**Ildasm.exe (MSIL Disassembler)**

**.NET Framework 4.5**

The MSIL Disassembler is a companion tool to the MSIL Assembler (Ilasm.exe). Ildasm.exe takes a portable executable (PE) file that contains Microsoft intermediate language (MSIL) code and creates a text file suitable as input to Ilasm.exe.

This tool is automatically installed with Visual Studio and with the Windows SDK. To run the tool, we recommend that you use the Visual Studio Command Prompt or the Windows SDK Command Prompt (CMD Shell). These utilities enable you to run the tool easily, without navigating to the installation folder. For more information, see [Visual Studio and Windows SDK Command Prompts](http://msdn.microsoft.com/en-us/library/ms229859.aspx).

* If you have Visual Studio installed on your computer: On the taskbar, click **Start**, click **All Programs**, click **Visual Studio**, click **Visual Studio Tools**, and then click **Visual Studio Command Prompt**.

-or-

If you have the Windows SDK installed on your computer: On the taskbar, click **Start**, click **All Programs**, click the folder for the Windows SDK, and then click **Command Prompt** (or **CMD Shell**).

* At the command prompt, type the following:

ildasm [options] [PEfilename] [options]

[**Parameters**](javascript:void(0))

The following options are available for .exe, .dll, .obj, .lib, and .winmd files.

|  |  |
| --- | --- |
| Option | Description |
| **/out=***filename* | Creates an output file with the specified *filename*, rather than displaying the results in a graphical user interface. |
| **/rtf** | Produces output in rich text format. Invalid with the **/text** option. |
| **/text** | Displays the results to the console window, rather than in a graphical user interface or as an output file. |
| **/html** | Produces output in HTML format. Valid with the **/output** option only. |
| **/?** | Displays the command syntax and options for the tool. |

The following additional options are available for .exe, .dll, and .winmd files.

|  |  |
| --- | --- |
| Option | Description |
| **/bytes** | Shows actual bytes, in hexadecimal format, as instruction comments. |
| **/caverbal** | Produces custom attribute blobs in verbal form. The default is binary form. |
| **/linenum** | Includes references to original source lines. |
| **/nobar** | Suppresses the disassembly progress indicator pop-up window. |
| **/noca** | Suppresses the output of custom attributes. |
| **/project** | Displays metadata the way it appears to managed code, instead of the way it appears in the native Windows Runtime. If *PEfilename* is not a Windows metadata (.winmd) file, this option has no effect. See [.NET Framework Support for Metro Style Apps and Windows Runtime](http://msdn.microsoft.com/en-us/library/hh694558.aspx). |
| **/pubonly** | Disassembles only public types and members. Equivalent to **/visibility:PUB**. |
| **/quoteallnames** | Includes all names in single quotes. |
| **/raweh** | Shows exception handling clauses in raw form. |
| **/source** | Shows original source lines as comments. |
| **/tokens** | Shows metadata tokens of classes and members. |
| **/visibility:***vis*[+*vis*...] | Disassembles only types or members with the specified visibility. The following are valid values for *vis*:  **PUB** — Public  **PRI** — Private  **FAM** — Family  **ASM** — Assembly  **FAA** — Family and Assembly  **FOA** — Family or Assembly  **PSC** — Private Scope  For definitions of these visibility modifiers, see [MethodAttributes](http://msdn.microsoft.com/en-us/library/system.reflection.methodattributes.aspx) and [TypeAttributes](http://msdn.microsoft.com/en-us/library/system.reflection.typeattributes.aspx). |

The following options are valid for .exe, .dll, and .winmd files for file or console output only.

|  |  |
| --- | --- |
| Option | Description |
| **/all** | Specifies a combination of the **/header**, **/bytes**, **/stats**, **/classlist**, and **/tokens** options. |
| **/classlist** | Includes a list of classes defined in the module. |
| **/forward** | Uses forward class declaration. |
| **/headers** | Includes file header information in the output. |
| **/item:***class*[**::***member*[**(***sig*]] | Disassembles the following depending upon the argument supplied:   * Disassembles the specified *class*. * Disassembles the specified *member* of the *class*. * Disassembles the *member*of the *class* with the specified signature *sig*. The format of *sig* is:   [**instance**] *returnType*(*parameterType1*, *parameterType2*, …, *parameterTypeN*)  **Note** In the .NET Framework versions 1.0 and 1.1, *sig* must be followed by a closing parenthesis: (*sig*). Starting with the Net Framework 2.0 the closing parenthesis must be omitted: (*sig*. |
| **/noil** | Suppresses MSIL assembly code output. |
| **/stats** | Includes statistics on the image. |
| **/typelist** | Produces the full list of types, to preserve type ordering in a round trip. |
| **/unicode** | Uses Unicode encoding for the output. |
| **/utf8** | Uses UTF-8 encoding for the output. ANSI is the default. |

The following options are valid for .exe, .dll, .obj, .lib, and .winmd files for file or console output only.

|  |  |
| --- | --- |
| Option | Description |
| **/metadata**[=*specifier*] | Shows metadata, where *specifier* is:  **MDHEADER** — Show the metadata header information and sizes.  **HEX** — Show information in hex as well as in words.  **CSV** — Show the record counts and heap sizes.  **UNREX** — Show unresolved externals.  **SCHEMA** — Show the metadata header and schema information.  **RAW** — Show the raw metadata tables.  **HEAPS** — Show the raw heaps.  **VALIDATE** — Validate the consistency of the metadata.  You can specify **/metadata** multiple times, with different values for *specifier*. |

The following options are valid for .lib files for file or console output only.

|  |  |
| --- | --- |
| Option | Description |
| **/objectfile**=*filename* | Shows the metadata of a single object file in the specified library. |
| **Description: NoteNote** | |
| All options for Ildasm.exe are case-insensitive and recognized by the first three letters. For example, **/quo** is equivalent to **/quoteallnames**. Options that specify arguments accept either a colon (:) or an equal sign (=) as the separator between the option and the argument. For example, **/output:***filename* is equivalent to **/output=***filename*. | |

[**Remarks**](javascript:void(0))

Ildasm.exe only operates on PE files on disk. It does not operate on files installed in the global assembly cache.

The text file produced by Ildasm.exe can be used as input to the MSIL Assembler (Ilasm.exe). This is useful, for example, when compiling code in a programming language that does not support all the runtime metadata attributes. After compiling the code and running its output through Ildasm.exe, the resulting MSIL text file can be hand-edited to add the missing attributes. You can then run this text file through the MSIL Assembler to produce a final executable file.

|  |
| --- |
| **Description: NoteNote** |
| Currently, you cannot use this technique with PE files that contain embedded native code (for example, PE files produced by Visual C++). |

You can use the default GUI in the MSIL Disassembler to view the metadata and disassembled code of any existing PE file in a hierarchical tree view. To use the GUI, type **ildasm** at the command line without supplying the *PEfilename* argument or any options. From the **File** menu, you can navigate to the PE file that you want to load into Ildasm.exe. To save the metadata and disassembled code displayed for the selected PE, select the **Dump** command from the **File** menu. To save the hierarchical tree view only, select the **Dump Treeview** command from the **File** menu. For a detailed guide to loading a file into Ildasm.exe and interpreting the output, see the Ildasm.exe Tutorial, located in the Samples folder that ships with the Windows Software Development Kit (SDK).

If you provide Ildasm.exe with a *PEfilename* argument that contains embedded resources, the tool produces multiple output files: a text file that contains MSIL code and, for each embedded managed resource, a .resources file produced using the resource's name from metadata. If an unmanaged resource is embedded in *PEfilename*, a .res file is produced using the filename specified for MSIL output by the **/output** option*.*

|  |
| --- |
| **Description: NoteNote** |
| Ildasm.exe shows only metadata descriptions for .obj and .lib input files. MSIL code for these file types is not disassembled. |

You can run Ildasm.exe over an.exe or .dll file to determine whether the file is managed. If the file is not managed, the tool displays a message stating that the file has no valid common language runtime header and cannot be disassembled. If the file is managed, the tool runs successfully.

**Version Information**

Starting with the .NET Framework 4.5, Ildasm.exe handles an unrecognized marshal BLOB (binary large object) by displaying the raw binary content. For example, the following code shows how a marshal BLOB generated by a C# program is displayed:

// C#

public void Test([MarshalAs((short)70)] int test) { }

// MSIL from Ildasm.exe output

.method public hidebysig instance void

Test(int32 marshal({ 46 }) test) cil managed

Starting with the .NET Framework 4.5, Ildasm.exe displays attributes that are applied to interface implementations, as shown in the following excerpt from Ildasm.exe output:

.class public auto ansi beforefieldinit MyClass

extends [mscorlib]System.Object

implements IMyInterface

{

.interfaceimpl type IMyInterface

.custom instance void

[mscorlib]System.Diagnostics.DebuggerNonUserCodeAttribute::.ctor() = ( 01 00 00 00 )

…

[**Examples**](javascript:void(0))

The following command causes the metadata and disassembled code for the PE file MyHello.exe to display in the Ildasm.exe default GUI.

ildasm myHello.exe

The following command disassembles the file MyFile.exe and stores the resulting MSIL Assembler text in the file MyFile.il.

ildasm MyFile.exe /output:MyFile.il

The following command disassembles the file MyFile.exe and displays the resulting MSIL Assembler text to the console window.

ildasm MyFile.exe /text

If the file MyApp.exe contains embedded managed and unmanaged resources, the following command produces four files: MyApp.il, MyApp.res, Icons.resources, and Message.resources:

ildasm MyApp.exe /output:MyApp.il

The following command disassembles the method MyMethod within the class MyClass in MyFile.exe and displays the output to the console window.

ildasm /item:MyClass::MyMethod MyFile.exe /text

In the previous example, there could be several methods named MyMethod with different signatures. The following command disassembles the instance method MyMethod with the return type of **void** and the parameter types **int32** and **string**.

ildasm /item:"MyClass::MyMethod(instance void(int32,string)" MyFile.exe /text

|  |
| --- |
| **Description: NoteNote** |
| In the .NET Framework versions 1.0 and 1.1, the left parenthesis that follows the method name must be balanced by a right parenthesis after the signature: MyMethod(instance void(int32)). Starting with the .NET Framework 2.0 the closing parenthesis must be omitted: MyMethod(instance void(int32). |

To retrieve a **static** method (**Shared** method in Visual Basic), omit the keyword **instance**. Class types that are not primitive types like **int32** and **string** must include the namespace and must be preceded by the keyword **class**. External types must be preceded by the library name in square brackets. The following command disassembles a static method named MyMethod that has one parameter of type [AppDomain](http://msdn.microsoft.com/en-us/library/system.appdomain.aspx) and has a return type of [AppDomain](http://msdn.microsoft.com/en-us/library/system.appdomain.aspx).

ildasm /item:"MyClass::MyMethod(class [mscorlib]System.AppDomain(class [mscorlib]System.AppDomain)" MyFile.exe /text

A nested type must be preceded by its containing class, delimited by a forward slash. For example, if the MyNamespace.MyClass class contains a nested class named NestedClass, the nested class is identified as follows: class MyNamespace.MyClass/NestedClass.