Contents

A DevForce **n-tier client** is often a Silverlight, Windows Store, Windows Phone, Windows Presentation Foundation (WPF), or Windows Forms application, but it might also be a console application, a Windows Service, or an web application. Although deployment differs for these different application types, the requirements of each are generally the same. Silverlight, Windows Store and Phone applications have unique deployment requirements and are discussed separately; here we discuss deployment of other client application types.

Files and assemblies

The following assemblies are always required:

IdeaBlade.Core.dll
IdeaBlade.EntityModel.dll
IdeaBlade.Linq.dll
IdeaBlade.Validation.dll

You'll usually need your .config file too:
app.config.exe (or web.config)

If you're deploying a Code First model, you'll also need the following:

IdeaBlade.Aop.dll
PostSharp.dll

You'll of course also need all of your own client-side assemblies and other files needed by your application. If you've implemented any custom extensions of DevForce components, remember that some will be pertinent to only the client or only the server. In these cases, particularly for server-only components, it's best to deploy these assemblies to only the appropriate tier.

You may place the DevForce assemblies in the GAC, although there's generally no compelling reason to do so, unless you have a large number of DevForce applications installed and they are all using the same DevForce version.

You do not need to, and in most cases should not, install DevForce to the target machine. Runtime license information is found in your model assembly and not the registry.

The config file

You do not need the <*connectionStrings*> (and <*edmKeys*>) in your client application (unless you are also directly accessing local resources). Having database connection information present represents a security vulnerability, easily fixed by removing this information.

You'll generally use the <*ideablade.configuration*> section to provide connection information to the *EntityServer* and logging information, although neither is strictly necessary.

Logging

You can use the <logging> element to set the file name and location of the debug log, or to turn logging off altogether. You also might want to archive log files. Generally other logging attributes are not needed, or used only for debugging purposes.

A typical config file might contain the following logging information:

```
<logging logFile="log\DevForceDebugLog.xml" archiveLogs="true" />
```

... to place the file in a sub-folder named log and archive files automatically.

Also see the Log topic for more information.

ObjectServer

You'll usually use the <objectServer> element to specify the location of the remote *EntityServer* and to enable remote communications.

We say "usually" because you can also specify this information programmatically, by working directly with the *IdeaBladeConfig.Instance* when the application starts. Other options are to use the <*system.serviceModel*> element to configure communications, or to implement a custom *ServiceProxyEvents* to customize configuration at run time. For more information on how DevForce determines the defaults and how to customize them, see the advanced configuration topic.

A typical config file might the contain the following to communicate with an IIS-hosted *EntityServer*:

```
<objectServer remoteBaseURL="http://yourserver"
             serverPort="80"
             serviceName="yourapp/EntityService.svc"
             >
  <clientSettings isDistributed="true" />
</objectServer>
```

If your server is using SSL, then it might look something like this instead:

```
<objectServer remoteBaseURL="https://yourserver"
             serverPort="443"
             serviceName="yourapp/EntityService.svc"
             >
  <clientSettings isDistributed="true" />
</objectServer>
```

... where we've changed the protocol and port information.

If your *EntityServer* is hosted by either the DevForce-supplied *ServerService* or *ServerConsole* hosts, then it might look something like this instead:

```
<objectServer remoteBaseURL="http://yourserver"
             serverPort="9009"
             serviceName="EntityService"
             >
  <clientSettings isDistributed="true" />
</objectServer>
```

Note the differences from when your server is hosted under IIS: the serviceName does not contain the web application name or .svc extension, and the port is usually not port 80. The port can be any free port that you've configured your server to listen on - we often use port 9009 in our samples (as we did here) but there are no DevForce requirements for this port number.

Regardless of the host, your client configuration should match the server's configuration. If the *EntityServer* is using https, then the client must too. If the server is listening on port 80, then the client must use that port.

The <*serverSettings*> are not pertinent to the client application in an n-tier deployment.

Packaging for deployment

The computers on which you deploy the application likely have the .NET Framework 4.5 already installed or you will install it as part of your deployment.

Generally client applications are installed via an installer or a ClickOnce deployment.

Security

Remember that a client application can be compromised. Its code can be disassembled, its configuration, log and other files peered at, and its communications with the application server tampered with.

See the Security topic for more information on steps you can take to secure your application.