

Microsoft

Exam 70-513

**TS: Windows Communication Foundation Development with
Microsoft .NET Framework 4**

Version: 57.0

[Total Questions: 323]

Topic 1, C#**Question No : 1 - (Topic 1)**

You are developing a Windows Communication Foundation (WCF) service named CalculatorService, which implements the ICalculatorService contract. The service is configured to be discoverable through UDP. CalculatorService contains multiple endpoints. One of the endpoints is configured with the following behavior.

```
<behavior name="calculatorEndpointBehavior">
  <endpointDiscovery enabled="true">
    <extensions>
      <Information>
        ICalculatorService Endpoint.
      </Information>
      <Information>
        Udp Exposed Calculator Endpoint
      </Information>
    </extensions>
  </endpointDiscovery>
</behavior>
```

You need to log all the endpoint metadata information that is added by the service host.

Which code segment should you use?

- A.

```
var discoveryClient =
    new DiscoveryClient(new UdpDiscoveryEndpoint());
var findCriteria =
    new FindCriteria(typeof(ICalculatorService));
var findResponse = discoveryClient.Find(findCriteria);

foreach (var meta in findResponse.Endpoints)
{
    foreach (var xElement in meta.Extensions)
    {
        Log("Endpoint Information: "
            + xElement.Element("Information").Value);
    }
}
```
- B.

```
var discoveryClient =
    new DiscoveryClient(new UdpDiscoveryEndpoint());
var findCriteria = new FindCriteria();
var findResponse = discoveryClient.Find(findCriteria);
var meta = discoveryClient.Endpoint;

foreach (var xElement in meta.Contract.Operations) {
    Log("Endpoint Information: "
        + xElement.Behaviors.ToString());
}
```
- C.

```
var discoveryClient =
    new DiscoveryClient(new UdpDiscoveryEndpoint());
var findCriteria =
    new FindCriteria(typeof(ICalculatorService));
var findResponse = discoveryClient.Find(findCriteria);
var meta = findResponse.Endpoints[0];

foreach (var xElement in meta.Extensions)
{
    Log("Endpoint Information: "
        + xElement.Element("Information").Value);
}
```
- D.

```
var discoveryClient =
new DiscoveryClient(new UdpDiscoveryEndpoint());
var findCriteria =
new FindCriteria(typeof(ICalculatorService));
var findResponse = discoveryClient.Find(findCriteria);
foreach(var meta in findResponse.Endpoints)
{
    foreach(var xElement in meta.Extentions)
    {
        Log("Endpoint Information: "
            + xElement.Element("Information").Value);
    }
}
```

- A. Option A
 B. Option B
 C. Option C
 D. Option D

Answer: A

Question No : 2 - (Topic 1)

You have an existing Windows Communication Foundation (WCF) service.

You need to ensure that other services are notified when the service is started.

What should you do?

A. Add the following standard endpoint to the service.

```
<endpoint name="udpAnnouncementEndpoint"  
kind="udpDiscoveryEndpoint" />
```

B. Add the following standard endpoint to the service.

```
<endpoint name="udpDiscoveryEndpoint"  
kind="udpAnnouncementEndpoint" />
```

C. Add a service behavior with the following element.

```
<serviceDiscovery>  
<announcementEndpoints>  
<endpoint kind="udpDiscoveryEndpoint" />  
</announcementEndpoints>  
</serviceDiscovery>
```

D. Add a service behavior with the following element.

```
<serviceDiscovery>  
<announcementEndpoints>  
<endpoint kind="udpAnnouncementEndpoint" />  
</announcementEndpoints>  
</serviceDiscovery>
```

Answer: D

Question No : 3 - (Topic 1)

You develop a Windows Communication Foundation (WCF) service. You enable all performance counters and run multiple calls to the service.

The service must isolate session data for each user.

You need to monitor the instancing behavior used in the service.

Which performance counter should you monitor?

A. ServiceModelService 4.0.0.0\Calls

B. ServiceModelService 4.0.0.0\Instances

- C. ASP.NET State Service\State Server Sessions Active
- D. ASP.NET State Service\State Server Sessions Total

Answer: B

Question No : 4 - (Topic 1)

You develop a Windows Communication Foundation (WCF) service.

You name the service MovieService in the Movie namespace. The service is hosted in Microsoft Internet Information Services (IIS).

You copy the assembly containing the service to the bin folder in the virtual directory path.

You need to set up the URI that is mapped to the service.

What should you do?

A. Add the following code segment to the web.config file.

```
<serviceHostingEnvironment>  
<serviceActivations>  
odd relativeAddress="./Movie" service="Movie.MovieService"/>  
</serviceActivations>  
</serviceHostingEnvironment>
```

B. Add a Movie.svc file in the root of the virtual path with the following line.

```
<%&ServiceHost language="C#" Service="MovieService"*>
```

C. Add the following code segment to the web.config file.

```
<serviceHostingEnvironment>  
<serviceActivations>  
odd relativeAddress=" ./Movie, svc" service="Hovie.MovieService"/>  
</serviceActivations>  
</serviceHostingEnvirorunent>
```

D. Add a Movie.svc file in the root of the virtual path with the following line.

```
<%&ServiceHost language="C#" Service="MovieService.svc"%>
```

Answer: B

Topic 2, VB

Question No : 5 - (Topic 2)

A Windows Communication Foundation (WCF) service that handles corporate accounting must be changed to comply with government regulations of auditing and accountability.

You need to configure the WCF service to execute under the Windows logged-on identity of the calling application.

What should you do?

- A.** Within the service configuration, add a serviceAuthorization behavior to the service, and set impersonateCallerForAllOperations to true.
- B.** Within the service configuration, add a serviceAuthenticationManager behavior to the service, and set serviceAuthenticationManagerType to Impersonate.
- C.** Within the service configuration, add a serviceSecurityAudit behavior to the service, and set serviceAuthorizationAuditLevel to SuccessOrFailure.
- D.** Within the service configuration, add a serviceCredentials behavior to the service, and set type to Impersonate.

Answer: A

Question No : 6 - (Topic 2)

A Windows Communication Foundation (WCF) solution uses the following contract to share a message across its clients. (Line numbers are included for reference only.)

```
01 <ServiceContract(>  
02 Public Interface ITeamMessageService  
03  
04 <OperationContract(>  
05 Function GetMessage() As String  
06  
07 <OperationContract(>  
08 Sub PutMessage(ByVal message As String)  
09 End Interface
```

The code for the service class is as follows.

```
10 Public Class TeamMessageService
11 Implements ITeamMessageService
12
13 Dim key As Guid = Guid.NewGuid()
14 Dim message As String = "Today s Message"
15
16 Public Function GetMessage() As String _
17 Implements ITeamMessageService.GetMessage
18
19 Return String.Format("Message:{0}. Key:{1}", message, key)
20 End Function
21
22 Public Sub PutMessage(ByVal message As String) _
23 Implements ITeamMessageService.PutMessage
24
25 Me.message = message
26 End Sub
27
28 End Class
```

The service is self-hosted. The hosting code is as follows.

```
29 Dim host As ServiceHost =
New ServiceHost(GetType(TeamMessageService))
30 Dim binding As BasicHttpBinding =
New BasicHttpBinding(BasicHttpSecurityMode.None)
```

```
31 host.AddServiceEndpoint(  
"MyApplication.ITeamMessageService", binding,  
"http://localhost:12345")
```

```
32 host.Open()
```

You need to ensure that all clients calling GetMessage will retrieve the updated string if the message is updated by any client calling PutMessage.

What should you do?

- A.** Add the following attribute to the TeamMessageService class, before line 10.
<ServiceBehavior(InstanceContextMode:=InstanceContextMode.Single)>
- B.** Add the following attribute to the TeamMessageService class, before line 10002E
<ServiceBehavior(InstanceContextMode:=
InstanceContextMode.PerSession)>
- C.** Pass a service instance to the instancing code in line 29, as follows.
Dim host As ServiceHost = New ServiceHost(New TeamMessageService())
- D.** Redefine the message string in line 14, as follows.
Shared message As String = "Today s Message"
- E.** Then change the implementation of PutMessage in lines 22-26 to the following.
Public Sub PutMessage(ByVal message As String) _
Implements ITeamMessageService.PutMessage
TeamMessageService.message = message
End Sub

Answer: A

Question No : 7 - (Topic 1)

You are creating a Windows Communication Foundation (WCF) service application. The application needs to service many clients and requests simultaneously.

The application also needs to ensure subsequent individual client requests provide a stateful conversation.

You need to configure the service to support these requirements.

Which attribute should you add to the class that is implementing the service?

- A. [ServiceBehavior (InstanceContextMode = InstanceContextMode.PerSession, ConcurrencyMode = ConcurrencyMode.Single)]
- B. [ServiceBehavior (InstanceContextMode = InstanceContextMode.PerCall, ConcurrencyMode = ConcurrencyMode.Reentrant)]
- C. [ServiceBehavior (InstanceContextMode = InstanceContextMode.PerSession, ConcurrencyMode = ConcurrencyMode.Multiple)]
- D. [ServiceBehavior (InstanceContextMode = InstanceContextMode.PerCall, ConcurrencyMode = ConcurrencyMode.Multiple)]

Answer: C

Question No : 8 - (Topic 1)

You are configuring services to be discoverable. The services must be discoverable without relying on a central server. Client applications that consume the services are on a network segment that is separate from the network segment that the services are located on.

A firewall blocks all TCP ports between the two network segments, but allows other protocols to pass through.

You need to ensure that the client applications can discover the services.

What should you do?

- A. Use ad-hoc discovery mode over HTTP.
- B. Use ad-hoc discovery mode over UDP.
- C. Use managed discovery mode over HTTP.
- D. Use managed discovery mode over UDP.

Answer: B

Explanation: Explanation/Reference:

Managed discovery modes are incorrect, they require central server for discovery. By default the .NET Framework contains support for Ad-Hoc discovery over the UDP protocol

Question No : 9 - (Topic 2)

You need to modify a client application that consumes a Windows Communication Foundation (WCF) service.

The service metadata is no longer available.

You need to modify the previously generated proxy to include asynchronous calls to the service.

What should you do?

- A. Update the service reference with the Generate asynchronous operations option.
- B. Create a partial class for the previously generated proxy and include the new asynchronous methods.
- C. Create a class with the same name as the previously generated proxy and add the new asynchronous methods. Add the new class to a namespace that is different from the original proxy.
- D. Create a class with the same name as the previously generated proxy and add the new asynchronous methods as partial methods. Add the new class to a namespace that is different from the original proxy.

Answer: B



Question No : 10 DRAG DROP - (Topic 1)

You have a client application that uses an existing Windows Communication Foundation (WCF) service. The client application contains a defined EndpointAddress object named endpointAddress.

A class named ServiceClient is generated by using the Svcutil tool to invoke the WCF service. Instances of the ServiceClient class are created as follows:

```
ServiceClient client = new ServiceClient(CreateBinding(), endpointAddress);
```

The client application must meet the following requirements:

-  Optimize message-level security when transporting both text files and large files.
-  Provide transport-level security by using the HTTPS protocol.