Contents

You can implement a custom *ServiceHostEvents* to modify the communication configuration of your EntityServer for any deployment type. The *ServiceHostEvents* allows you to customize the configuration of the *EntityServer* WCF services 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
- Add additional endpoint behaviors
- Change the endpoint address
- Change any part of the binding or replace the binding altogether.

public virtual void OnServiceHostCreated(ServiceHost host)

Called when a ServiceHost has been created. You can override this method to perform additional customization of the service before it is opened. For example, you can:

- Add additional endpoints or remove or modify existing ones
- Add, modify or remove service behaviors

As both methods indicate, they're called *after* the endpoint or service 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 when the *EntityServer* is hosted by IIS, since the base address of the IIS application is used to determine the default configuration.

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

Solution

The *ServiceHostEvents* allows you to customize the configuration of the *EntityServer* WCF services 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 ServiceEvents : IdeaBlade.EntityModel.Server.ServiceHostEvents {
 public override void OnEndpointCreated(System.ServiceModel.Description.ServiceEndpoint endpoint) {
  base.OnEndpointCreated(endpoint);
  }
 public override void OnServiceHostCreated(System.ServiceModel.ServiceHost host) {
  base.OnServiceHostCreated(host);
  }
}
```

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

```
Inherits IdeaBlade.EntityModel.Server.ServiceHostEvents
Public Overrides Sub OnEndpointCreated(ByVal endpoint As System.ServiceModel.Description.ServiceEndpoint)
  MyBase.OnEndpointCreated(endpoint)
End Sub
Public Overrides Sub OnServiceHostCreated(ByVal host As System.ServiceModel.ServiceHost)
  MyBase.OnServiceHostCreated(host)
End Sub
End Class
```

Example

Accept Windows credentials when using Windows Authentication in IIS

Here's an example which allows for the service to receive Windows credentials when Windows Authentication is enabled in IIS. The *AuthenticationScheme* on the transport element defaults to *anonymous*, but if you've enabled Windows authentication for your application you should set the *AuthenticationScheme* to correspond. Here we set the scheme to "Negotiate".

Note how the binding elements are obtained; when modifying any elements within the binding or inserting or replacing elements, you'll need to first obtain the elements of the current binding, and then replace the endpoint binding when done.

```
public override void OnEndpointCreated(System.ServiceModel.Description.ServiceEndpoint endpoint) {
  base.OnEndpointCreated(endpoint);
  if (endpoint.Binding is CustomBinding) {
     var binding = endpoint.Binding as CustomBinding;
     var elements = binding.CreateBindingElements();
     var tbe = elements.Find<TransportBindingElement>();
     var httpbe = tbe as HttpTransportBindingElement;
     httpbe.AuthenticationScheme = System.Net.AuthenticationSchemes.Negotiate;
     endpoint.Binding = new CustomBinding(elements);
    }
  }
```

```
Public Overrides Sub OnEndpointCreated(ByVal endpoint As System.ServiceModel.Description.ServiceEndpoint)
  MyBase.OnEndpointCreated(endpoint)
  If TypeOf endpoint.Binding Is CustomBinding Then
 Dim binding = TryCast(endpoint.Binding, CustomBinding)
 Dim elements = binding.CreateBindingElements()
 Dim tbe = elements.Find(Of TransportBindingElement)()
 Dim httpbe = TryCast(tbe, HttpTransportBindingElement)
 httpbe.AuthenticationScheme = System.Net.AuthenticationSchemes.Negotiate
 endpoint.Binding = New CustomBinding(elements)
  End If
End Sub
```

Add MEX support

Here's a sample which adds a MEX endpoint and metadata behavior to the service, using a helper function in DevForce. By default, metadata publishing is not enabled for DevForce services.

```
public override void OnServiceHostCreated(System.ServiceModel.ServiceHost host) {
   IdeaBlade.EntityModel.RemoteServiceFns.AddMexEndpoint(host);
  }
```

```
Public Overrides Sub OnServiceHostCreated(ByVal host As System.ServiceModel.ServiceHost)
  IdeaBlade.EntityModel.RemoteServiceFns.AddMexEndpoint(host)
End Sub
```

Additional resources

The ABCs of Endpoints
Custom Bindings