1. HOW TO ENABLE EXTENDED TRACING

For advanced troubleshooting, it is sometimes necessary to obtain information about QuickOPC-UA internal status and activities. This can be done by enabling extended tracing, as described in this application note.

The logging is enabled by various .NET trace switches, and settings in configuration file sections. Unless you are able to observe the output in the debugger or using a special tool, you also need to direct the trace to a proper listener, using the standard means provided by .NET tracing facility. It is up to you how you enable the switches 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 extended tracing enabled.

```xml
<?xml version="1.0"?>
<configuration>
  <configSections>
    <section
      name="OpcLabs.EasyOpc.UA.Toolkit.SdkTrace"
      type="OpcLabs.EasyOpc.UA.Toolkit.SdkTraceSection,OpcLabs.EasyOpcUAInternal" />
  </configSections>
  <OpcLabs.EasyOpc.UA.Toolkit.SdkTrace traceOutput="3" >
  </OpcLabs.EasyOpc.UA.Toolkit.SdkTrace>
  <startup>
    <supportedRuntime version="v4.0" sku=".NETFramework,Version=v4.5"/>
  </startup>
  <system.diagnostics>
    <switches>
      <add name="OpcLabs.CallDiagnostics.Display.CallPort" value="1" />
      <add name="OpcLabs.CallDiagnostics.Display.Enabled" value="1" />
      <add name="OpcLabs.CallDiagnostics.Display.EnterPort" value="1" />
      <add name="OpcLabs.CallDiagnostics.Display.ExitPort" value="1" />
      <add name="OpcLabs.CallDiagnostics.Display.LeavePort" value="1" />
      <add name="OpcLabs.CallDiagnostics.Display.TickCount" value="1" />
      <add name="OpcLabs.EasyOpc.UA.Toolkit.Sdk.DisplayCalls" value="1" />
      <add name="OpcLabs.EasyOpc.UA.Toolkit.SdkCallback.DisplayCalls" value="1" />
      <add name="OpcLabs.EasyOpc.UA.Toolkit.SdkEnvironment.DisplayCalls" value="1" />
      <add name="OpcLabs.EasyOpc.UA.Toolkit.SdkMethod.DisplayCalls" value="1" />
      <add name="OpcLabs.EasyOpc.UA.Toolkit.SdkTarget.DisplayCalls" value="1" />
      <add name="OpcLabs.EventTracing.TraceLogEntries" value="1" />
    </switches>
  </system.diagnostics>
</configuration>
```

The relevant parts of the file are highlighted. Specifically, following information will be contained in the traces:

- All trace information from inside OPC UA .NET Stack and SDK.
- Calls to and from OPC UA .NET Stack and SDK.
- Log entries generated by the component.

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.

In hosted environments, such as ASP.NET, the name and location of the configuration file may be different as well.

By default, the traces are directed to the Windows debug output, which you can observe e.g. when you run the program under a debugger, or it can be viewed and captured using specialized tools such Sysinternals' DebugView (http://technet.microsoft.com/en-us/sysinternals/bb896647.aspx).

The configurable extended tracing is available in following QuickOPC versions and builds:

- QuickOPC 5.31.1424.1, and later builds of QuickOPC 5.31.
- QuickOPC 5.32 (not released at the time of writing this document) and later versions of QuickOPC.

2. TRACE LISTENER CONFIGURATION

If you do not want to use the default debug trace configuration (directing the traces to the debug output), you can configure the trace listener(s) in various ways. Please refer to Microsoft documentation for details. For example, to store the output into a comma-delimited file "Trace.csv" (in the same folder as the application), use the following configuration part:

```
<system.diagnostics>
    <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>
```

***