

# Bidirectional communications with .NET remoting

**Source:**

<http://www.tech-archive.net/Archive/DotNet/microsoft.public.dotnet.framework.remoting/2004-12/0390.html>

---

**From:** Matt Kosorok (*MattKosorok\_at\_discussions.microsoft.com*)

**Date:** 12/30/04

Date: Thu, 30 Dec 2004 10:09:01 -0800

I'm getting really close ( thanks to Ken Kolda, you're awesome, dude ).  
Here's what I'm trying to do: ( warning, long message here )

I have a remote object that has a public delegate. The host application uses the following code to publish the remote object:

```
// serverProvider object
BinaryServerFormatterSinkProvider serverProvider = new
BinaryServerFormatterSinkProvider() ;
// set filter level
serverProvider.TypeFilterLevel =
System.Runtime.Serialization.Formatters.TypeFilterLevel.Full ;
// set up IDictionary
IDictionary props = new Hashtable() ;
props["port"] = nMyTcpPort ;

// set up tcp channel
TcpChannel channel = new TcpChannel( props, null, serverProvider ) ;
ChannelServices.RegisterChannel( channel ) ;

// register remote object
ObjRef remoteObject = RemotingServices.Marshal( myRemoteObject, strTitle ) ;
```

So now I've published my remote object. My client object uses  
RemotingServices.Configure( client.config ) ;  
and the client.config file looks like this:

```
<configuration>
<system.runtime.remoting>
<application>
<client>
<wellknown
type = "MyObjectType, mydll"
url = "tcp://localhost:nMyTcpPort/strTitle" />
</client>
<channels>
<channel
```

```
ref = "tcp"
port = "0" >
<clientProviders>
  <formatter ref = "binary" />
</clientProviders>
<serverProviders>
  <formatter ref = "binary" typeFilterLevel = "Full" />
</serverProviders>
</channels>
</application>
</system.runtime.remoting>
</configuration>
```

I also have a listener dll on the client side, and an injector on the server side. With the client, I register a callback in the remote object that points to a method in the listener. I also register a callback with the listener that points to the method I want called in the client. Both the server-side injector and the client-side client app have a method that increases a count variable in the remote object.

When I test this, everything connects and communicates correctly with the remote singleton object. With the injector, I can increase the count variable of the remote object. With the client, it can also increase it, and both the injector's and the client's remote object count changes are seen correctly by both objects. ( i.e., the injector sets the remote object's count to 1, the client sees it as one, and increases it to 23, and the injector then sees 23 and so on ).

However, when I use the injector to execute the callback registered in the remote object, which is supposed to execute the method on the client-side in the listener.dll, which is then supposed to execute the method in the client-side client application, I don't actually get anything executed.

My guess is there's something different with how the server-side remote object is set up programmatically ( versus a config file ), and that my client-side config file is not appropriately set in the config file.

When I switch the host to provide the exact same remote object using a config file, I get full communication in both directions, but lose the ability to terminate the remote host at will ( I have to shut down the application ).

Here is the server-side config file:

```
<configuration>
<system.runtime.remoting>
  <application name = "Host" >
    <service>
      <wellknown
        mode = "Singleton"
        type = "myType, myDll"
```

microsoft.public.dotnet.framework.remoting: Bidirectional communications with .NET remoting

```
    objectUri = "strTitle" />
  </service>
</channels>
<channel ref = "tcp" port = myTcpPort>
  <serverProviders>
    <formatter ref = "binary" typeFilterLevel = "Full" />
  </serverProviders>
</channel>
</channels>
</application>
</system.runtime.remoting>
</configuration>
```

I'm so close to making this thing work exactly how I want to. I just have to overcome this last major obstical, and I'll be a very happy man.

```
--
- Matt Kosorok
```