Top of Form

**Intermediate Language (IL)**

**Definition - What does *Intermediate Language (IL)* mean?**

Intermediate language (IL) is an object-oriented programming language designed to be used by compilers for the .NET Framework before static or dynamic compilation to machine code. The IL is used by the .NET Framework to generate machine-independent code as the output of compilation of the source code written in any .NET programming language.  
  
IL is a stack-based assembly language that gets converted to bytecode during execution of a virtual machine. It is defined by the common language infrastructure (CLI) specification. As IL is used for automatic generation of compiled code, there is no need to learn its syntax.  
  
This term is also known as Microsoft intermediate language (MSIL) or common intermediate language (CIL).

**Techopedia explains *Intermediate Language (IL)***

With the help of a suitable just-in-time (JIT) compiler, IL code can be executed on any computer architecture supported by the JIT compiler. Unlike interpreters, JIT compilation provides better performance, preserves memory, and saves time during application initialization. IL enables the platform- and CPU-independence feature of the .NET framework, by allowing compiled source code to be executed in any environment supporting the CLI specification.  
  
Verification of code safety, for IL code, provides better security and reliability than natively-compiled executable files. The metadata, describing the MSIL code in the portable executable, eliminates the need for type libraries and intermediate definition language files as was used in the Component Object Model (COM) technology. Combined with metadata and a common type system, IL forms the means to integrate modules written in different languages into one single application, thus enabling language independence.  
  
Although IL is similar to Java bytecode in its usage by compilers, it differs from the latter in that it is designed for platform independence and language independence. It also differs in that it is compiled and not interpreted.  
  
Two types of instruction sets are included with IL; base instructions, similar to native CPU instructions, and Object model instructions used by the high-level language. IL includes all instructions necessary for loading, storing, initializing, and calling methods on objects. It also includes arithmetic and logical operations, control flow, direct memory access, exception handling and other operations. Unlike the common object file format used for executable content in the traditional Microsoft portable executable, the portable executable generated, after the compilation of managed code, contains both IL instructions and metadata.  
  
The two tools associated with IL code are the MSIL Assembler (Ilasm.exe) and the MSIL Disassemble (Ildasm.exe). The former generates a portable executable file from IL code and permits viewing the IL code in human-readable format, while the latter converts a portable executable file back to a text file, for viewing and modification. Both are automatically installed as part of Visual Studio.

Bottom of Form