Contents

You can implement a custom *ServiceProxyEvents* to modify the communication configuration of your n-tier client application. The *ServiceProxyEvents* allows you to customize the configuration of the client WCF communications at run time.

Problem

Before showing some samples let's briefly discuss the API and why you might use this approach to customizing communication configuration.

public virtual void OnEndpointCreated(ServiceEndpoint endpoint)
Called when a ServiceEndpoint has been created. You can override this method if you need to customize the endpoint. For example, you can:

- Modify the default timeout values
- Add security to the binding (non-Silverlight)
- Add additional endpoint behaviors
- Change the endpoint address
- Change any part of the binding or replace the binding altogether.

public virtual void OnFactoryCreated(ChannelFactory factory)
Called when a ChannelFactory has been created. You can override this method to perform additional customization of the proxy before the channel is opened. For example, you can:

- Add additional behaviors
- Supply transport credentials

As both methods indicate, they're called *after* the endpoint or factory has been created. If your application has not supplied either an <objectServer> or <*serviceModel*> section then DevForce will not have enough information to continue. The exception is Silverlight applications, where the <*objectServer*> information defaults based on the download URL.

The methods will be called when communications are initialized to both the *EntityService* and *EntityServer* services. You can inspect the contract to see which service the endpoint or proxy is for.

Neither method is called on a per-request basis; in other words you cannot use these to customize every message sent between client and server. However, via behaviors, you can add message inspectors and filters which will be called on a per-request basis.

Remember that the configuration of your client and server must match. For example, you can't use transport security on one side and message security on the other.

WCF support in Silverlight is not as robust as in other .NET environments, so some options will not be available on the Silverlight client.

Solution

The *ServiceProxyEvents* allows you to customize the configuration of the client WCF communications at run time.

Here's a dummy class, which we'll show once so you can see the namespaces in use.

```
using System.ServiceModel.Channels;
using IdeaBlade.Core.Wcf.Extensions;
public class ProxyEvents  : IdeaBlade.EntityModel.ServiceProxyEvents {
 public override void OnEndpointCreated(System.ServiceModel.Description.ServiceEndpoint endpoint) {
   base.OnEndpointCreated(endpoint);
  }
 public override void OnFactoryCreated(System.ServiceModel.ChannelFactory factory) {
   base.OnFactoryCreated(factory);
  }
}
```

```
Imports System.ServiceModel.Channels
Imports IdeaBlade.Core.Wcf.Extensions
Public Class ProxyEvents
```

```
Inherits IdeaBlade.EntityModel.ServiceProxyEvents
Public Overrides Sub OnEndpointCreated(ByVal endpoint As System.ServiceModel.Description.ServiceEndpoint)
  MyBase.OnEndpointCreated(endpoint)
End Sub
Public Overrides Sub OnFactoryCreated(ByVal factory As System.ServiceModel.ChannelFactory)
  MyBase.OnFactoryCreated(factory)
End Sub
End Class
```

Example

Override communication timeouts

Here's a sample re-setting communication timeouts. DevForce uses the WCF default values for all timeouts. On the client, the only timeout of concern is the *SendTimeout*, which defaults to 1 minute.

```
 public override void OnEndpointCreated(System.ServiceModel.Description.ServiceEndpoint endpoint) {
   base.OnEndpointCreated(endpoint);
    endpoint.Binding.SendTimeout = TimeSpan.FromMinutes(2);  // governs request/reply
 }
```

```
Public Overrides Sub OnEndpointCreated(ByVal endpoint As System.ServiceModel.Description.ServiceEndpoint)
  MyBase.OnEndpointCreated(endpoint)
endpoint.Binding.SendTimeout = TimeSpan.FromMinutes(2) ' governs request/reply
End Sub
```

Other modifications

Here's a sample changing the service address. Note in Silverlight you'll need a cross domain policy in place if you do something like this, since the XAP download address will differ from the service address.

```
 public override void OnEndpointCreated(System.ServiceModel.Description.ServiceEndpoint endpoint) {
   base.OnEndpointCreated(endpoint);
    endpoint.Address = new System.ServiceModel.EndpointAddress(endpoint.Address.Uri.AbsoluteUri.Replace("myserver", "yourserver"));
  }
```

```
Public Overrides Sub OnEndpointCreated(ByVal endpoint As System.ServiceModel.Description.ServiceEndpoint)
  MyBase.OnEndpointCreated(endpoint)
endpoint.Address = New System.ServiceModel.EndpointAddress(endpoint.Address.Uri.AbsoluteUri.Replace("myserver", "yourserver"))
End Sub
```

Additional resources

The ABCs of Endpoints