**How to: View Assembly Contents**

**.NET Framework 4.5**

You can use the [Ildasm.exe (MSIL Disassembler)](http://msdn.microsoft.com/en-us/library/f7dy01k1.aspx) to view Microsoft intermediate language (MSIL) information in a file. If the file being examined is an assembly, this information can include the assembly's attributes, as well as references to other modules and assemblies. This information can be helpful in determining whether a file is an assembly or part of an assembly, and whether the file has references to other modules or assemblies.

**To display the contents of an assembly using Ildasm.exe**

* Type **ildasm** <*assembly name*> at the command prompt. For example, the following command disassembles the Hello.exe assembly.

[Copy](javascript:CodeSnippet_CopyCode('CodeSnippetContainerCode_dd38d80f-1719-4e84-9c2e-42d915a4b6a3');" \o "Copy to clipboard.)

ildasm Hello.exe

**To view assembly manifest information**

* Double-click the MANIFEST icon in the MSIL Disassembler window.

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

The following example starts with a basic "Hello, World" program. After compiling the program, use Ildasm.exe to disassemble the Hello.exe assembly and view the assembly manifest.

**C#**

[**C++**](javascript:%20CodeSnippet_SetLanguage('CodeSnippetContainerLang',%20'Programming',%20'Visual%20C++');)

[**VB**](javascript:%20CodeSnippet_SetLanguage('CodeSnippetContainerLang',%20'Programming',%20'Visual%20Basic');)

[Copy](javascript:CodeSnippet_CopyCode('CodeSnippetContainerCode_8f530159-68e2-44a5-9c81-10478d299eda');)

using System;

class MainApp

{

public static void Main()

{

Console.WriteLine("Hello World using C#!");

}

}

Running the command ildasm.exe on the Hello.exe assembly and double-clicking the MANIFEST icon in the IL DASM window produces the following output:

[Copy](javascript:CodeSnippet_CopyCode('CodeSnippetContainerCode_fc4bb157-6fd2-4c7b-b148-f81ebb210d1b');)

// Metadata version: v4.0.30319

.assembly extern mscorlib

{

.publickeytoken = (B7 7A 5C 56 19 34 E0 89 ) // .z\V.4..

.ver 4:0:0:0

}

.assembly Hello

{

.custom instance void [mscorlib]System.Runtime.CompilerServices.CompilationRelaxationsAttribute::.ctor(int32) = ( 01 00 08 00 00 00 00 00 )

.custom instance void [mscorlib]System.Runtime.CompilerServices.RuntimeCompatibilityAttribute::.ctor() = ( 01 00 01 00 54 02 16 57 72 61 70 4E 6F 6E 45 78 // ....T..WrapNonEx

63 65 70 74 69 6F 6E 54 68 72 6F 77 73 01 ) // ceptionThrows.

.hash algorithm 0x00008004

.ver 0:0:0:0

}

.module Hello.exe

// MVID: {7C2770DB-1594-438D-BAE5-98764C39CCCA}

.imagebase 0x00400000

.file alignment 0x00000200

.stackreserve 0x00100000

.subsystem 0x0003 // WINDOWS\_CUI

.corflags 0x00000001 // ILONLY

// Image base: 0x00600000

The following table describes each directive in the assembly manifest of the Hello.exe assembly used in the example.

|  |  |
| --- | --- |
| Directive | Description |
| **.assembly extern <***assembly name***>** | Specifies another assembly that contains items referenced by the current module (in this example, mscorlib). |
| **.publickeytoken <***token***>** | Specifies the token of the actual key of the referenced assembly. |
| **.ver <***version number***>** | Specifies the version number of the referenced assembly. |
| **.assembly <***assembly name***>** | Specifies the assembly name. |
| **.hash algorithm <***int32 value***>** | Specifies the hash algorithm used. |
| **.ver <***version number***>** | Specifies the version number of the assembly. |
| **.module <***file name***>** | Specifies the name of the modules that make up the assembly. In this example, the assembly consists of only one file. |
| **.subsystem <***value***>** | Specifies the application environment required for the program. In this example, the value 3 indicates that this executable is run from a console. |
| **.corflags** | Currently a reserved field in the metadata. |

An assembly manifest can contain a number of different directives, depending on the contents of the assembly. For an extensive list of the directives in the assembly manifest, see the ECMA documentation, especially "Partition II: Metadata Definition and Semantics" and "Partition III: CIL Instruction Set". The documentation is available online; see [ECMA C# and Common Language Infrastructure Standards](http://go.microsoft.com/fwlink/?LinkID=99212) on MSDN and [Standard ECMA-335 - Common Language Infrastructure (CLI)](http://go.microsoft.com/fwlink/?LinkID=65552) on the Ecma International Web site.