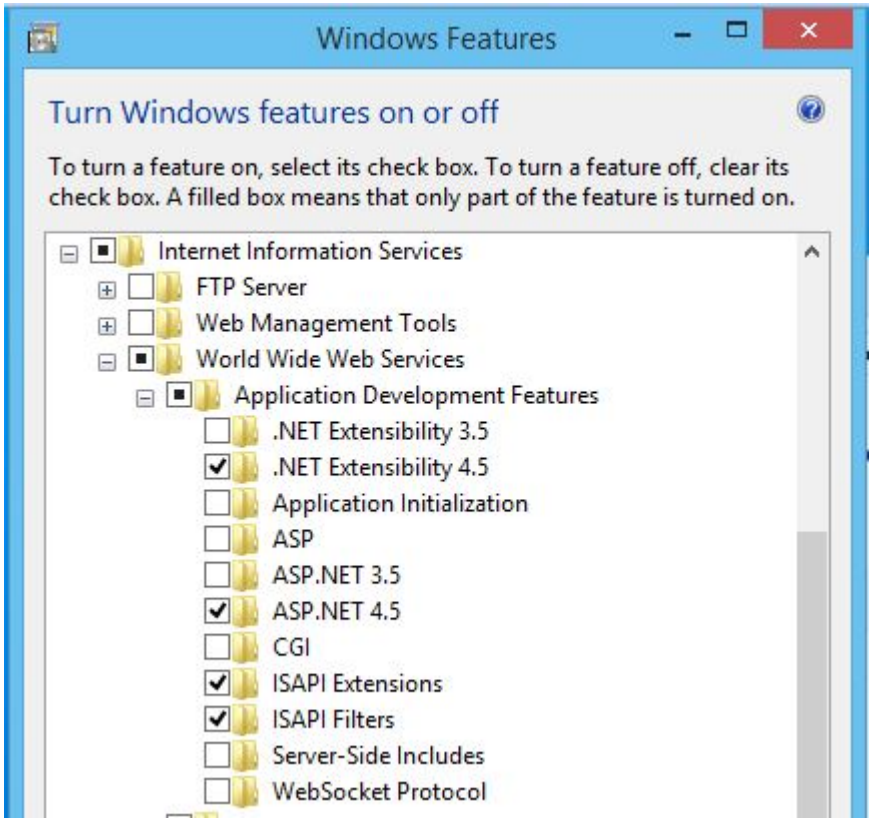
Contents

The most typical, and most robust, deployment of the EntityServer is with Internet Information Services (IIS). Unfortunately, even here, issues may arise from time to time.

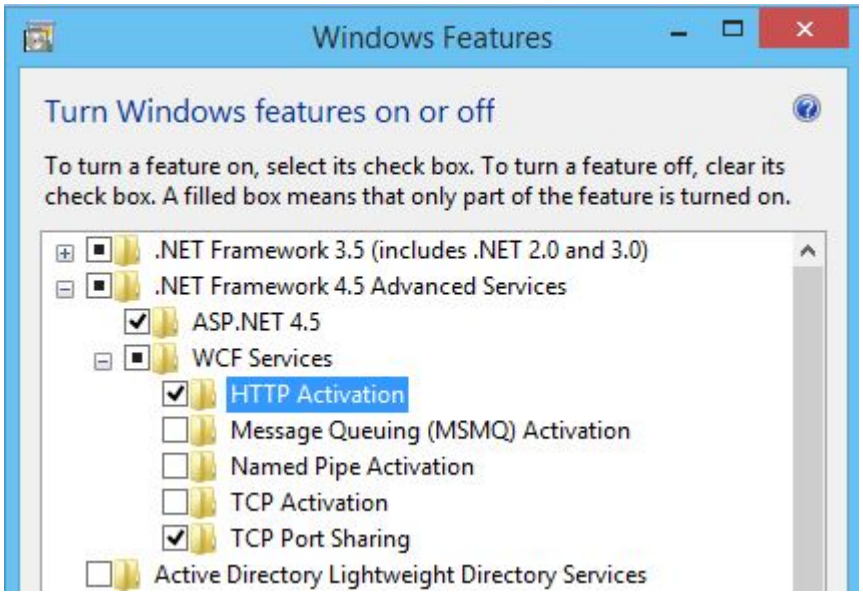
Miscellaneous

- Remember that .NET 4.5 must be installed.
- The *IdeaBlade.EntityModel.**Web*** assembly must always be deployed with an IIS-hosted *EntityServer*.
- If you get an exception with the message, "*'IdeaBlade.EntityModel.Server.EntityServiceHostFactory' could not be loaded*", confirm that you are using an **application pool** that supports **.NET Framework 4.5** and is in the **integrated pipeline mode**.
- If the EntityServer will not start in IIS on Windows 8 -

  Ensure that IIS is installed and will serve dynamic content:



  Also ensure that HTTP Activation is enabled for WCF Services:



- WCF must be registered with IIS (non-Windows 8 OS) - See here for information on using *ServiceModelReg*. You may also need to register ASP.NET with IIS using aspnet_regiis.
- "The login credentials supplied are invalid" when using ASP.NET Forms authentication - If you've moved your aspnetdb database, or used the database prior to ASP.NET 4.5, the hash algorithm may have changed. See here for information on setting the machineKey and hashAlgorithmType.
- "Could not load file or assembly 'App_Web_..." after making a change to one or more of the files in the application directory. You may have encountered a problem that occurs when files in the application folder no longer match the compiled version located in the “Temporary ASP.NET Files” folder. You can force a rebuild of your application by deleting the “bin” folder and then replace it with a copy or by running the “aspnet_compiler.exe” command with the “-c” switch. You can find the command by first browsing to the folder “%SystemRoot%\Microsoft.NET\Framework\” and then open the v4.0.xxxxx subfolder (the numbers after v4.0 can vary). Here is an example using the virtual directory name of the application: aspnet_compiler –v /MyApp -c .
- "System.BadImageFormatException: Could not load file or assembly 'XXXX.XXXXX' or one of its dependencies. An attempt was made to load a program with an incorrect format." - By default IIS 7 won't load x86 (32 bit) assemblies on a 64-bit OS. You can fix this either by changing the target platform on the assembly to "Any CPU", or by setting the "Enable 32-bit Applications" setting for the application pool.
- All communications from DevForce services to client applications are compressed by default. Messages will carry a content type of "application/x-gzip". If you attempt to use IIS compression without disabling DevForce compression you will receive a "No endpoint listening" error.

Must CopyLocal be true in Visual Studio?

When you deploy your application to the web server - where DevForce will not be installed - you'll want all required assemblies to be in the bin subfolder. It's easier to see which assemblies you'll need to deploy if they're in your bin folder during testing.

If you navigate to the service page and an assembly is missing, you'll see a compilation error explaining the problem.

ASP.NET settings

You may need to modify several ASP.NET runtime settings to control the execution timeout and maximum request size.

The default execution timeout is 110 seconds. If your application experiences operation timeouts you can modify the *executionTimeout* setting. The value is specified in seconds.

The default maximum request size is 4MB. If your client application sends large data packets to the server, for example if saving many entities or very large entities, increase the *maxRequestLength* value. The value is specified in kilobytes.

Here's a sample *httpRuntime* element with both settings modified from their defaults:

```
<system.web>
  <httpRuntime maxRequestLength="8192" executionTimeout="130" />
</system.web>
```

When you created your project an *httpRuntime* element was added to target .NET 4.5. You should not remove this setting, as without it .NET defaults to "4.0" and quirks mode, which will likely cause your application to fail.

```
<system.web>
  <httpRuntime  targetFramework="4.5" />
</system.web>
```

Web.config deployment transformations

You may be using web.config transforms when deploying the *EntityServer* to IIS. You can use transforms with the *ideablade.configuration* section elements and attributes too. The default web.config added to your web project contained an *ideablade.configuration* section which opened with the following:

```
<ideablade.configuration version="6.00" xmlns="http://schemas.ideablade.com/2010/IdeaBladeConfig">
...
```

The namespace is primarily used to provide Intellisense support when editing the file within Visual Studio, but it does complicate transforms and XPath expressions. Unless you're an XPath expert, the quickest way to get your transforms working is to include the namespace in the transform file. For example, the following transform will replace the *logging* attributes:

```
<configuration xmlns:xdt="http://schemas.microsoft.com/XML-Document-Transform">
  <ideablade.configuration version="6.00" xmlns="http://schemas.ideablade.com/2010/IdeaBladeConfig">
    <logging logFile="log\DebugLog.xml" shouldLogSqlQueries="true" xdt:Transform="Replace" />
  </ideablade.configuration>
</configuration>
```

Alternately, you can remove the namespace attribute from the *ideablade.configuration* element in your web.config file, but you'll lose Intellisense support.

Additional Resources

- ASP.NET and IIS Configuration
- IIS Learning Center