

.NET Reflector® at a Glance



.NET Reflector was the first .NET assembly browser, allowing you to inspect, navigate, search and analyze assemblies and executables without the luxury of original source code. As originally developed by Lutz Roeder, Reflector decompiles assemblies back into source code. Red Gate took up his mantle and made .NET Reflector the first (and presently the *only*) such tool providing first-class debugging capability within Visual Studio.

There are three editions available: one desktop and two VS plugins (lower-right table). The desktop edition as a standalone utility provides more features per pound (lower-left table) but the VS plugin is arguably the “killer app” by providing live debugging capability. The accompanying 4-part series [.NET Reflector Through the Looking Glass](#) on Simple-Talk.com describes how to get started with both editions and how to take advantage of all they have to offer.

Navigating Reflector’s (Desktop) Assembly Browser

Node	Child Nodes	Description	Decompilation contents	Image
Assembly	Module(1), Resources folder(0..1)	Top-level nodes in the assembly browser (or second-level if within a zip file) identifying the assembly name and version. Because there is no single root above the assemblies the File menu provides a Collapse all Assemblies command for convenient cleanup.	Assembly details in part from VS >> Project >> Properties >> Application >> Assembly Information	
Zip	Assembly(1..n)	Reflector-introduced top-level nodes for assemblies loaded as zipped packages. Selecting a child assembly of a <i>zip node</i> yields exactly the same context as a top-level assembly node.	None	
Module	Namespaces(1..n), References folder(1)	Represented by an assembly’s file name in the assembly browser, the <i>module node</i> often provides no information. I chanced about one assembly in the figure with some data, but most framework classes simply indicate “UnverifiableCode” whereas all of my own classes present no data at all.	Some meta details about the assembly; sometimes “UnverifiableCode”; sometimes no data.	
Namespace	Type(1..n) Struct(1..n) Delegate(1..n), Interface(1..n), Enum(1..n)	A <i>namespace node</i> is the entry point to the actual code of an assembly. On entry it provides an overview of the assembly allowing quick scanning. You can then dive deeper either by picking an individual element in the assembly browser, or use the Expand Types link at the bottom of the overview.	Source code of a namespace with each element collapsed to one line. The Expand Types link at the bottom expands all elements allowing scrolling access to all in one view.	
Type	Base Types(1), Derived Types(1), Constructors(0..n), Events(0..n), Properties(0..n), Methods(0..n), Fields(0..n), Nested Types(0..n)	A <i>type node</i> displays an overview of the selected type allowing quick scanning. Properties, methods, events, and fields are set apart in labeled grouped. You can then dive deeper either by picking an individual element in the assembly browser, or use the Expand Methods link at the bottom of the overview.	Source code of a type with each element collapsed to one line. The Expand Methods link at the bottom expands all methods in place allowing scrolling access to all in one view.	
Property	Getter(0..1), Setter(0..1)	A <i>property node</i> displays the full code of a property. With auto-implemented properties this amounts to just a single line, but even with those, you have child nodes to see the compiler-generated code.	Source code of a property.	
Method	None	A <i>method node</i> displays the full code of a method.	Source code of a method.	
Field	None	A <i>field node</i> displays the single line declaration of a field.	Source code of a field.	

Reflector Features

The accompanying 4-part series [.NET Reflector Through the Looking Glass](#) provides an in-depth discussion of each item listed in this table. Each subheading corresponds to a separate part of the series. **[Bracketed]** items indicate functionality provided by Reflector add-ins.

Item	Standard (desktop) Meat and Potatoes	VS (plugin)
View code without source	Yes	Yes
Debug code without source	-NA-	Yes (<i>pro edition only</i>)
View code signature	Hover over code	Hover over code (normal VS behavior)
Manage bookmarks	Yes	Yes (normal VS behavior)
Filter code	Yes (Select all code or just public elements)	No
Filter navigation tree	Yes (select local (non-inherited) or all items)	No
Navigate to code from code	Yes	Yes
Navigate to code from documentation	Yes	-NA-
Search Code	Yes	Yes (version 8)
Copy code	Yes	Yes
The Pudding		
Select disassembly language	C#, VB, F#, C++, IL	C#, VB
Convert source files between .NET languages	Yes	-NA-
Decompile multiple assemblies at one time	Yes	Yes
Select framework or language version	Framework selectable: 1.0 through 4.5	Language selectable: C# 2.0 – 5.0; VB 8.0 – 11.0
View code documentation	Yes (inline XML or formatted)	No
View dependencies (text list)	Yes	Only references used
View dependencies (graph)	Yes [Assembly Visualizer]	No
View ancestry	Yes [Assembly Visualizer]	No
View type hierarchy	Yes [Assembly Visualizer]	No
Extract resources from an assembly	Yes	No
Extend Reflector	Yes	No
The Cheese Course		
Browse assemblies	multiple user-defined assembly lists	project references only
Exercise assemblies	Yes [SmokeTest]	No
Execute code snippets	Yes [Snippy]	No
Compare assemblies	Yes [Diff]	No
Meta-query assemblies	Yes [PowerCommands]	No

Reflector (Desktop) Main Window

Reflector operation revolves around the *assembly browser*. When you load assemblies they appear collapsed in this tree-structured navigation panel. Drill down to an item of interest (the table above enumerates item categories); selecting it in the assembly browser automatically updates the other panels labeled in white: *signature*, *decompilation*, and *documentation* panels. The dependencies panel also acts on the current item but only on-demand (via the **Analyze** command) and accumulates each request. Each of the panels on the right side may be opened or closed as needed, maximizing working space.



Reflector Editions

Features	Standard	VS	VS Pro
Desktop (to view code)	✓	✓	✓
VS plug-in (to view code)		✓	✓
VS plug-in (step into code & set breakpoints)			✓