

1. **You create a Windows service that processes XML messages placed in a MSMQ queue. You discover that the service is not functioning properly. You need to debug the service to correct the program. What should you do?**

- A. Start the Windows service.
Then attach a debugger to the process.
- B. Attach a debugger to the Windows service.
Then start the Windows service.
- C. Start the Windows service.
Then run the .NET Services Installation tool (Regsvcs.exe).
- D. Place a breakpoint in the Main method of the Windows service.
Then run the application within the Visual Studio .NET integrated development environment (IDE).

Answer: A

2. **You create version 1.0.0.0 of an assembly named Assembly. You register the assembly in the global assembly cache.**

Assembly consists of two .NET Remoting objects named RemoteObject1 and RempoteObject2 These objects are configured in the App.config file of Assembly as shown in the following code segment:

```
<system.runtime.remoting>
  <application>
    <service>
      <activated type=" Assembly.RemoteObject1,
        Assembly, Version=1.0.0.0, Culture=neutral,
        PublicKeyToken=28dckd83491duj" />
      <wellKnown mode="SingleCall"
        objectUri="RemoteObject2.rem"
        type="Assembly.RemoteObject2, Assembly,
        Version=1.0.0.0, Culture=neutral,
        PublicKeyToken=28dckd83491duj" />
    <channels>
      <channel ref="http" />
    </channels>
  </service>
</application>
</system.runtime.remoting>
```

You create an application named App that resides on a different computer than Assembly. App references version 1.0.0.0 of Assembly. App contains code that activates instances of RemoteObject1 and RemoteObject2 to use their services.

Due to changes in business needs, you must update Assembly. You create version 2.0.0.0 of Assembly, which is backward compatible, but you do not update any information in

the App.config file of Assembly. You register version 2.0.0.0 of Assembly in the global assembly cache. You then rebuild App.

Which version of the remote object will App activate?

- A. Version 1.0.0.0 of RemoteObject1; version 1.0.0.0 of RemoteObject2.
- B. Version 1.0.0.0 of RemoteObject1; version 2.0.0.0 of RemoteObject2.
- C. Version 2.0.0.0 of RemoteObject1; version 1.0.0.0 of RemoteObject2.
- D. Version 2.0.0.0 of RemoteObject1; version 2.0.0.0 of RemoteObject2.

Answer: B

3. You have an ASP.NET application named WebApp. This application uses a private assembly named Employee to store and retrieve employee data. Employee is located in the bin directory of WebApp.

You develop a new ASP.NET application named WebApp2 that also needs to use Employee.

You assign Employee a strong name, set its version to 1.0.0.0, and install it in the global assembly cache. You then create a publisher policy assembly for version 1.0.0.0 and install it in the global assembly cache.

You compile WebApp2 against version 1.0.0.0. You do not recompile WebApp.

You then run WebApp.

What is the most likely result?

- A. A VersionNotFoundException is thrown.
- B. Employee is loaded from the bin directory.
- C. Version 1.0.0.0 of Employee is loaded from the global assembly cache.
- D. Version 1.0.0.0 of Employee is loaded by the publisher policy assembly.

Answer: D

4. You create a serviced component named SessionDispenser. This computer is in the Ser.Utilities assembly and is registered in a COM+ server application. SessionDispenser has multiple callers.

You discover that there are logic problems in the Create New Session method. You want to debug any calls to this method.

What should you do?

- A. Open the SessionDispenser solution.
Set a breakpoint on the CreateNewSession method.
Start the debugger.
- B. Attach the debugger to the client process.
Set a breakpoint on the SessionDispenser.CreateNewSession method.
- C. Attach the debugger to the Ser.Utilites.exe process.
Set a breakpoint on the CreateNewSession method.

- D. Attach the debugger to a Dllhost.exe process.
Set a breakpoint on the CreateNewSession method.

Answer: C