In the past, a software component (.exe or .dll) that was written in one language could not easily use a software component that was written in another language. COM provided a step towards solving this problem. The .NET Framework makes component interoperation even easier by allowing compilers to emit additional declarative information into all modules and assemblies. This information, called metadata, helps components to interact seamlessly.

Metadata is binary information describing your program that is stored either in a common language runtime portable executable (PE) file or in memory. When you compile your code into a PE file, metadata is inserted into one portion of the file, and your code is converted to Microsoft intermediate language (MSIL) and inserted into another portion of the file. Every type and member that is defined and referenced in a module or assembly is described within metadata. When code is executed, the runtime loads metadata into memory and references it to discover information about your code's classes, members, inheritance, and so on.

Metadata describes every type and member defined in your code in a language-neutral manner. Metadata stores the following information:

* Description of the assembly.
  + Identity (name, version, culture, public key).
  + The types that are exported.
  + Other assemblies that this assembly depends on.
  + Security permissions needed to run.
* Description of types.
  + Name, visibility, base class, and interfaces implemented.
  + Members (methods, fields, properties, events, nested types).
* Attributes.
  + Additional descriptive elements that modify types and members.

[**Benefits of Metadata**](javascript:void(0))

Metadata is the key to a simpler programming model, and eliminates the need for Interface Definition Language (IDL) files, header files, or any external method of component reference. Metadata enables .NET Framework languages to describe themselves automatically in a language-neutral manner, unseen by both the developer and the user. Additionally, metadata is extensible through the use of attributes. Metadata provides the following major benefits:

* Self-describing files.

Common language runtime modules and assemblies are self-describing. A module's metadata contains everything needed to interact with another module. Metadata automatically provides the functionality of IDL in COM, so you can use one file for both definition and implementation. Runtime modules and assemblies do not even require registration with the operating system. As a result, the descriptions used by the runtime always reflect the actual code in your compiled file, which increases application reliability.

* Language interoperability and easier component-based design.

Metadata provides all the information required about compiled code for you to inherit a class from a PE file written in a different language. You can create an instance of any class written in any managed language (any language that targets the common language runtime) without worrying about explicit marshaling or using custom interoperability code.

* Attributes.

The .NET Framework lets you declare specific kinds of metadata, called attributes, in your compiled file. Attributes can be found throughout the .NET Framework and are used to control in more detail how your program behaves at run time. Additionally, you can emit your own custom metadata into .NET Framework files through user-defined custom attributes.

[**Metadata and the PE File Structure**](javascript:void(0))

Metadata is stored in one section of a .NET Framework portable executable (PE) file, while Microsoft intermediate language (MSIL) is stored in another section of the PE file. The metadata portion of the file contains a series of table and heap data structures. The MSIL portion contains MSIL and metadata tokens that reference the metadata portion of the PE file. You might encounter metadata tokens when you use tools such as the MSIL Disassembler (Ildasm.exe) to view your code's MSIL, for example.