# .NET BASICS

Table of Contents

[.NET BASICS 1](#_Toc171438009)

[.NET Program Execution 2](#_Toc171438010)

[ILDASM and ILASM 2](#_Toc171438011)

[Compilation: 2](#_Toc171438012)

[Components of Assembly: 2](#_Toc171438013)

[Manifest Metadata: 2](#_Toc171438014)

[Modifying Assembly Version: 2](#_Toc171438015)

[Peeking Inside an Assembly with ILDASM: 3](#_Toc171438016)

[Dumping an Assembly with ILDASM: 3](#_Toc171438017)

[Rebuilding an Assembly with ILASM Short Notes 3](#_Toc171438018)

[.NET Assemblies Classification 3](#_Toc171438019)

[Assembly Name Components 3](#_Toc171438020)

[Assembly Version in .NET 3](#_Toc171438021)

[Version Number Components 3](#_Toc171438022)

[Using Wildcards 4](#_Toc171438023)

[Strong Naming an Assembly in .NET 4](#_Toc171438024)

[Generating the Key File 4](#_Toc171438025)

[Using AssemblyKeyFile Attribute 4](#_Toc171438026)

[Requirements for Strongly Named Assembly 4](#_Toc171438027)

[Weak Named Assemblies 4](#_Toc171438028)

[Strong Named Assemblies 4](#_Toc171438029)

[**GAC**: Global Assembly Cache 5](#_Toc171438030)

[Installing into GAC: 5](#_Toc171438031)

[Considerations: 5](#_Toc171438032)

[GAC in .NET 4.0: 5](#_Toc171438033)

[Installing an Assembly into GAC: 5](#_Toc171438034)

[Uninstalling from GAC using Gacutil: 5](#_Toc171438035)

[Key Points to Remember about .NET Assembly Loading 6](#_Toc171438036)

[Notes on DLL Hell and .NET Versioning 6](#_Toc171438037)

## .NET Program Execution

## ILDASM and ILASM

Compilation: Entire project source code → Intermediate Language (IL) → Packaged into assembly.

### Components of Assembly:

* Manifest: Contains metadata.
* Intermediate Language (IL): Compiled code.

### Manifest Metadata:

* Assembly name.
* Version.
* Culture.
* Strong name information.
* Reference Assembly Details

### Modifying Assembly Version:

* Properties Folder: Expand it in Solution Explorer.
* AssemblyInfo.cs: Open this file under the Properties folder.
* AssemblyVersion Attribute:
* Default: 1.0.0.0
* Change to: 2.0.0.0

### Peeking Inside an Assembly with ILDASM:

* Open Visual Studio Command Prompt: Path: Start -> All Programs -> Microsoft Visual Studio 2010 -> Visual Studio Tools.
* Right-click "Visual Studio Command Prompt 2012" -> Select "Run as Administrator".
* Command: In the command prompt, type: Ildasm.exe C:\YourDirectoryPath\YourAssembly.exe
* View Assembly:
  + Manifest and types (classes, structs, etc.) are displayed.
  + Assembly version is visible at the bottom.

### Dumping an Assembly with ILDASM:

1. **File Menu**: Select from ILDASM tool.
2. **Dump**: Select and view "Dump Options Window".
3. **Click OK**: On "Dump Options Window".
4. **File Name**: Enter "sample" and save to C: drive.
5. **Check**: Navigate to C: drive in Windows Explorer to see Sample.il

### Rebuilding an Assembly with ILASM Short Notes

1. **Command Prompt**:
   * Open "Visual Studio Command Prompt".
   * Type: ILASM.exe C:\Sample.il
   * Press Enter.
2. **Check Output**:
   * Navigate to C: drive in Windows Explorer.
   * You should see Sample.exe.

### .NET Assemblies Classification

1. **Weak Named Assemblies**
2. **Strong Named Assemblies**

### Assembly Name Components

1. **Simple Textual Name**
2. **Version Number**
3. **Culture Information** (If provided, otherwise language neutral)
4. **Public Key Token**

### Assembly Version in .NET

* **AssemblyVersion Attribute**: Used to specify the assembly version.
* **Default Version**: 1.0.0.0

### Version Number Components

1. **Major Version**
2. **Minor Version**
3. **Build Number**
4. **Revision Number**

### Using Wildcards

* **Syntax**: [assembly: AssemblyVersion("2.1.\*")]
* **Purpose**: Defaults the Revision and Build Numbers using \*

### Strong Naming an Assembly in .NET

* **AssemblyKeyFile Attribute**: Used to sign the assembly with a strong name.
* **Constructor Argument**: Path of the key file containing private and public keys.

### Generating the Key File

1. Open Visual Studio Command Prompt.
2. Execute the command: sn.exe -k c:\KeyFile.snk.
   1. **Key File Location**: C:\KeyFile.snk.
   * **SN.exe**: Stands for Strong Name.
   * **Key File Extension**: .snk.

### Using AssemblyKeyFile Attribute

* In AssemblyInfo.cs file:

[assembly: AssemblyKeyFile("KeyFile.snk")]

* **Build the Project**: This process will strongly name the assembly.

### Requirements for Strongly Named Assembly

1. **Textual Assembly Name**
2. **Assembly Version Number**
3. **Signed with Private/Public Key Pair**

### Weak Named Assemblies

* **Not Signed** with private/public key pair.
* **Not Guaranteed Unique**: May cause DLL hell.
* **Cannot be Installed into GAC**: Only strongly named assemblies can be installed.

### Strong Named Assemblies

* **Guaranteed Unique**
* **Solves DLL Hell**
* **Required for GAC Installation**

## **GAC**: Global Assembly Cache

* Stores strongly named assemblies.
* Allows assemblies to be shared among multiple applications on the same machine.
* Avoids the need to copy assemblies locally for each application.

### Installing into GAC:

* When: Install an assembly into the GAC only when it needs to be shared by multiple applications on the same machine.
* How: Use tools like gacutil.exe or Windows Explorer to install assemblies into the GAC.

### Considerations:

* Deployment: If you plan to deploy your application using XCopy deployment (copying files directly without installation), do not install assemblies into the GAC. This is because GAC contents are not automatically copied during XCopy deployment.
* Private vs Shared: Keep assemblies private (local to the application) unless they need to be shared globally.
* Versioning: Ensure proper versioning of assemblies in the GAC to avoid conflicts and ensure compatibility across applications.

### GAC in .NET 4.0:

* Two GACs: Introduced with .NET 4.0.
* 1st GAC Path: C:\Windows\Assembly
  + Contains assemblies for .NET 2.0 to 3.5.
* 2nd GAC Path: C:\WINDOWS\Microsoft.NET\assembly
  + Stores assemblies specifically for .NET 4.0 and later versions.

### Installing an Assembly into GAC:

* Drag and Drop:
  + Method: Simply drag the assembly file (\*.dll) into the appropriate GAC folder (C:\Windows\Assembly or C:\WINDOWS\Microsoft.NET\assembly).
  + Usage: Quick and straightforward for manual installation, suitable for simple deployments.
* GacUtil.exe (GAC Utility Tool):
  + Method: Command-line tool (gacutil.exe) provided by .NET Framework SDK.
  + Usage: Execute gacutil.exe /i assembly\_name.dll to install the assembly into the GAC, allowing finer control and scripting for deployment processes.

### Uninstalling from GAC using Gacutil:

* Command: gacutil -u AssemblyName
  + Removes all versions of AssemblyName from the GAC.
  + Specify Version:
    - Command: gacutil -u AssemblyName,Version=1.0.0.0,PublicKeyToken=eeaabf36d7783129
    - Removes a specific version of AssemblyName identified by its version and public key token.
* Notes:
  + Spaces: Ensure no spaces between commas and keywords (Version, PublicKeyToken).
  + File Extension: Do not include .dll or .exe in the assembly name when uninstalling to avoid errors.
  + Multiple Versions: Uninstalling without specifying version removes all versions; specifying version removes only that specific version.

### Key Points to Remember about .NET Assembly Loading

1. Version Determination:
   1. The application's assembly manifest contains information about dependent assemblies.
   2. The Common Language Runtime (CLR) checks the following for version override information:
      1. Application configuration file.
      2. Publisher policy file (if exists).
      3. Machine configuration file.
2. GAC (Global Assembly Cache) Search: .NET searches the GAC only if the assembly is strongly named.
3. Search Order:
   1. If the assembly is not found in the GAC
      1. If there is a .config file, .NET searches the location specified in the configuration file.
      2. If there is no .config file, .NET searches the directory containing the executable (.EXE).
4. Failure Handling:
   1. If the assembly is not found, the application terminates with an error.

#### Note: Version checking is not done for Weakly Named Assemblies (Assemblies without a strong name).

### Notes on DLL Hell and .NET Versioning

1. **Applications Installed**:

* **A1**
* **A2**

1. **Shared Assembly**:

* shared.dll

1. **New Version of A2**:

* Available online
* Downloaded and installed

1. **Installation Effects**:

* shared.dll overwritten
* **A2** works fine
* **A1** fails due to backward incompatibility of new shared.dll

1. **DLL Hell**:

* Problem where a new version of a shared component (DLL) is installed
* New version is not backward compatible
* Causes other applications relying on the older version to break

1. **Solution in .NET**:

* .NET versioning eliminates DLL Hell
* Applications can specify which version of a DLL they depend on