

**Module 9: Deploying  
Windows Forms  
Applications**

# Overview

- .NET Assemblies
- Deploying Windows Forms Applications

# Lesson: .NET Assemblies

- What is an Assembly?
- What Are Private Assemblies?
- What Are Strong-Named Assemblies?
- How to Build a Strong-Named Assembly
- How to Call a Strong-Named Assembly
- Demonstration: Viewing Assembly Metadata
- Practice: Calling a Strong-Named Assembly
- How to Install Assemblies into the Global Assembly Cache
- Demonstration: Using the .NET Framework Configuration Tool to Work with the Global Assembly Cache
- Practice: Working with the Global Assembly Cache

# What is an Assembly?

- A functional unit of sharing, versioning, and identity in the .NET Framework
- A unit for which permissions are requested and granted
- Can be shared across .NET applications

# What are Private Assemblies?

- Private assemblies

Private assemblies are deployed with and used exclusively by a single application

- Where private assemblies can reside

- Default probing process

Application folder tree

- Use **Assembly.LoadFrom** for these locations

Any folder on the local computer

Any folder on a remote computer

A URL

```
Dim PrivateAssembly As [Assembly]
PrivateAssembly = Assembly.LoadFrom("C:\PrivateAssembly.dll")
' Obtain a reference to a method known to exist in assembly.
Dim Method As MethodInfo = _
    PrivateAssembly.GetTypes(0).GetMethod("CalculateSum")
```

# What are Strong-Named Assemblies?

- Strong-named assemblies

Strong names identify assemblies uniquely and allow for features that guarantee the assembly is authentic and has not been tampered with

- Benefits of strong names

- Where strong-named assemblies can reside

- Where strong-named assemblies should reside

- Private assemblies vs. strong-named assemblies

# How to Build a Strong-Named Assembly

- 1 Use the Sn.exe utility to create a strong name key file

```
sn -k CalcKey.snk
```

- 2 Add attributes that describe the assembly  
Add to AssemblyInfo file

```
<Assembly: AssemblyKeyFile("CalcKey.snk")>  
<Assembly: AssemblyVersion("2.1.45.0")>
```

- 3 Build the project

# How to Call a Strong-Named Assembly

- 1** Add a reference to the strong-named assembly  
Browse to the correct location

- 2** Add Imports statement for the namespace of the strong-named assembly

- 3** Build the application

## Troubleshooting

If built using the same code as a private assembly, then you must add a reference and rebuild the calling application

# Demonstration: Viewing Assembly Metadata



In this demonstration, you will see how to

- Add external tools to the Visual Studio .NET environment
- Use ILDASM to view the metadata of an application that references a type that is built into a separate assembly

# Practice: Calling a Strong-Named Assembly



In this practice, you will

- Create a strong-named assembly
- Create an application that calls a strong-named assembly
- View the metadata for the application

Begin reviewing the objectives for  
this practice activity

15 min



# How to Install Assemblies into the Global Assembly Cache

Several tools available for installing assemblies in the global assembly cache

- During development and testing phases, use  
GACUtil.exe  
GACUtil.exe Options
  - i (install)
  - l (list)
  - u (uninstall)
- For deployment, use  
MSCORCFG.msc  
Windows Installer Setup Project

# Demonstration: Using the .NET Framework Configuration Tool to Work with the Global Assembly Cache



In this demonstration, you will see how to use the .NET Framework Configuration Tool (mscorcfg.msc) to install an assembly into the global assembly cache and how to examine the contents of the global assembly cache

# Practice: Working with the Global Assembly Cache



In this practice, you will

- Build a new version of an assembly
- Use GACutil.exe to install this assembly in the global assembly cache
- Create an application that references the assembly that you installed in the global assembly cache

Begin reviewing the objectives for  
this practice activity

15 min



# Lesson: Deploying Windows Forms Applications

- What Are Application Configuration Files?
- Elements of Application Configuration Files
- Element Attributes
- Practice: Creating and Using Application Configuration Files
- Other Configuration Files
- Demonstration: Tracing the Assembly Loading Process
- Packaging and Deploying .NET Applications
- Components of a Windows Installer Setup Project
- How to Create and Use a Windows Installer Setup Project
- Practice: Creating and Using a Windows Installer Deployment Project

# What are Application Configuration Files?

- Application configuration files provide a way of overriding the metadata in assemblies without having to rebuild the application
  - Name: *MyApp.exe.config*
  - Format: XML
  - Location: in the same folder as the application executable file
- Runtime searches for and examines (if found) application configuration files
- Application configuration files allow you to override default treatment regarding application, assembly and policies
- Application configuration files contain
  - Hierarchical elements
  - Some elements have attributes

# Elements of Application Configuration Files

```
<configuration>
  <runtime>
    <assemblyBinding>
      <dependentAssembly>
        <assemblyIdentity>
        <bindingRedirect>
        <codeBase>
        <publisherPolicy>
      </dependentAssembly>
    </assemblyBinding>
    <probing>
    <publisherPolicy>
  </runtime>
  . . .
</configuration>
```

# Element Attributes

- **<assemblyIdentity>**
  - name (required)
  - publicKeyToken (optional)
  - culture (optional)
- **<codeBase>**
  - version (required)
  - href (required)
- **<bindingRedirect>**
  - oldVersion (required)
  - newVersion (required)
- **<probing>**
  - privatePath (required)
- **<publisherPolicy>**
  - apply (required)

# Practice: Creating and Using Application Configuration Files



In this practice, you will

- Create a new version of the CalculatorEngine assembly
- Create an application configuration file for the WindowsCalculator application that points to the new version of the CalculatorEngine assembly and verify that, without rebuilding it, the application references the new version of the CalculatorEngine assembly

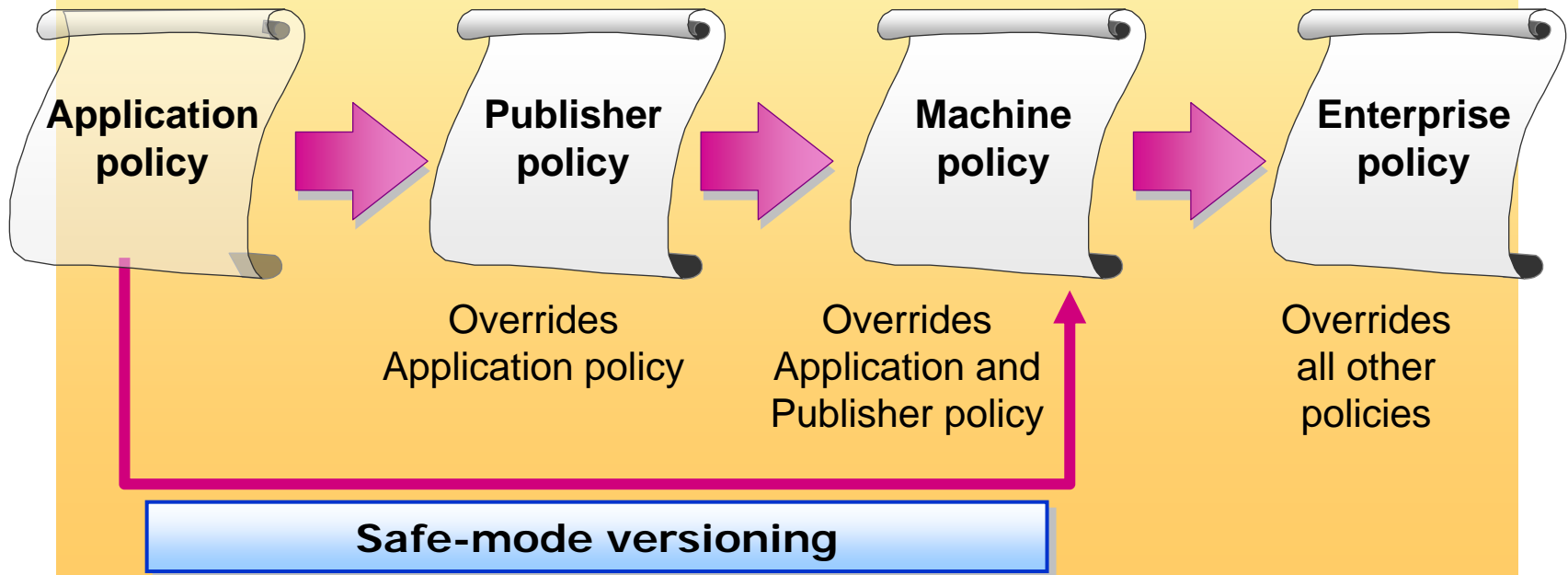
**Begin reviewing the objectives for  
this practice activity**

15 min



# Other Configuration Files

- Publisher policy
- Machine policy
- Enterprise policy



# Demonstration: Tracing the Assembly Loading Process



In this demonstration, you will see how to use the Fusion Log Viewer (FUSLOGVW) to trace the assembly loading process and solve probing errors

# Packaging and Deploying .NET Applications

## ■ Packaging Applications

- As a set of executables and DLLs
- Microsoft Windows Installer 2.0 package
- Cabinet files

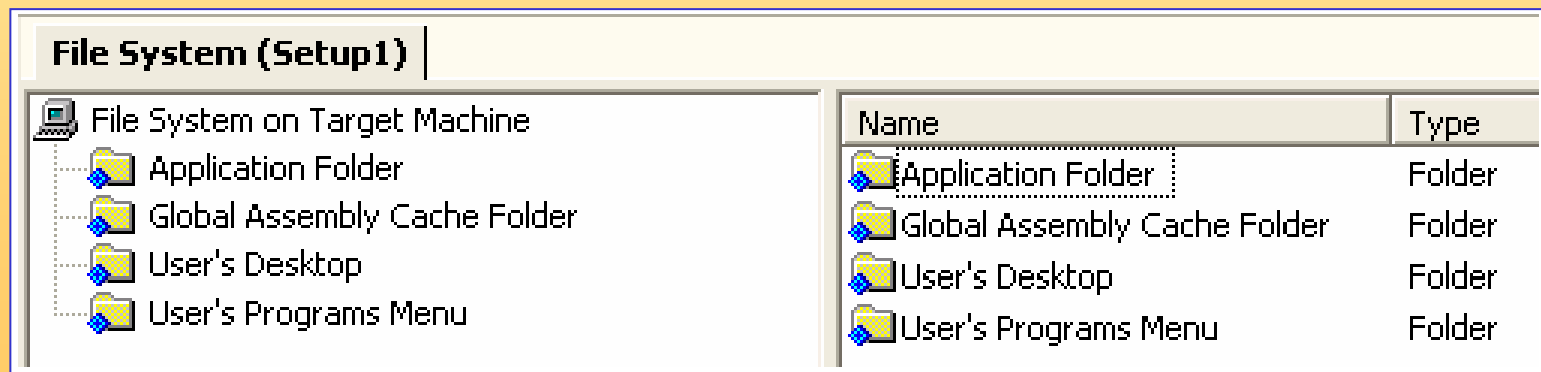
## ■ Deploying Applications

- XCOPY
- Windows Installer 2.0
- Code download

# Components of a Windows Installer Setup Project

## Setup projects for applications

- Application folder
- Global assembly cache folder
- User's desktop
- User's Program Menu



# How to Create and Use a Windows Installer Setup Project

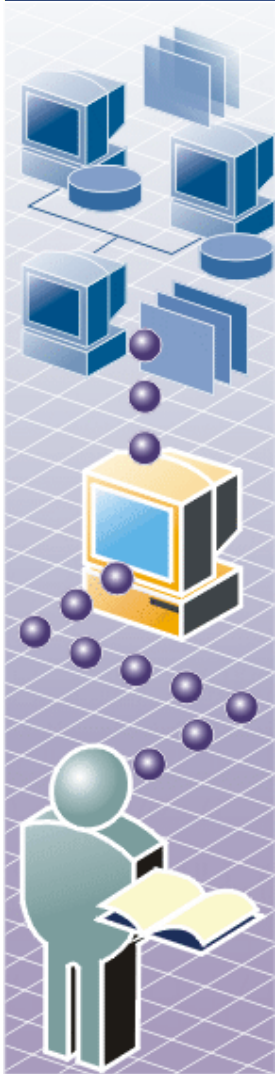
- 1** Create a new setup project in Visual Studio .NET
- 2** Set project properties as appropriate
- 3** Add the application files to be installed in the application folder  
For example, .exe, .dll, and locale files
- 4** Add icons for your application to the setup project
- 5** Add any shared assemblies to the Global Assembly Cache folder of the project
- 6** Build the project
- 7** In Explorer double-click the setup.exe file created by the setup project to install the application



# Review

- .NET Assemblies
- Deploying Windows Forms Applications

# Lab 9.1: Deploying an Application



- Exercise 1: Building and Referencing a Strong-Named Assembly
- Exercise 2: Installing a Strong-Named Assembly into the Global Assembly Cache
- Exercise 3: Deploying a .NET Application
- Exercise 4: Using an Application Configuration File