How to log the internal assembly loading

For OPC “Classic” (COM-based) specifications, QuickOPC.NET internally loads additional assemblies that you do not reference explicitly. The whole process is rather complicated. In case of problems, the procedure described in this application note will help with the diagnosis, by logging the messages associated with assembly loading.

The logging is enabled by a .NET trace switch, named "`OpcLabs.Reflection.AssemblyLoading`". Setting its value to "`1`" enables the logging. You also need to direct the trace to proper listener, using the standard means provided by .NET tracing facility. It is up to you how you enable the switch or configure the listener(s). Typically, it is done using the application configuration file, with the advantage that it can be done without rebuilding the application.

The example below shows the application configuration file with the tracing for assembly loading enabled, and the listener configured to store the output into a comma-delimited file “Trace.csv” (in the same folder as the application).

```
<?xml version="1.0"?>
<configuration>
  <startup>
    <supportedRuntime version="v4.0" sku=".NETFramework,Version=v4.5"/>
  </startup>
  <system.diagnostics>
    <switches>
      <add name="OpcLabs.Reflection.AssemblyLoading" value="1" />
    </switches>
    <trace autoflush="true" indentsize="4">
      <listeners>
        <add name="myListener"
             type="System.Diagnostics.DelimitedListTraceListener"
             traceOutputOptions="DateTime, ProcessId, ThreadId"
             delimiter=","
             initializeData="Trace.csv" />
        <remove name="Default" />
      </listeners>
    </trace>
  </system.diagnostics>
</configuration>
```

The relevant parts of the file are highlighted.

Please refer to Microsoft documentation for details on the application configuration files, the diagnostic switches, the trace listeners, and so on. The application configuration file needs to be named the same as your application, with an added “.config” extension, and placed alongside the application, For example, for “MyApp.exe”, the configuration file is “MyApp.exe.config”. Note that the development tools sometimes provide “shortcuts” for the naming and placing procedure. For example, Visual Studio C# projects contain an app.config file, which becomes the application configuration file automatically – Visual Studio copies it to the output folder and renames it appropriately.