Contents

- Assemblies
- Packaging
- Configuration
- Security
- Troubleshooting
    - FIPS Compliance

Deployment of your **Silverlight application** will generally be performed as part of the deployment of the EntityServer to a web site. We have additional information on configuring the web site here.

Assemblies

The following assemblies are always required:

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

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

IdeaBlade.Aop.SL.dll
PostSharp.dll (or PostSharp.SL.dll for DevForce versions prior to 7.2.0)

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.

Packaging

Generally, Visual Studio has taken care of the packaging of your Silverlight application into one or more deployment packages, but there are a few additional things to consider.

Your Silverlight application will generally have a main XAP, and possibly additional secondary deployment packages to be downloaded as needed.

You can enable Application Library Caching to help reduce the XAP file size. All DevForce Silverlight assemblies support ALC. You'll see associated zip files in the web project's *ClientBin* folder for the DevForce assemblies when ALC is used.

You can enable your application to run out-of-browser and automatically update when the application changes. DevForce supports OOB Silverlight applications, and has additional offline support.

Since the client browser will usually cache your XAP files, you should ensure that updates to your application will be seen by your users. There are several ways of ensuring this, but the easiest might be to append information to the URL for the XAP file to indicate your current application version. For example:

```
<object data="data:application/x-silverlight-2," type="application/x-silverlight-2" width="100%" height="100%">
  <param name="source" value="ClientBin/MySilverlightApp.xap?version=1.0.0" />
  .....
</object>
```

You'll need to ensure that the client browser has the appropriate version of Silverlight installed, and you may want to customize the Silverlight install experience for your users.

On the server, you'll deploy any XAPs (and ZIPs for ALC-enabled assemblies) to the *ClientBin* folder of your web site. You'll also need to ensure that the XAP mime type is registered in IIS. See http://learn.iis.net/page.aspx/262/silverlight for more information on registering the mime type for different IIS versions.

Configuration

By default DevForce will assume that the *EntityServer* is located in the same location from which the XAP was downloaded. For example, if the XAP for your application was downloaded from http://myhost/myapp/default.aspx, DevForce will default the service URL to http://myhost/myapp/EntityService.svc. If your *EntityServer* is located at this address, then you won't need an app.config in your Silverlight application.

If you're using SSL and the XAP is downloaded using https, then the default *EntityServer* address will also use https.

If your EntityServer is at a different location, or you need to customize the communication defaults, then you will need to override the DevForce defaults.

If only the location needs to be changed you can include an app.config file in your application with the appropriate <*objectServer*> information. Instead of including an app.config, you can also specify <*objectServer*> information programmatically by working directly with the *IdeaBladeConfig.Instance* when the application starts.

If you need to customize communications (such as timeouts) you can add a *ServiceReferences.ClientConfig* file to your application to specify the <*system.serviceModel*> information required, or implement a custom *ServiceProxyEvents* to customize configuration at run time. (Note that you cannot define the GZip binding extension in the *ClientConfig*; DevForce uses this in the default configuraiton to compress all requests to and responses from the server. If you use a *ClientConfig* file you will need to add the GZip binding element programmatically. See this sample for more information.)

For more information on the default configuration and how to customize it, see here.

If your Silverlight application is not at the same location as your *EntityServer* you will also need to deploy a clientaccesspolicy.xml file to the server hosting the *EntityServer* to enable cross domain access.

Sample Silverlight app.config, ServiceReferences.ClientConfig and policy files are provided in the deployment snippets.

Security

Remember that a Silverlight application is vulnerable to malicious users. Its code can be disassembled, its Isolated Storage 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.

Troubleshooting

See the topic on n-tier troubleshooting for more information.

FIPS Compliance

If your Silverlight application will be served from a web server on which FIPS (Federal Information Processing Standards) compliance is enforced, you will need to make the following changes to both the web.config and startup pages.

In the web.config, you must set debug to **false** when FIPS is enabled. This is true even during development: you *cannot* set debug to true with FIPS enabled!

```
<system.web>
  <compilation debug="false">
</system.web>
```

If you use an .html page instead of an .aspx page to host the Silverlight XAP control, you will need to delete the following:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
```