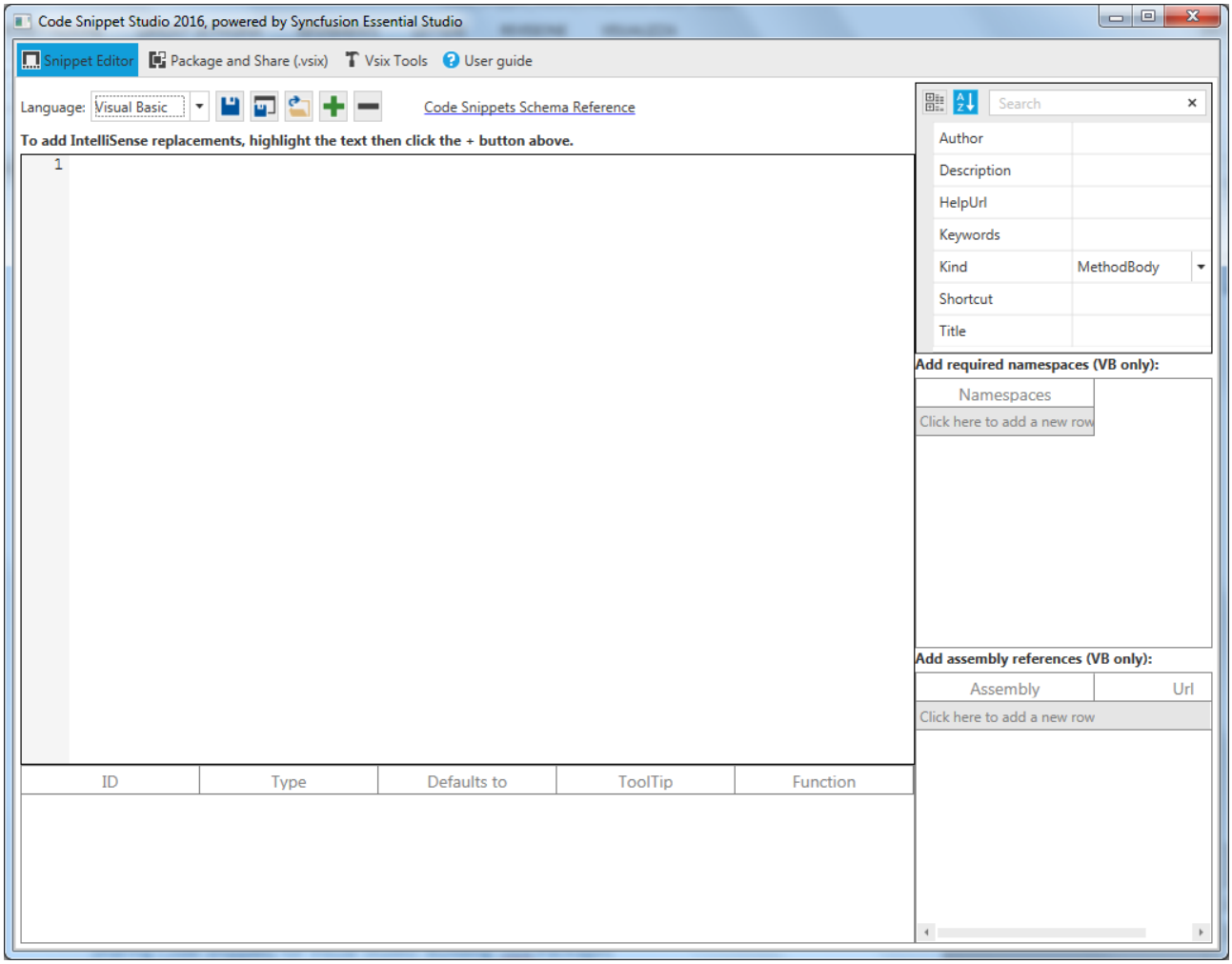
Welcome! Code Snippet Studio is an open source application that allows creating, editing, packaging, and sharing IntelliSense code snippets for Visual Studio 2015 and 2013, including the Express Editions. With Code Snippet Studio, you can:

- Create, edit, and save code snippets (.snippet files) via a convenient user interface and through a code editor that supports syntax highlighting.
- Share your code snippets with other developers by packaging your snippet files into a Visual Studio extensibility installer (.vsix file) which allows automating the installation of code snippets onto other machines under the form of a Visual Studio extension.
- Work with existing .vsix packages by adding digital signatures, extracting package contents, importing existing .vsix files, and even converting old-style .vsi content installers into the .vsix file format.

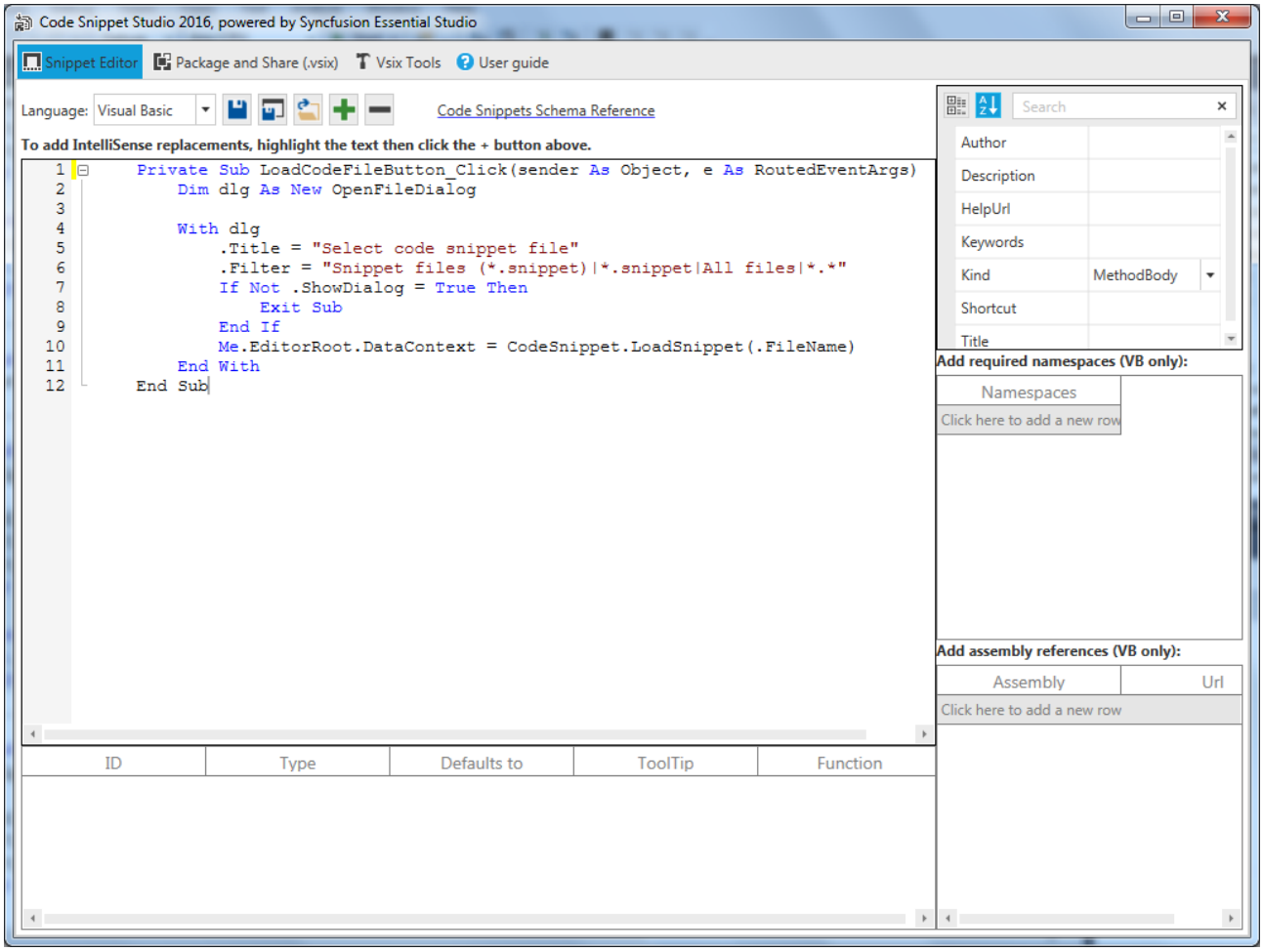
This guide explains with text and figures how to use the application. Code Snippet Studio is available as a stand-alone WPF application and as an extension for Visual Studio 2015 (tool window).

## Creating Code Snippets

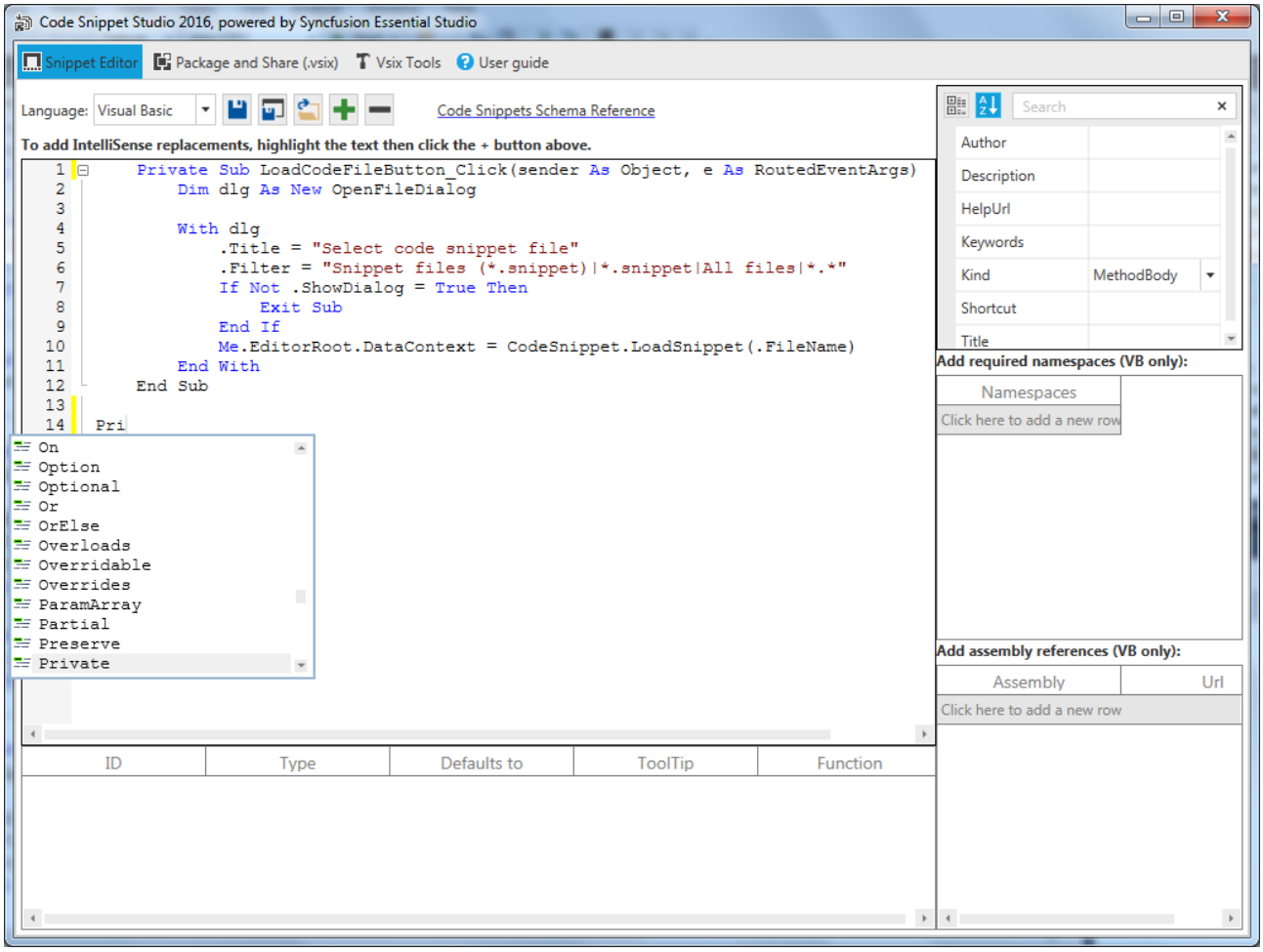
To create a code snippet, you click the Snippet Editor tab. This enables the proper user interface for editing a code snippet:




Select the programming language you want to write your snippet for, then type or paste your code:

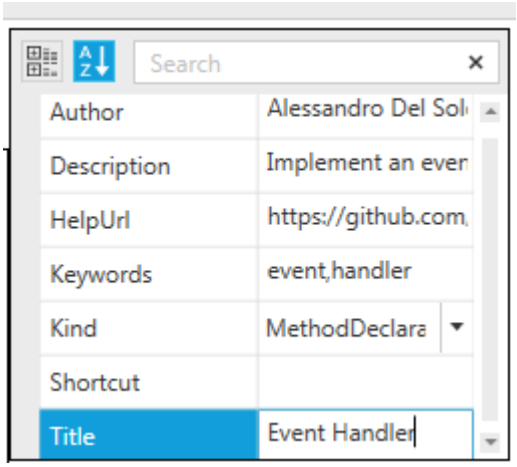



The code editor provides syntax highlighting, and offers basic IntelliSense features:




## Supplying Code Snippet Properties

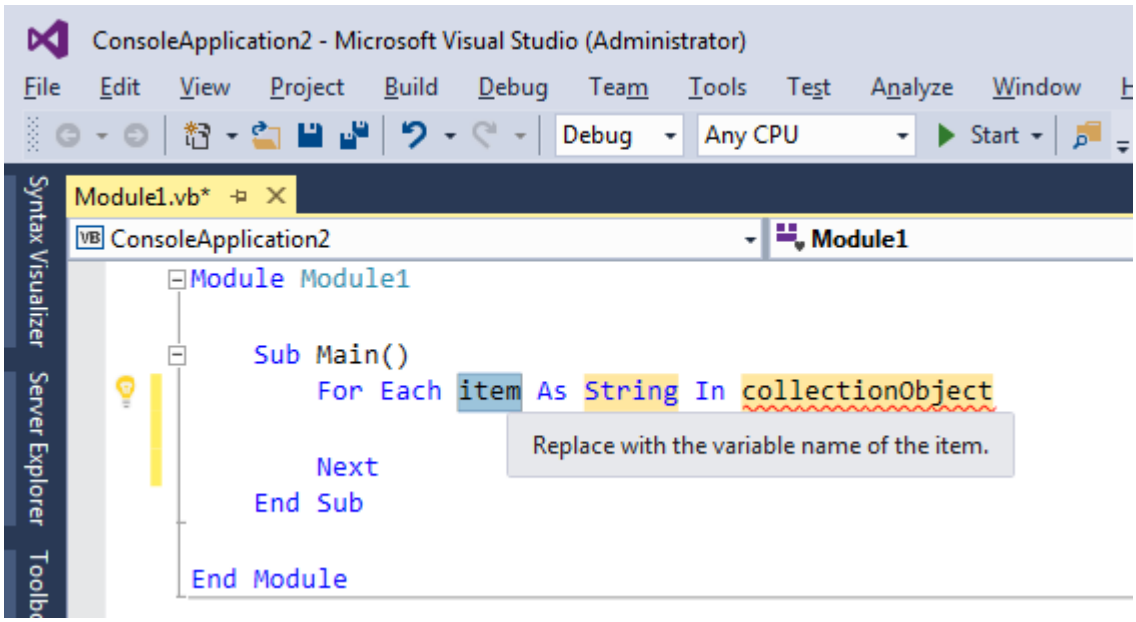
Once you have your code written, you must specify the code snippet properties. This can be done in the Properties dialog, in the upper right corner of the UI:




**Author**, **Description**, and **Title** are mandatory properties. All other properties are optional. Pay attention to the **Kind** property. Here you can specify what kind of code snippet you are creating (method body, method declaration, type declaration, a whole code file, or any kind of usage).

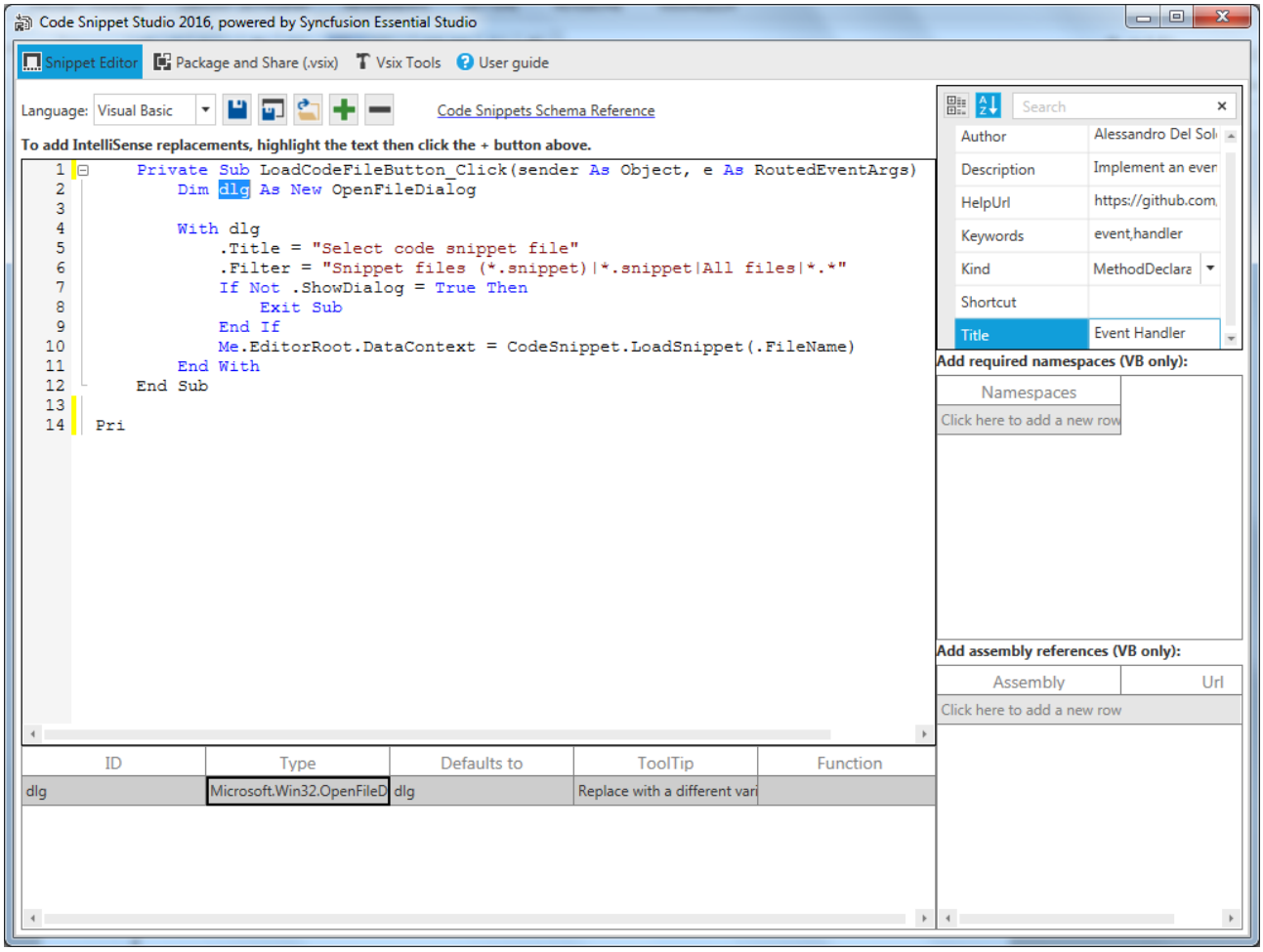
## Adding Declarations for IntelliSense replacements

Declarations allow the Visual Studio's IntelliSense to highlight words in the code and to provide suggestions, so that users can replace the highlighted word with a different one, like in the following example:




To add your declaration, follow these steps:

1. Select a word, identifier, or type name in the code editor.
2. Click the "+" button
3. When a new declaration definition appears in the grid below the code editor, enter the required information as in the following figure:



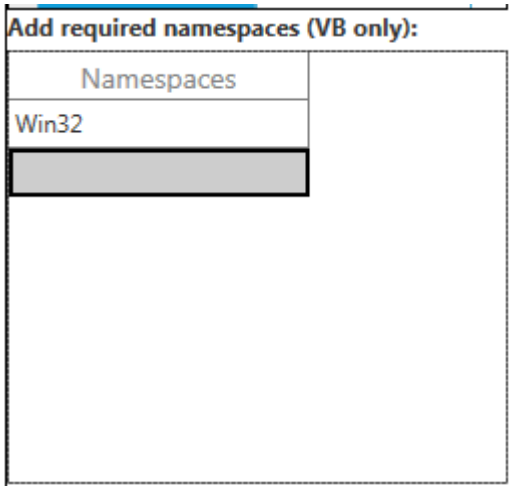

Notice that:

- “Defaults to” means the text you currently selected and should never be changed
- “ID” is a unique identifier for the declaration used by Visual Studio and can be renamed, however Code Snippet Studio automatically generates both field for you.
- “Type” represents the .NET type of the selected code, but is totally optional and not necessary with replacements that are not about types or type names.
- “ToolTip” must be filled with the text you want to be visible in the declaration description inside the code editor.

You can add as many declarations as you like.

## Specifying Required Namespaces (Visual Basic only)

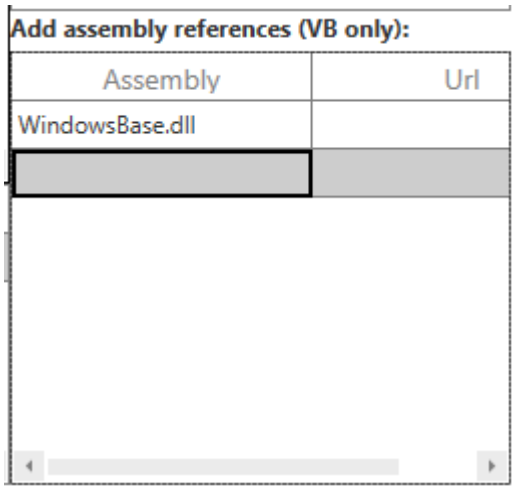
If your code snippets target Visual Basic, you can optionally specify a list of namespaces that are required for the snippet to be compiled properly. To accomplish this, you enter namespaces in the proper grid:




You must only enter the fully qualified name of the namespace, without the Imports keyword. If Code Snippet Studio detects that the language is not Visual Basic, this grid is disabled.

## Specifying Required Assembly References (Visual Basic only)

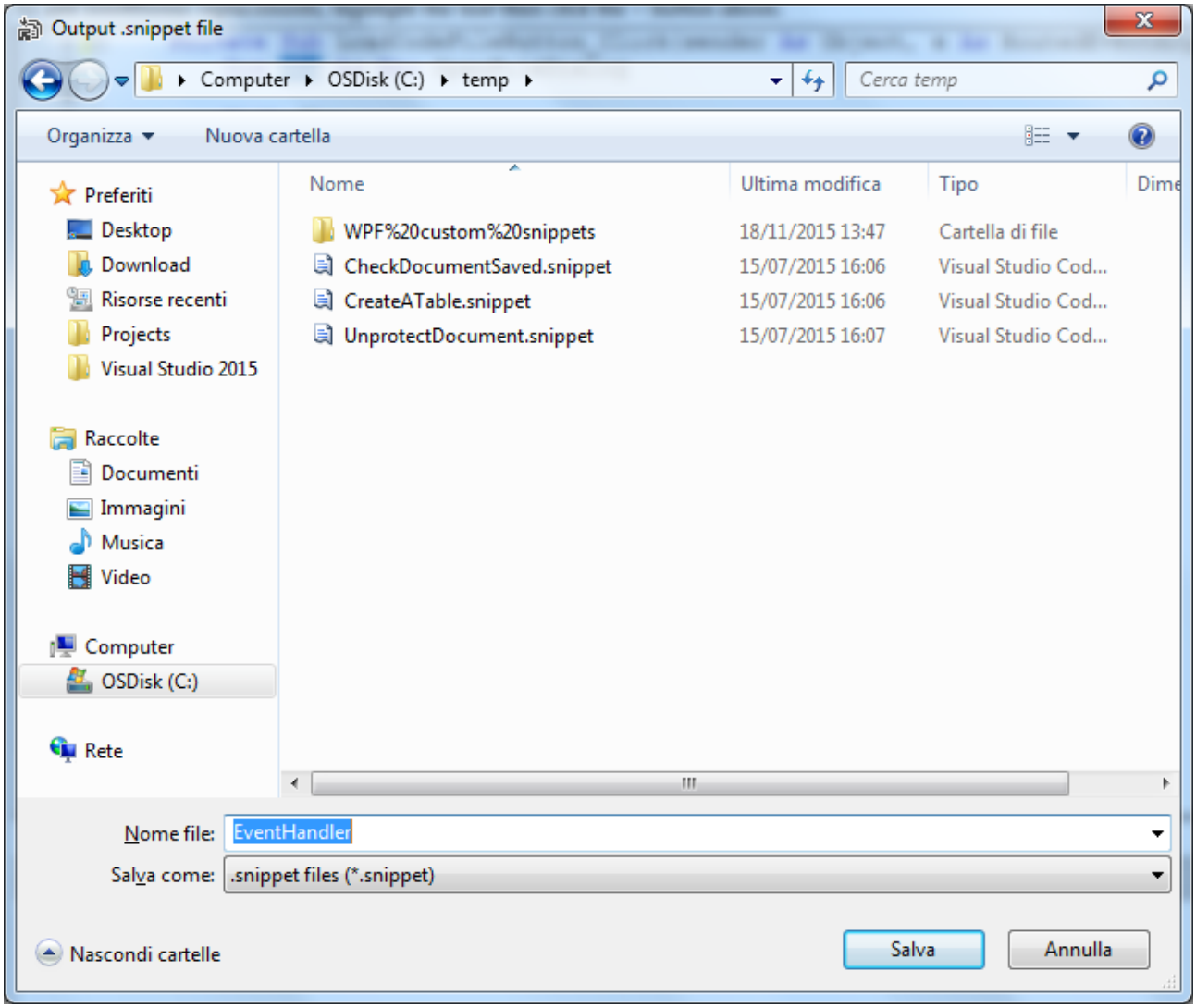
If your code snippets target Visual Basic, you can optionally specify a list of assemblies that are required for the snippet to be compiled properly. To accomplish this, you enter assembly names in the proper grid:




If the assembly is not a .NET assembly available in the GAC, you must supply the full pathname. Optionally, you can enter a Web address that users can click to discover more information about the assembly. If Code Snippet Studio detects that the language is not Visual Basic, this grid is disabled.

## Saving Code Snippets

You click the  button to save the code snippet as an IntelliSense code snippet file (.snippet):



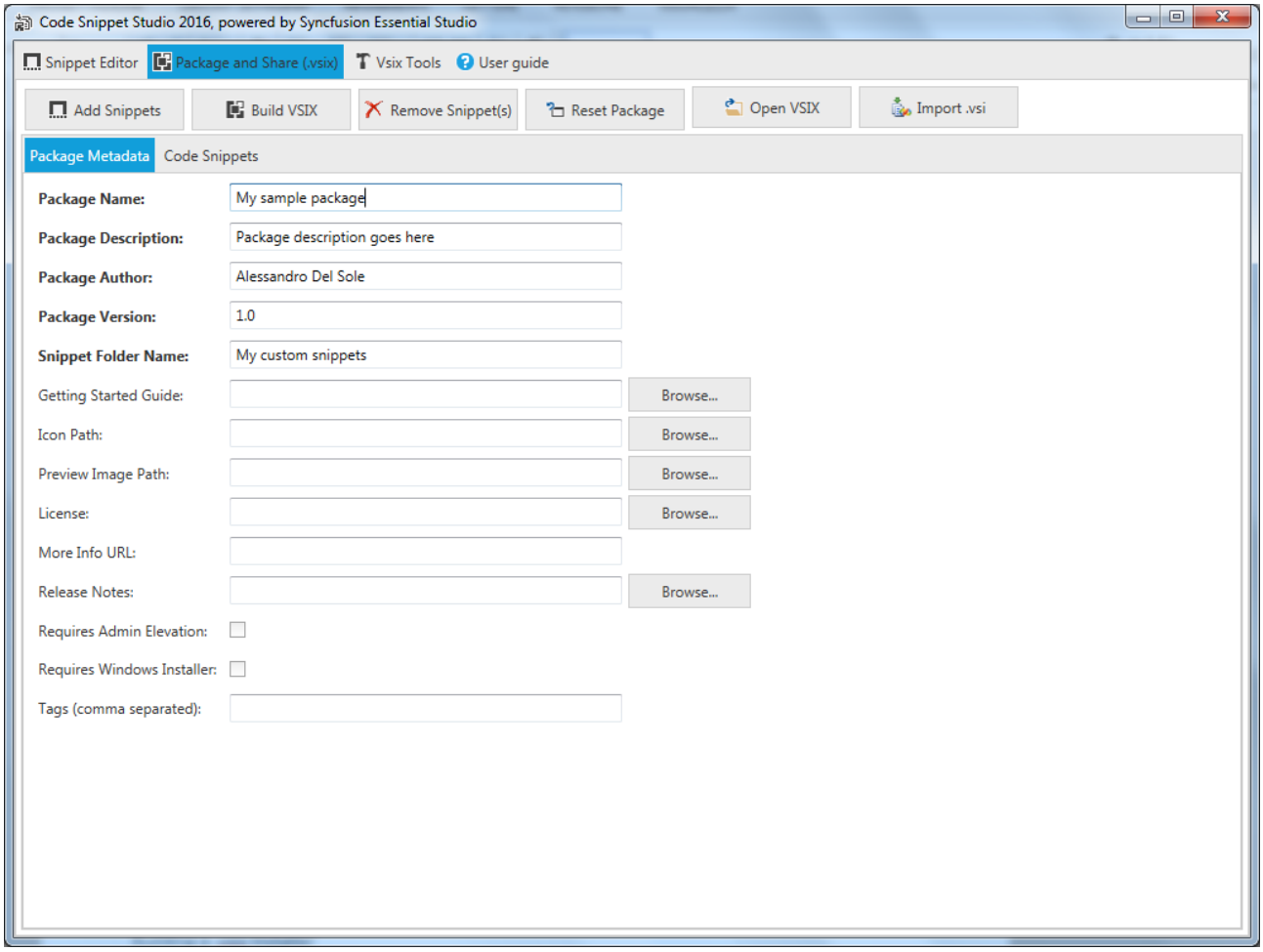

You can also save the code snippet as a plain code file that you can use in Visual Studio, by clicking  .

## Importing Code Snippets

You can open existing .snippet files by clicking  . Code Snippet Studio will load the .snippet file and will populate the user interface with the code, declarations, Imports, and References. So, you will be able to make further edits and to save back the code snippet.

## Sharing Code Snippets for Visual Studio: Building .vsix Packages

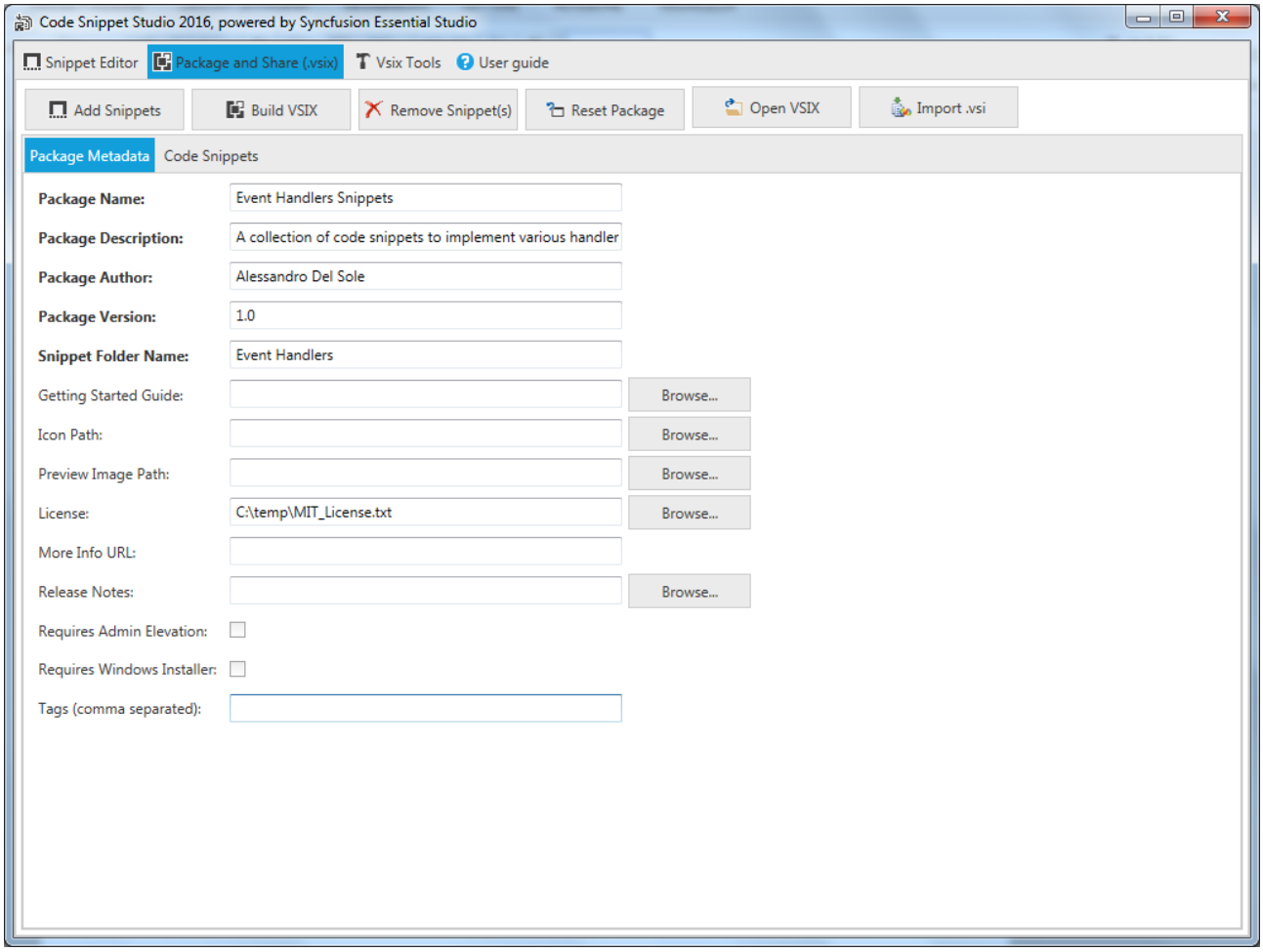
In order to make your code snippets available to Visual Studio's code editor and IntelliSense, these must be installed into the appropriate locations and you should update the VS' configuration manually. However, you can package your code snippets into a .vsix installer. The .vsix file format is typically used to share and install extensions for Visual Studio, but it is also the perfect choice to redistribute or simply install code snippets onto a different machine. Code Snippet Studio includes a complete environment to build a .vsix package for your code snippets, which you enable by clicking the **Package and Share (.vsix)** tab:




At a higher level, a .vsix package is made of the metadata, which identify the package, and of the actual content, in this case the .snippet files.

## Supplying the Package Metadata

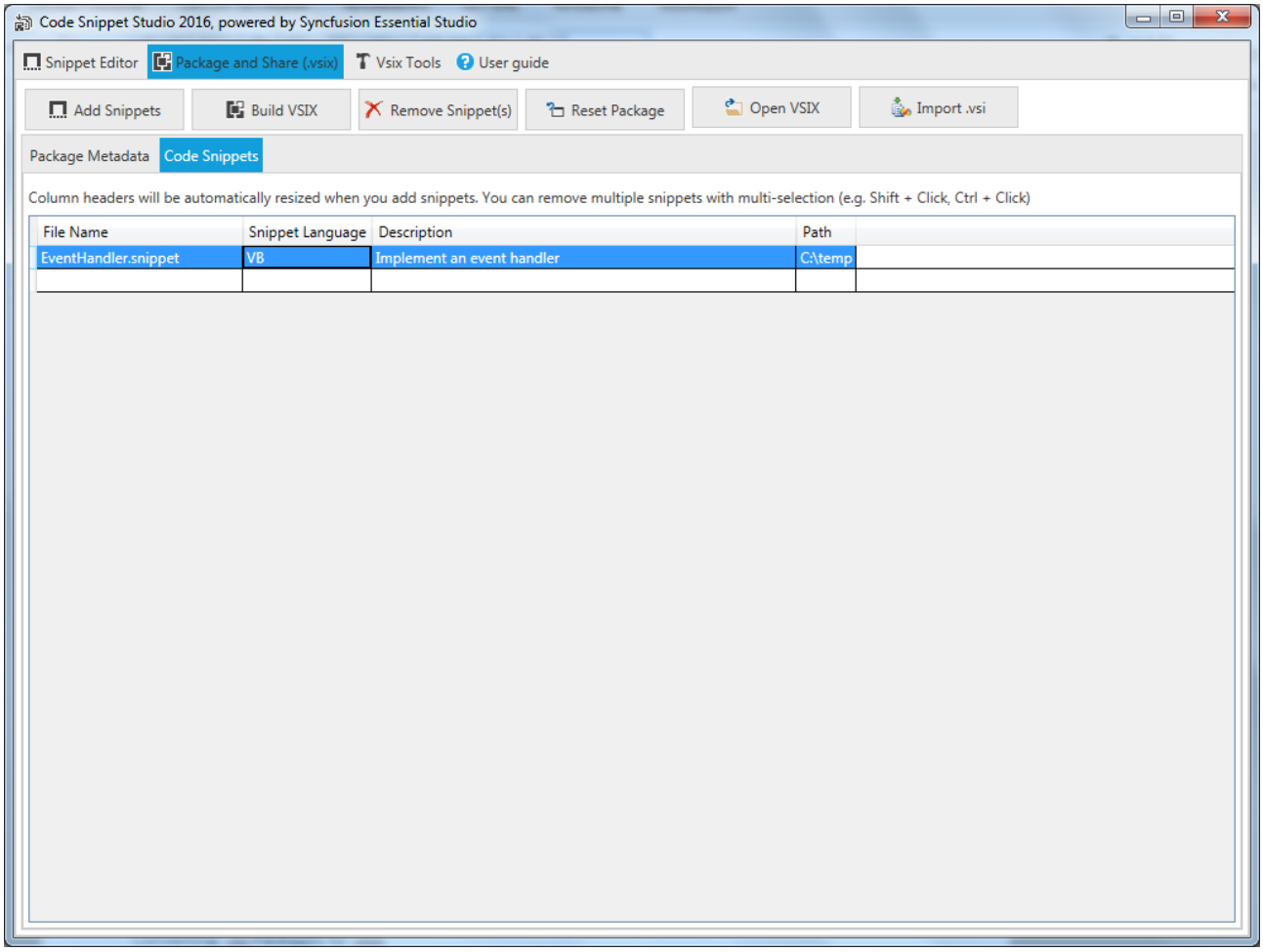
The package metadata contain all the information a user will see when installing your extension/code snippets, including the product name, the author, the license, and so on. You simply have to fill the required properties; mandatory properties are in bold. The following figure shows an example:




Pay attention to the **Snippet Folder Name**: This represents the folder name displayed in the IntelliSense and that will contain all the supplied code snippets, so it is very important that you provide a proper name for a better categorization.

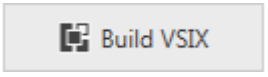
## Adding Code Snippet Files

After you specify the package metadata, you enter the code snippet files you want to package into the .vsix. To accomplish this, you click the **Code Snippets** tab, then click the **Add Snippets** button. In the appearing dialog, you will be able to multi-select an infinite number of .snippet files. Once selected, Code Snippet Studio appears like this:

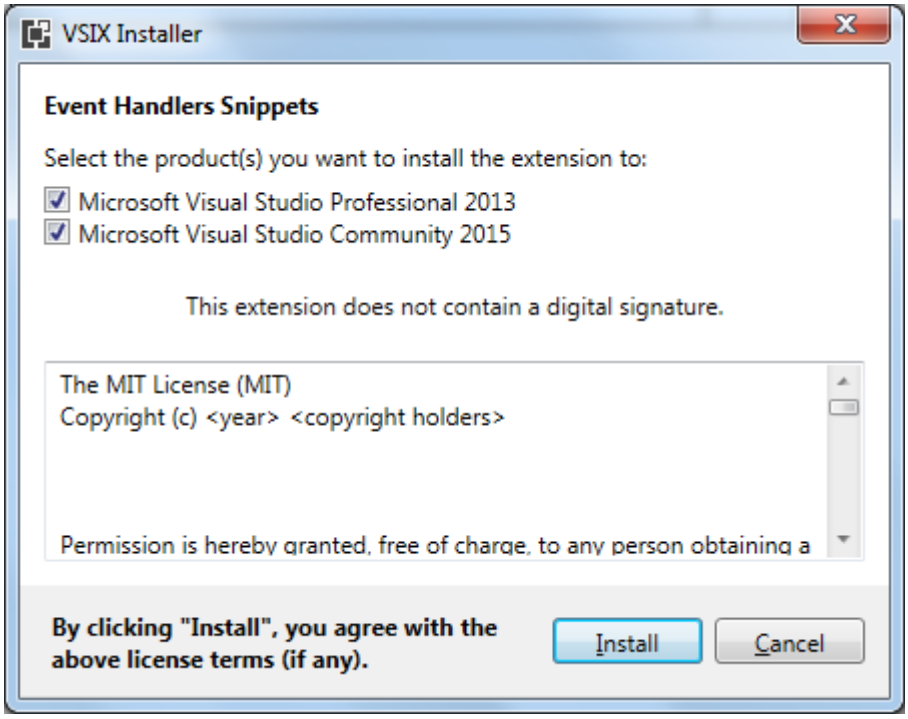



The information in this list is auto-generated based on the .snippet file analysis. You can remove snippet from this list and you can even completely reset the package with the **Remove Snippet(s)** and **Reset Package** buttons respectively.

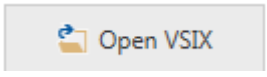
## Building a .vsix Installer

Once you have supplied both package metadata and code snippets, you can generate a .vsix package by simply clicking 

. You just specify the package name and you are done. When the generation completes, you will be asked if you want to immediately start the newly generated .vsix or not. The following figure shows the VSIX Installer in action, based on the sample package described before:

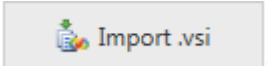



## Importing Existing Installers

Code Snippet Studio makes it easy to edit existing .vsix files by clicking 

. In this way, you can open any existing .vsix packages, not just those created by you, so that you can add other code snippets or make edits that do not violate copyrights.

**Important note**: opening an existing .vsix works only if the specified package contains at least one code snippet. All the other packaged extensions will be ignored, but the process will fail if no code snippet is detected in the package.

Code Snippet Studio also supports the old .vsi installer format, which developers used in Visual Studio 2005, 2008, and 2010 to package and share additional contents, including code snippets. If you have any existing .vsi packages from the past, you can easily convert them into a new package by clicking 

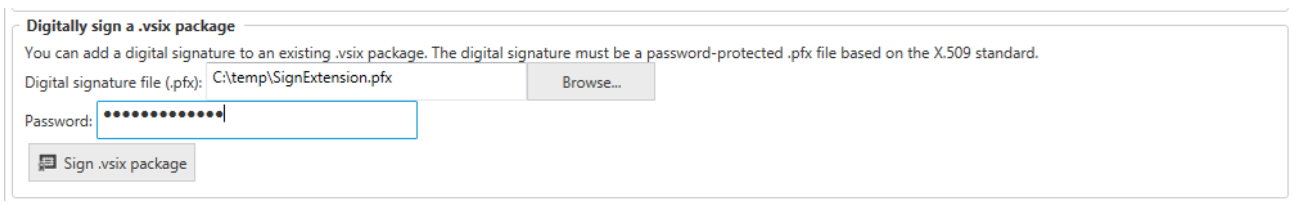
. This will not convert the .vsi to a .vsix directly (see the next section about that) but will give you an option to create a new package via the UI.

## Working With Packages: Signing, Importing, and Extracting .vsix Packages

Code Snippet Studio offers a number of additional useful tools to work with .vsix packages that you find under the **Vsix Tools** tab.

### Signing a .vsix Package

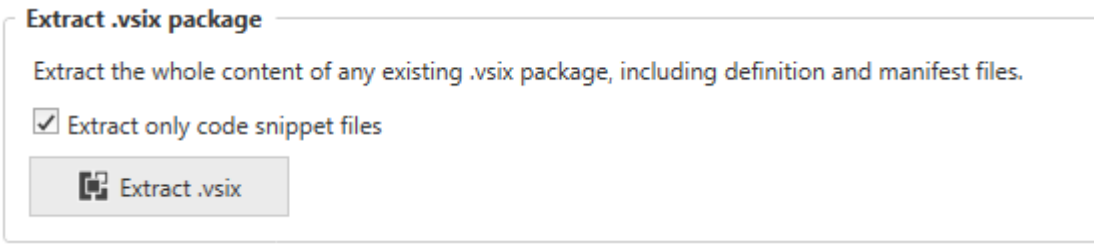
.vsix packages can be signed with a X.509 certificate stored in a .pfx password-protected file. To apply a digital signature, select a .pfx file and enter the password:



Next, click the **Sign .vsix package** button, pick up an existing .vsix file and wait for the operation to complete.

### Extracting a .vsix Package

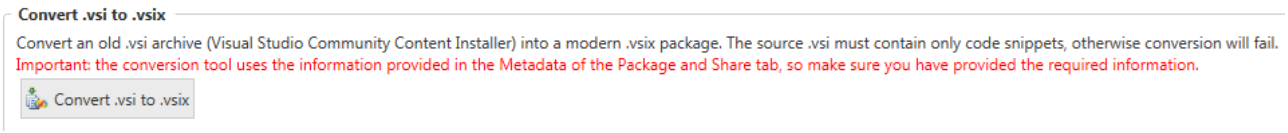
Code Snippet Studio gives you an option of extracting the content of a .vsix file into a folder:




If you select the **Extract only code snippet files** checkbox, the tool will only extract .snippet files from the .vsix package (if any). If it is unselected, the tool will extract the whole .vsix content, including the manifest file, the package definition file, extensions, and any other file.

### Converting .vsi Packages to .vsix

You heard about the old .vsi file format before. With Code Snippet Studio, you can easily convert a .vsi package into a .vsix one:




The only condition is that the source .vsi archive must contain at least one code snippet. To perform the conversion, follow these steps:

1. Open the **Package and Share (.vsix)** tab.
2. Fill-in the package metadata information (ignore the Code Snippets tab).
3. Go to **Vsix Tools** and click the **Convert .vsi to .vsix** button.

At this point, specify both the source and target file names and wait for the operation to complete.

## Credits and Information

Code Snippet Studio has been built using the following components:

Essential Studio for WPF by Syncfusion (Community license), a comprehensive set of high-quality controls for building amazing WPF user interfaces.

DelSole.VSIX library, an open source .NET library that provides APIs that make it easy to work with code snippets and .vsix archives.

## Collaboration: Join the Code Snippet Studio Project on GitHub

Code Snippet Studio is an open source project hosted on GitHub at: https://github.com/AlessandroDelSole/CodeSnippetStudio. If you want to become an active contributor on this project, create a fork and send a pull request or submit an issue.