# .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.

### 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.

### Prerequisites

In-depth knowledge of C# and all round experience using the .NET CLR

Experience of language design, compiler creation and low-level code manipulation useful

### *Contents of One-Day Training Course*

#### Review of CLR Issues

Assemblies & modules, how code executes, security issues, type loader
CLR architecture from IL viewpoint
Stack-based execution engine

#### IL Fundamentals

Overall IL Model
Verbose/compact IL
JIT compiler
Hello world in IL

#### IL Tools

Ilasm.exe, Ildasm.exe
Ngen.exe, PEVerify.exe

#### Introduction to Structure of IL

PE/COFF headers and sections
Metadata tables
Manifest
Managed code representations

#### Metadata Fundamentals

Set of tables with very detailed data about contained code; Table types and uses

#### Advanced Metadata

Important tables (ModuleDef, TypeDef, MethodDef, FieldDef, AssemblyRef, ModuleRef, ClassLayout, NestedClass

#### Types, Fields and Methods

The IL instruction set
Use of IL language constructs
How code from high-level .NET languages appears in IL

#### Advanced Types

Signatures, visibility, inheritance, ctors
Primitive/native/managed types

#### Other IL Features

Unmanaged code
Exception handling/events/delegates

#### Programming with IL

Writing more complex programs in IL
Coding issues to be aware of
Object interactions in IL

#### 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

#### Code Interactions

Coverage of why and how one might wish to programmatically interact with code
Overview of required code services

#### Reflection

System.Reflection namespace
Dynamically loading & invoking types
Browsing contents of assemblies

#### Emitting

System.Reflection.Emit.*-Builder classes
Emitting persistent & transient assemblies

#### .NET Native and IL

.NET Native – concepts and toolchain
Converting from IL to native code

#### Building Custom IL Tools

Coverage of why and how to programmatically interact with IL code
Overview of required code services

#### Project

How to integrate IL modules in your own custom project