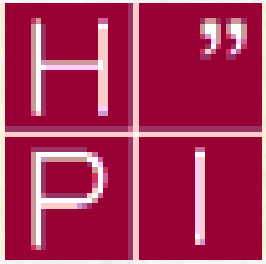


# Packaging & Deploying in Microsoft .NET

Wolfgang Schult

OSM – Operating Systems and Middleware

Hasso-Plattner-Institute Potsdam, Germany

# Packaging and Deploying

- Why is packaging and deployment so important?
  - Traditional deployment under windows lacks in several points
  - “DLL Hell”
  - Installation affects the hole system (Registry, File System etc.) – Isn’t easy to uninstall
  - Security issues

# Building Types into Modules

- What is a .Net (Portable) Executable (PE)?
- Two types:
  - Console user interface (CUI)
  - Graphical user interface (GUI)
- Managed PE file has four main parts:
  - PE Header
  - CLR Header
  - The metadata
  - The intermediate language (IL)

# CLR Header

```
typedef struct IMAGE_COR20_HEADER
{
    ULONG                   cb;
    USHORT                  MajorRuntimeVersion;
    USHORT                  MinorRuntimeVersion;

    IMAGE_DATA_DIRECTORY    MetaData;
    ULONG                   Flags;
    ULONG                   EntryPointToken;

    IMAGE_DATA_DIRECTORY    Resources;
    IMAGE_DATA_DIRECTORY    StrongNameSignature;

    ...
} IMAGE_COR20_HEADER;
```

*Header versioning*

*Symbol table and startup information*

*Binding information*

# Definition Metadata Tables

| | |
|---|---|
| ModuleDef | Filename (w/o path), version ID (GUID) |
| TypeDef | name, base type, flags,containing members |
| MethodDef | Name, flags, signature, offset to IL-Code |
| FieldDef | Name, flags |
| ParamDef | Name, flags (in, out, retval) |
| PropertyDef | Name, flags, type, backing field |
| EventDef | Name, flags |

# Reference Metadata Types

| | |
|---|---|
| AssemblyRef | One entry for each assembly referenced ...<br>Name, version, culture, public key, flags, hash |
| ModuleRef | One entry for each PE-module referenced ...<br>Filename (w/o path) |
| TypeRef | One entry for each type referenced ...<br>Name, Reference to..<br>TypeRef, ModuleDef, ModuleRef, AssemblyRef |
| MemberRef | One entry for each member referenced...<br>Name, signature, points to TypeRef member |

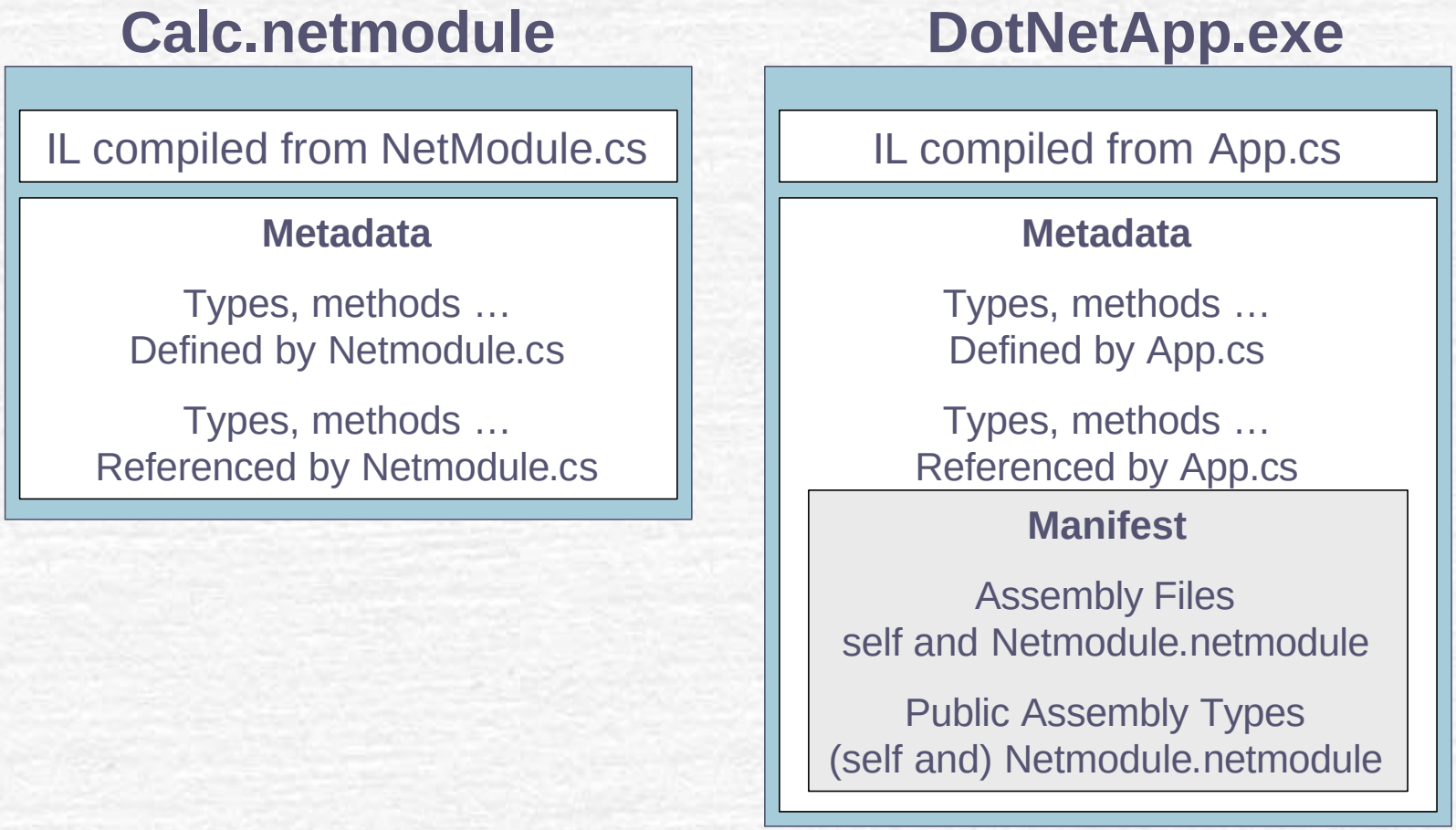
# What is an assembly?

- An assembly is a logical grouping of one or more managed modules or resource files
- An assembly is the smallest unit of reuse, security, and versioning
- One of the assembly's files is chosen to hold a *manifest* :
  - Metadata table containing name of all files
  - Assembly's version, culture, publisher, publicly exported types

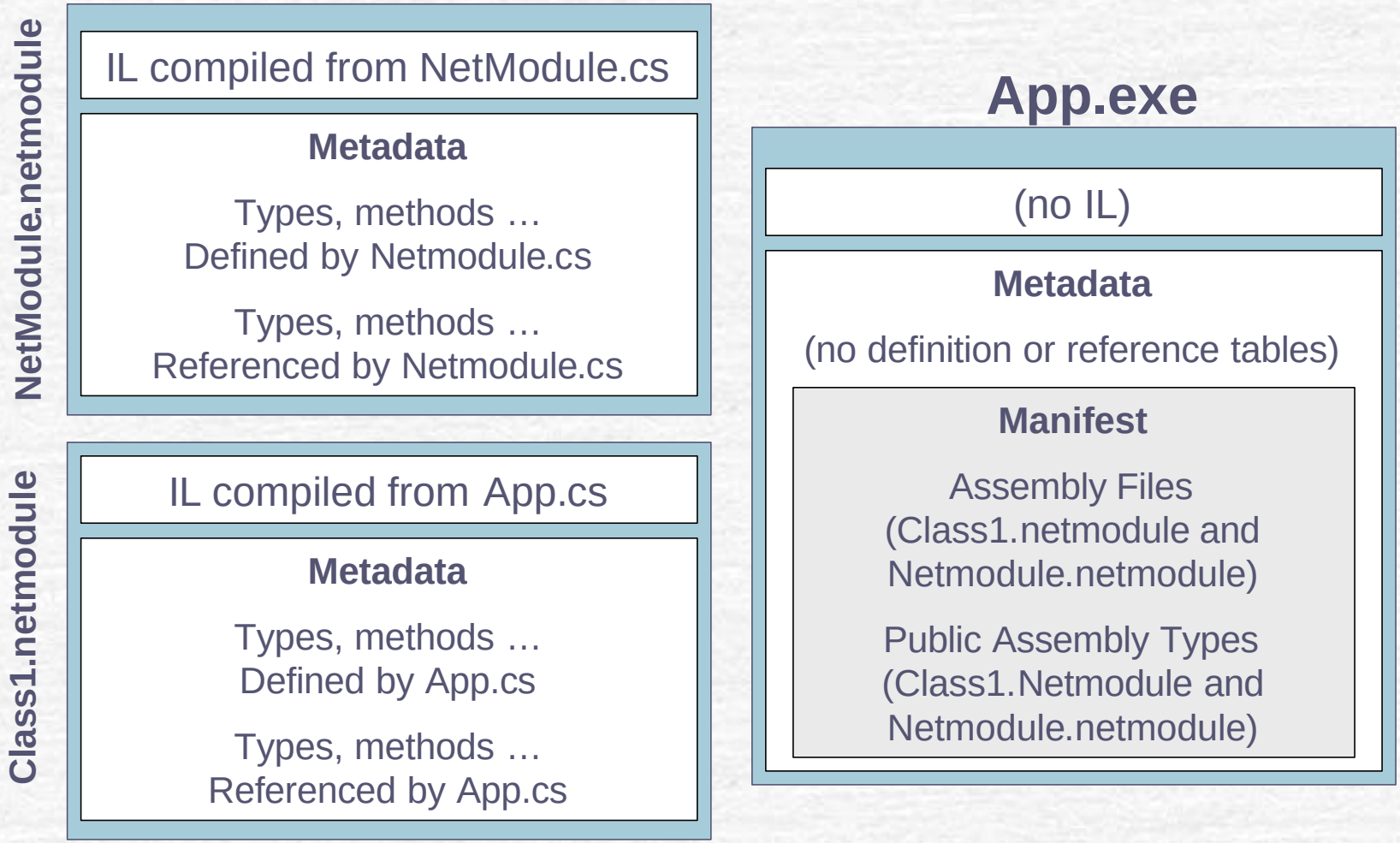
# Manifest Metadata Tables

| | |
|---|---|
| AssemblyDef | Assembly‘s name, version (Major, Minor, Build, Revision), culture, flags, hash algorithm, publisher‘s public key (can be null) |
| FileDef | One entry for each PE & resource file ...<br>Filename, hash, flags |
| Manifest-ResourceDef | One Entry for each resource ...<br>Name, flags, FileDef, stream-offset (optional) |
| Exported-TypesDef | One entry for each exported type...<br>Type’s name, FileDef, TypeDef |

# A Multifile Assembly



# A Multifile Assembly



# Assembly Resource Information

**AssemblyInfo.cs**

```
[assembly: AssemblyTitle("My Assembly")]
[assembly: AssemblyDescription("Cool Stuff")]
[assembly: AssemblyCompany("Hasso Plattner Institute")]
[assembly: AssemblyProduct("Hello App")]
[assembly: AssemblyCopyright(
  "(C) 2003 by Wolfgang Schult")]
[assembly: AssemblyTrademark("My Trademarks")]
[assembly: AssemblyCulture("")]
[assembly: AssemblyConfiguration("")]
[assembly: AssemblyVersion("1.0.*")]
```

# Version Numbers

| Part | Major Number | Minor Number | Build Number | Revision Number |
|---|---|---|---|---|
| Example: | 2 | 5 | 719 | 2 |
| | „public perception“ | | | |

Version: 2.5.719.2

- AssemblyFileVersion (Informational)
- AssemblyInformationalVersion (Informational)
- AssemblyVersion (stored in AssemblyDef)

# Culture

- Described in RFC1766

| Primary Tag | Secondary Tag | Culture |
|---|---|---|
| de | (none) | German |
| de | AT | Austrian German |
| De | CH | Swiss German |
| en | (none) | English |
| en | GB | British English |

# Culture specific Resource

- Create an assembly that contains the code and application‘s (default) or fallback resources
- Create one or more separate assemblies that contain only culture specific resources – no code at all.
- Assembly marked with a culture are called *satellite assemblies*.
- Use al.exe /c[ulture]:*Text* to build such assembly

# Simple Application Deployment

- Simple xcopy to users directory
- Assemblies include all information about dependant assembly and types -> the runtime will look for referenced assemblies
- No modification to the registry or active directory are necessary
- Uninstall: just delete all the files
- Packaging with .cab-files or Windows Installer (.msi)
- Assemblies deployed to the same directory as the application are called *privately deployed assemblies*

# Simple Administrative Control

- Config files in the application's directory
- Users and/or administrators can change could create/modify this file
- CLR interprets content of this file to alter its policies for locating and loading assembly files
- Contain XML and can be associated with an application or with the machine
- Examples :
  - Web.config
  - DotNetApp.exe.config
  - Machine.config

# Shared Assemblies

- Versioning problem is difficult to address and to solve
- Files will ship with bugs and developers will always want to add new features
- But there is no guarantee that new files don‘t break an application
- Easy way to restore applications „last-known good state“

# Weakly vs. Strongly Named Assemblies

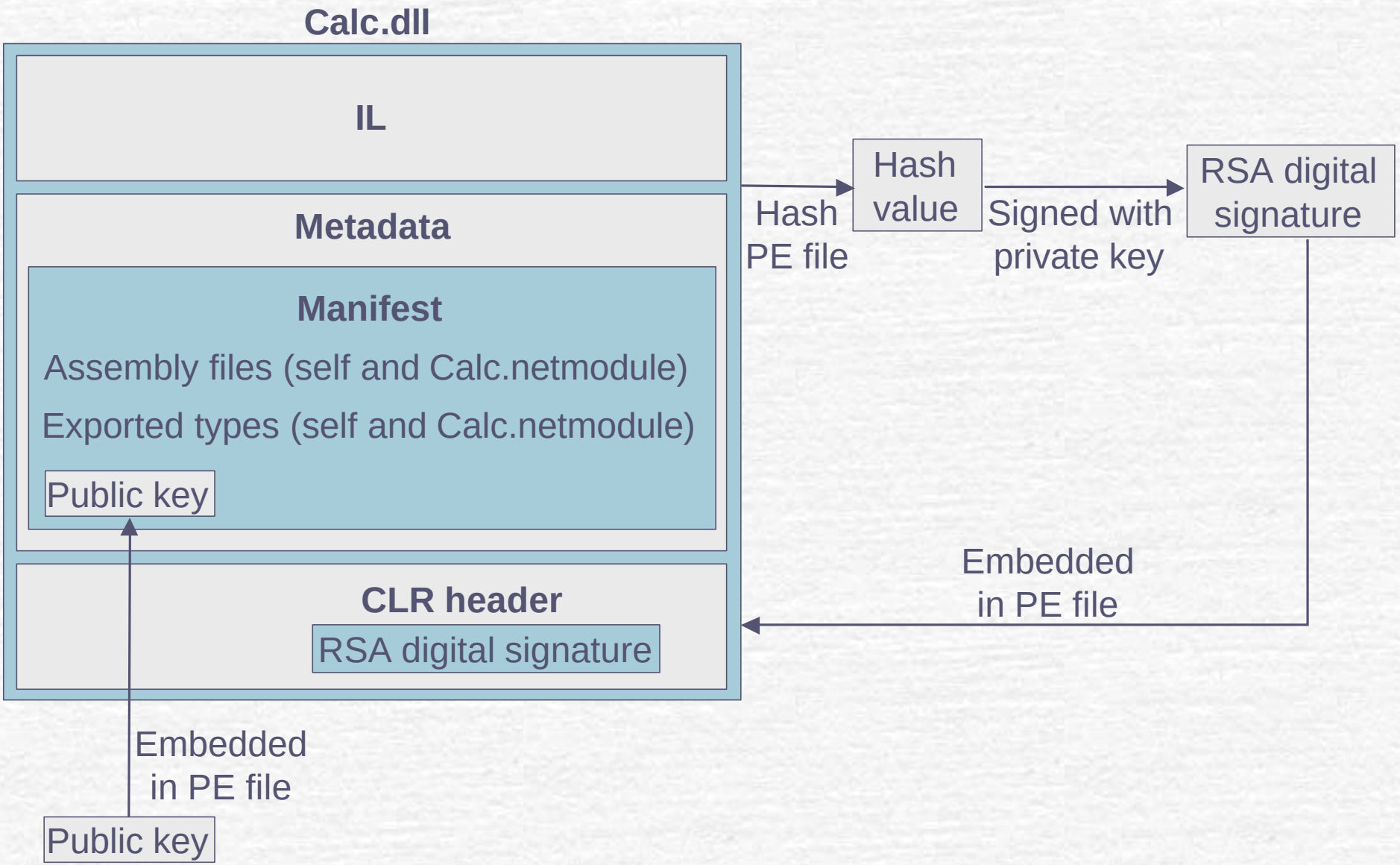
| Kind of Assembly | Can Be Privately Deployed? | Can Be Globally Deployed? |
|---|---|---|
| Weakly Named | YES | NO |
| Strong Named | YES | YES |

# Giving an Assembly a Strong Name

- Problem if two (or more) companies produce assemblies that have the same filename and both get copied into the same well known directory
- Assemblies filename is not good enough
- CLR uses four attributes
  - A filename
  - A version number
  - A culture identification
  - A public key token

# Example

- MyAssembly, Version=1.0.2302, Culture=neutral, PublicKeyToken=BAAD23021976A9D4
- MyAssembly, Version=1.0.2302, Culture=de-DE, PublicKeyToken=BAAD23021976A9D4
- MyAssembly, Version=2.0.1234, Culture=neutral, PublicKeyToken=BAAD23021976A9D4
- MyAssembly, Version=1.0.2302, Culture=neutral, PublicKeyToken=BA23A7D8F8228FFD


Calc.dll
IL
Metadata
Manifest
Assembly files (self and Calc.netmodule)
Exported types (self and Calc.netmodule)
Public key
CLR header
RSA digital signature
Hash
PE file
Hash value
Signed with
private key
RSA digital signature
Embedded
in PE file
Embedded
in PE file
Public key

# The Global Assembly Cache

- The Global Assembly Cache (GAC) is a well known directory for all .Net applications
- Structured directory with many subdirectories (auto generated)
- Never manually copy a file into GAC; use GACUtil.exe instead
- By default you need Administrator rights to use GACUtil
- GACUtil  is not part of the end user framework! (you must use windows installer instead)

# Reference a Strongly Named Assembly

- With CSC.exe's /reference switch
- If the filename is a full path, CSC loads the specified file and uses its metadata information
- Otherwise it looks:
  1. The working directory
  2. The directory that contains the CLR that the compiler itself is using
  3. Any directories specified in the /lib switch
  4. Any directories specified by the LIB environment variable

# Tamper-Resistance

- Signing with a private key ensures that the holder of the public key produced the assembly
- When assembly is installed into the GAC, the system compares the hashes
- When an application needs to bind an (strong name) assembly:
  1. looks in the GAC
  2. Looks in Application's base directory
  3. Any of private path in config file
  4. If installed with MSI, ask MSI
  5. Throw FileNotFoundException

Compare Hashes

# Delayed Signing (cont.)

- While developing and testing your assembly, gaining access to the secure private key can be a hassle
- .NET supports *delayed signing* sometimes referred to as *partial signing*
- Private key is not necessary but you lose all the tampering protection -> digital signature won‘t be embedded

# Delayed Signing

1. Add following attributes to source code:

```
[assembly: AssemblyDelaySign(true)]
[assembly: AssemblyKeyFile("MyPublicKey.key")]
```

2. Register assembly for verification skipping

```
Sn –Vr MyAssembly.dll
```

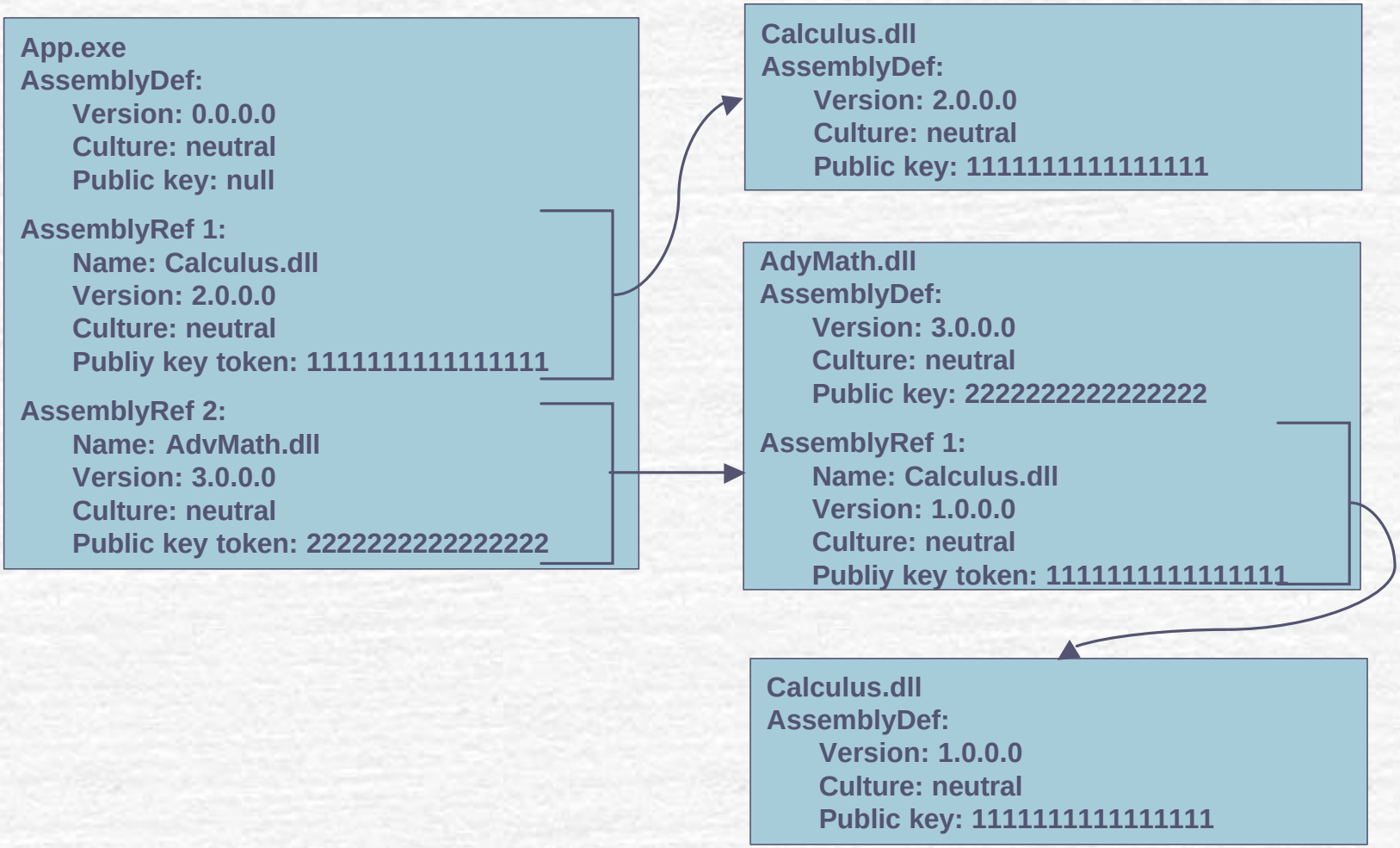
3. When ready to package and deploy

```
Sn –R MyAssembly.dll MyPrivateKey.keys
```

4. Turn verification back on

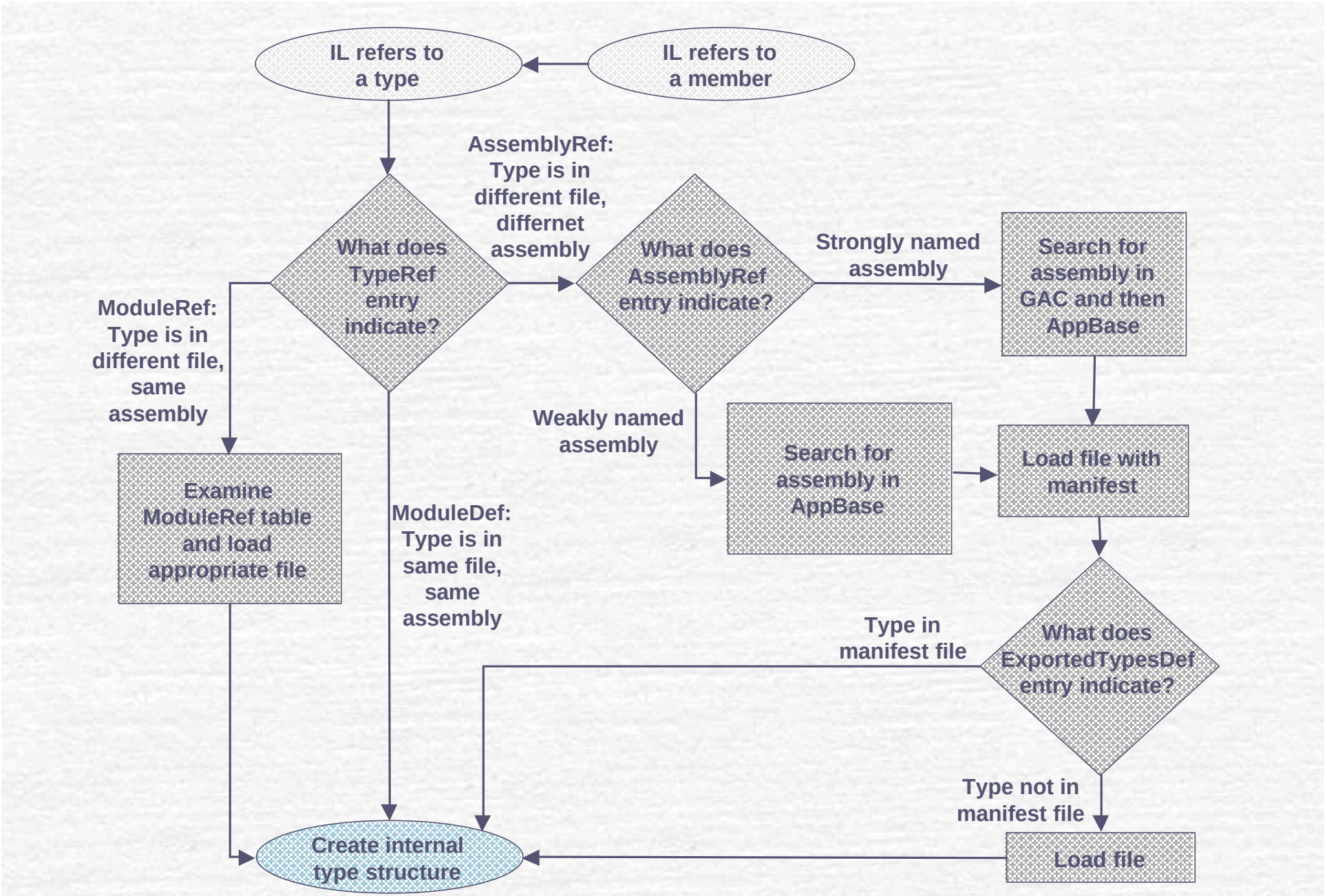
```
Sn –Vu MyAssembly.dll
```

# Side by Side Execution



# Resolving Type References

- When resolving a reference type, the CLR can find the type in one of the three places:
  - Same file
  - Different file, same assembly
  - Different file, different assembly


IL refers to a type
IL refers to a member
What does TypeRef entry indicate?
AssemblyRef: Type is in different file, differnet assembly
What does AssemblyRef entry indicate?
Strongly named assembly
Search for assembly in GAC and then AppBase
ModuleRef: Type is in different file, same assembly
Examine ModuleRef table and load appropriate file
ModuleDef: Type is in same file, same assembly
Weakly named assembly
Search for assembly in AppBase
Load file with manifest
Type in manifest file
What does ExportedTypesDef entry indicate?
Type not in manifest file
Load file
Create internal type structure

# Advanced Administratitive Control

- Several Elements in the config file describe how to locate an assembly

```
<assemblyBinding xmlns="urn:schemas-microsoft-
    com:asm.v1">
  <dependentAssembly>
    <assemblyIdentity name="Calc“
       publicKeyToken="3e7651c9da5f75dd" />
    <bindingRedirect oldVersion="1.0.0.0“
       newVersion="2.0.0.0" />
    <codeBase version="2.0.0.0“
      href="http:\\www.hpi.uni-potsdam.de/Calc" />
    <publisherPolicy apply="no" />
  </dependentAssembly>
</assemblyBinding>
```

# Publisher Policy Control

- Scenario: publisher sends a new version of the assembly to the administrator
- Administrator has to edit machine.config or app.config manually
- Automatic config with publisher policy control
- Configuration is a special assembly -> additionally shipped by the publisher

```
al /out:policy.1.0.calc.dll /version:1.0.0.0
   /keyfile:MyKeys.key /linkresource:calc.config
```