# **Client Channel-Level Programming**

* 03/29/2017
* 3 minutes to read

This topic describes how to write a Windows Communication Foundation (WCF) client application without using the [System.ServiceModel.ClientBase<TChannel>](https://docs.microsoft.com/en-us/dotnet/api/system.servicemodel.clientbase-1) class and its associated object model.

## **Sending Messages**

To be ready to send messages and receive and process replies, the following steps are required:

Create a binding.

Build a channel factory.

Create a channel.

Send a request and read the reply.

Close all channel objects.

#### **Creating a Binding**

Similar to the receiving case (see [Service Channel-Level Programming](https://docs.microsoft.com/en-us/dotnet/framework/wcf/extending/service-channel-level-programming)), sending messages starts by creating a binding. This example creates a new [System.ServiceModel.Channels.CustomBinding](https://docs.microsoft.com/en-us/dotnet/api/system.servicemodel.channels.custombinding) and adds an [System.ServiceModel.Channels.HttpTransportBindingElement](https://docs.microsoft.com/en-us/dotnet/api/system.servicemodel.channels.httptransportbindingelement) to its Elements collection.

#### **Building a ChannelFactory**

Instead of creating a [System.ServiceModel.Channels.IChannelListener](https://docs.microsoft.com/en-us/dotnet/api/system.servicemodel.channels.ichannellistener), this time we create a [System.ServiceModel.ChannelFactory<TChannel>](https://docs.microsoft.com/en-us/dotnet/api/system.servicemodel.channelfactory-1) by calling [ChannelFactory.CreateFactory](https://docs.microsoft.com/en-us/dotnet/api/system.servicemodel.channelfactory.createfactory) on the binding where the type parameter is [System.ServiceModel.Channels.IRequestChannel](https://docs.microsoft.com/en-us/dotnet/api/system.servicemodel.channels.irequestchannel). While channel listeners are used by the side that waits for incoming messages, channel factories are used by the side that initiates the communication to create a channel. Just like channel listeners, channel factories must be opened first before they can be used.

#### **Creating a Channel**

We then call [ChannelFactory<TChannel>.CreateChannel](https://docs.microsoft.com/en-us/dotnet/api/system.servicemodel.channelfactory-1.createchannel) to create an [IRequestChannel](https://docs.microsoft.com/en-us/dotnet/api/system.servicemodel.channels.irequestchannel). This call takes the address of the endpoint with which we want to communicate using the new channel being created. Once we have a channel, we call Open on it to put it in a state ready for communication. Depending on the nature of the transport, this call to Open may initiate a connection with the target endpoint or may do nothing at all on the network.

#### **Sending a Request and Reading the Reply**

Once we have an opened channel, we can create a message and use the channel’s Request method **to send the request and wait for the reply** to come back. When this method returns, we have a reply message that we can read to find out what the endpoint’s reply was.

#### **Closing Objects**

To avoid leaking resources, we close objects used in communications when they are no longer required.

The following code example shows a basic client using the channel factory to send a message and read the reply.

C#Copy

using System;using System.ServiceModel;using System.ServiceModel.Channels;using System.ServiceModel.Configuration;namespace ProgrammingChannels

{class Client

{

static void RunClient()

{

//Step1: Create a binding with just HTTP.

BindingElement[] bindingElements = new BindingElement[2];

bindingElements[0] = new TextMessageEncodingBindingElement();

bindingElements[1] = new HttpTransportBindingElement();

CustomBinding binding = new CustomBinding(bindingElements);

//Step2: Use the binding to build the channel factory.

IChannelFactory<IRequestChannel> factory =

binding.BuildChannelFactory<IRequestChannel>(

new BindingParameterCollection());

//Open the channel factory.

factory.Open();

//Step3: Use the channel factory to create a channel.

IRequestChannel channel = factory.CreateChannel(

new EndpointAddress("http://localhost:8080/channelapp"));

channel.Open();

//Step4: Create a message.

Message requestmessage = Message.CreateMessage(

binding.MessageVersion,

"http://contoso.com/someaction",

"This is the body data");

//Send message.

Message replymessage = channel.Request(requestmessage);

Console.WriteLine("Reply message received");

Console.WriteLine("Reply action: {0}",

replymessage.Headers.Action);

string data = replymessage.GetBody<string>();

Console.WriteLine("Reply content: {0}", data);

//Step5: Do not forget to close the message.

replymessage.Close();

//Do not forget to close the channel.

channel.Close();

//Do not forget to close the factory.

factory.Close();

}

public static void Main()

{

Console.WriteLine("Press [ENTER] when service is ready");

Console.ReadLine();

RunClient();

Console.WriteLine("Press [ENTER] to exit");

Console.ReadLine();

}

}

}

<https://docs.microsoft.com/en-us/dotnet/framework/wcf/extending/client-channel-level-programming>