COM Interoperability

Objectives

- On completion of this Session you will be able to
  - Distinguish between managed code and unmanaged code
  - Use Pointers in C# code in unsafe blocks
  - Invoke Win32 APIs using PInvoke service
  - Invoke ActiveX dlls in .NET client application
  - Invoke .NET component in COM client application

Managed Vs Unmanaged Code

- Managed Code
    - Targeted to CLR
    - Memory managed by Garbage collector
    - MSIL code
    - .NET compliant
- Unmanaged Code
    - Code not targeted to CLR
    - Memory not managed by Garbage collector
    - Operating System compliant
- CLR provides two mechanisms for interoperation with unmanaged code.
    - Platform Invocation Services
    - COM Interoperability

Unmanaged code – unsafe block

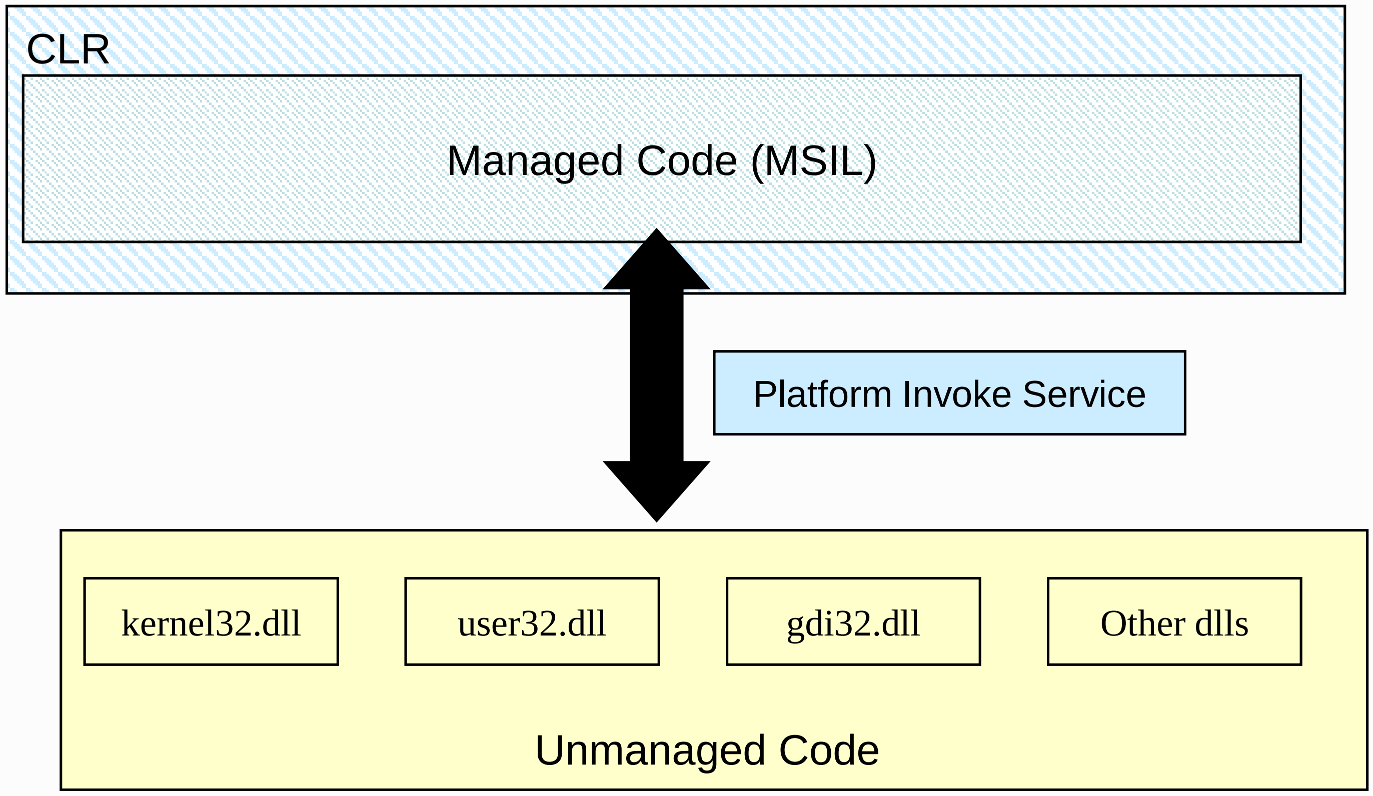
- Pointer are used in unsafe block.
- Unsafe block
  - Can be whole class, struct, interface, delegate or any of members of these .

```
public unsafe struct Node
{
  public int Value;
  public long* Left;
  public long* Top;
}
```

```
unsafe static void Main()
   {   Point point;
       Point* p = &point;
       p->x = 10;
       p->y=30;
}
```

Platform Invoke Service

- Used to invoke unmanaged functions (Win32 API) from dlls.



How `PInvoke` works ?

- Locates the dll containing the required Windows API function.
- Loads the dll in the memory.
- Locates the address of the function in the memory and pushes the arguments on stack.
- Transfers the control to unmanaged function for execution.
- Returns the exceptions from these functions to be handled by managed code.

Using PInvoke

```
using System.Runtime.InteropServices;
public class TestWin32
{
[DllImport("user32.dll", CharSet=CharSet.Auto,
  EntryPoint=“MessageBoxA”),CharSet.Ansi, ExactSpelling
  = false,CallingConvention=CallingConvention.StdCall]
public static extern int MsgBox(int
  hWnd, string text, string caption, uint type);
```

```
private void button3_Click(object sender, EventArgs e)
{
TestWin.MsgBox(0, "Hello World","Platform Invoke
                 Sample", 0);
}
```

Interoperability

- Reusing COM components into the .NET application irrespective of the technology in which it is developed.

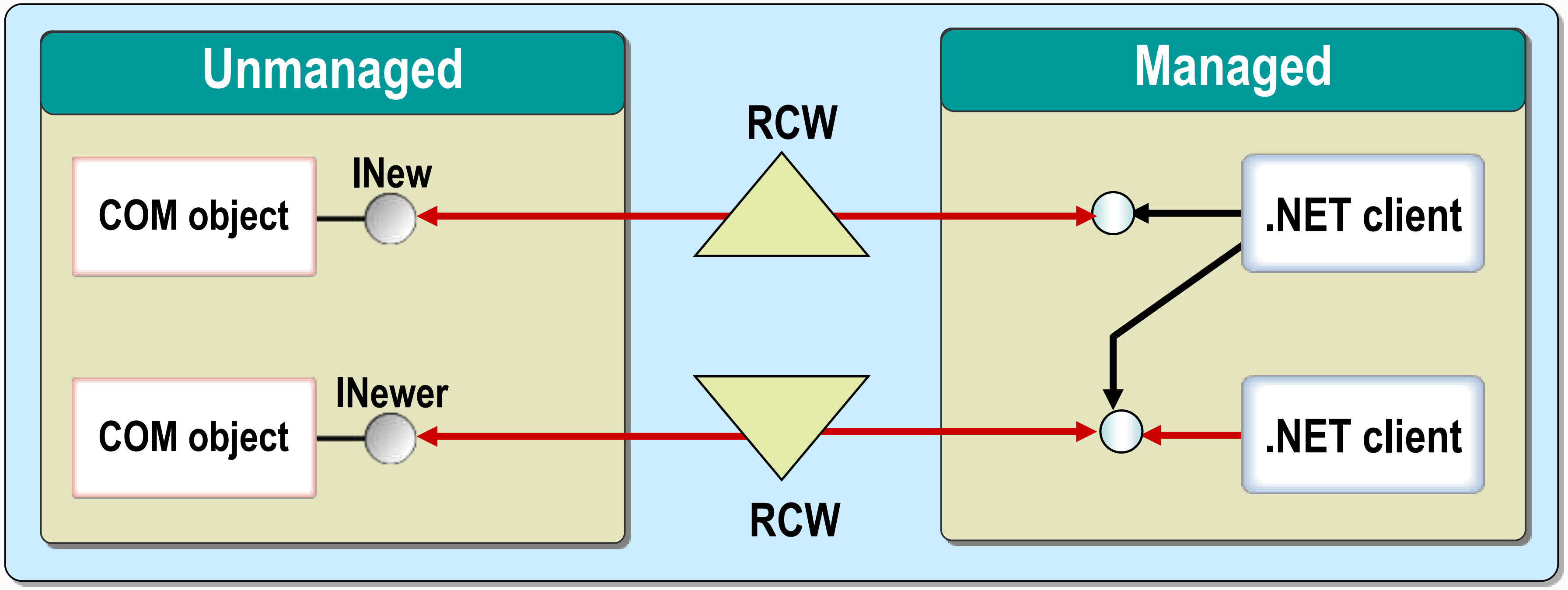
- Ensures
  - Installation of the .NET framework does not affect existing COM applications.
  - COM components and .NET applications can communicate with each other.

COM component vs. .NET Components

Component	COM	.NET
Coding model	Interface	Object
Identifier	GUID	Strong name
Compatibility	Binary	Type
Type definition	Type library	Metadata
Versioning	No	Yes

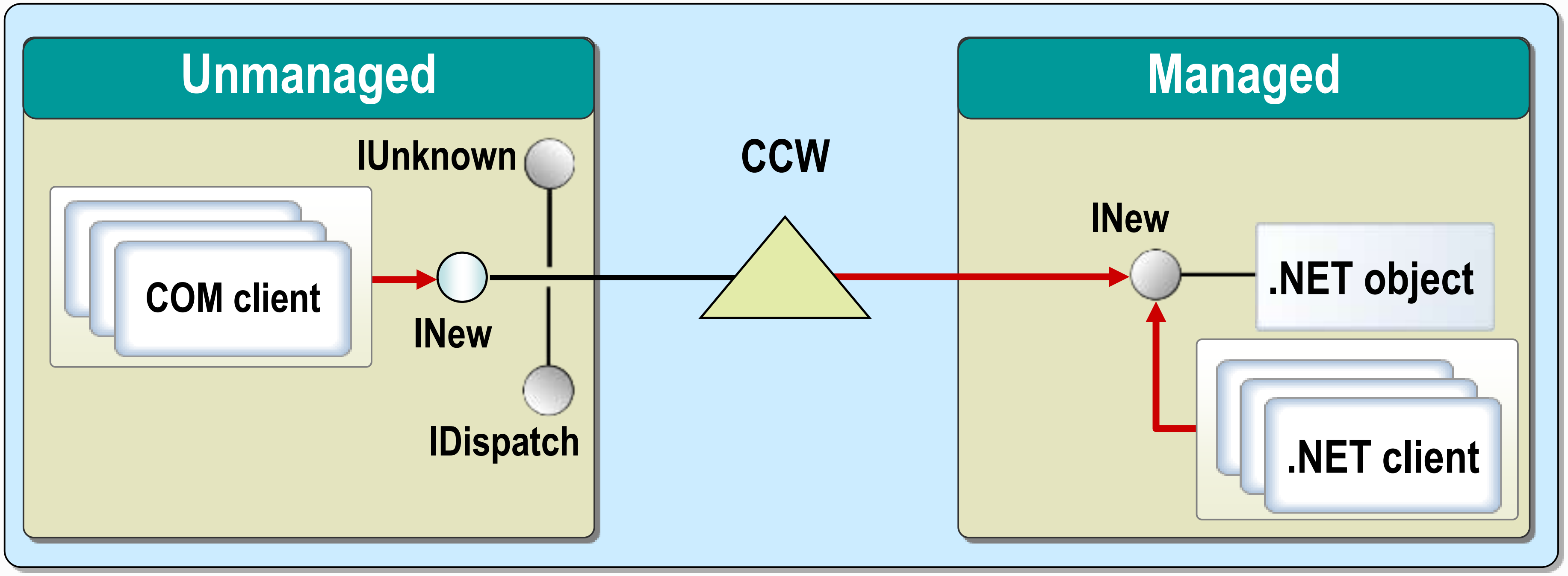
Using COM component in .NET environment

- Ensure that required COM components is registered with windows registry.
- Add Reference of COM component in .NET application
- Use component by importing namespace as usual



Using .NET component from COM client

- Set the attribute in AssemblyInfo.cs file [assembly: ComVisible(true)].
- Add a reference of .tlb (.NET component).
- Use .tlb components like activeX dll.



Quick Recap…

- Managed code targets the services of CLR.
- Unmanaged code is platform dependent code and does not target the services of CLR.
- Pointers could be used in C# program inside the unsafe block.
- Win32 APIs could be used in .NET application using PInvoke service.
- RCW enables the COM component to be used in .NET application.
- CCW enables the .NET component to be used in COM client application.