

.NET Intermediate Language (IL)

Stack Engine, IL Fundamentals, Metadata, IL Syntax, Opcode Injection, Building Compilers, Reflection

Intermediate Language (IL) is .NET's low-level platform-independent representation of an executable. Many .NET developers are content to write high-level code in an IDE and then compile/run it, oblivious to IL. More advanced developers and those with specialist needs are more ambitious – they wish to program directly in IL, to browse and edit the IL generated for them by high-level language compilers, to auto-generate source code from other logical representations, to create compilers, and to really know “under the hood” how code runs when using high level languages (to help optimize performance, aid more precise debugging, etc.)

This course covers all aspects of IL, including the opcodes, metadata, assembly syntax, compilation/de-compilation tools, binary file format and .NET's reflection (which provides classes to browse existing assemblies and to emit assemblies directly). We also examine usage scenarios, such as building your own compiler and code generation tools.

Attending this course will allow you get a jumpstart on understanding all aspects of .NET IL, to produce a variety of code manipulation functionality and gain a much better appreciation of how .NET code executes.

Contents of One-Day Training Course		
<p>Target Audience This course will interest advanced .NET developers who wish to code directly in IL, or who need a richer understanding of how their higher-level code executes, or who need to create code generators and specialist developer tools.</p> <p>Prerequisites In-depth knowledge of C# 8 and all round experience using the .NET CLR</p> <p>Experience of language design, compiler creation and low-level code manipulation useful</p>	<p>Review of CLR Issues Assemblies & modules, how code executes, security issues, type loader CLR architecture from IL viewpoint Stack-based execution engine</p> <p>IL Fundamentals Overall IL Model Verbose/compact IL JIT compiler Hello world in IL</p> <p>IL Tools Ilasm.exe, Ildasm.exe Ngen.exe, PEVerify.exe</p> <p>Introduction to Structure of IL PE/COFF headers and sections Metadata tables Manifest Managed code representations</p> <p>Metadata Fundamentals Set of tables with very detailed data about contained code; Table types and uses</p> <p>Advanced Metadata Important tables (ModuleDef, TypeDef, MethodDef, FieldDef, AssemblyRef, ModuleRef, ClassLayout, NestedClass</p> <p>Types, Fields and Methods The IL instruction set Use of IL language constructs How code from high-level .NET languages appears in IL</p> <p>Advanced Types Signatures, visibility, inheritance, ctors Primitive/native/managed types</p>	<p>Other IL Features Unmanaged code Exception handling/events/delegates</p> <p>Programming with IL Writing more complex programs in IL Coding issues to be aware of Object interactions in IL</p> <p>Profiler API The unmanaged Profiler API allows you to add custom code that will be called when the CLR is about to JIT IL code You can change the IL on-the-fly</p> <p>Code Interactions Coverage of why and how one might wish to programmatically interact with code Overview of required code services</p> <p>Reflection System.Reflection namespace Dynamically loading & invoking types Browsing contents of assemblies</p> <p>Emitting System.Reflection.Emit.*-Builder classes Emitting persistent & transient assemblies</p> <p>.NET Native and IL .NET Native – concepts and toolchain Converting from IL to native code</p> <p>Building Custom IL Tools Coverage of why and how to programmatically interact with IL code Overview of required code services</p> <p>Project How to integrate IL modules in your own custom project</p>