BizTalk CAT Instrumentation Framework Controller User Guide

Version 1.0

Updated August 19, 2010

Hosted on GitHub at https://github.com/tfabraham/BizTalkCATIFController

1. Introduction

The BizTalk CAT Instrumentation Framework is a high performance tracing/logging framework for BizTalk that builds upon the Event Tracing for Windows (ETW) infrastructure. It was created by Microsoft's BizTalk Customer Advisory Team (CAT). Microsoft used essentially the same framework to instrument BizTalk itself, as well as many recently released adapters.

The BizTalk CAT Instrumentation Framework is available from the GitHub site.

The BizTalk CAT team has also posted several blog entries about the CAT I.F. here: http://blogs.msdn.com/b/appfabriccat/

The downside of the CAT I.F. is that starting and stopping traces requires running command-line scripts, and by default the log data is viewable in a text file only after a trace is stopped. Many BizTalk developers are accustomed to using Trace.WriteLine() or Debug.WriteLine() in combination with the Microsoft SysInternals DebugView tool to see diagnostic messages in real time.

Enter the BizTalk CAT Instrumentation Framework Controller. The Controller is an easy-to-use GUI for the BizTalk CAT Instrumentation Framework. The Controller lets you start and stop a trace and adjust filter options. It can easily enable real-time tracing to Microsoft SysInternals DebugView (or other debuggers), to a log file or to both at the same time.

The Controller is designed for use both on development machines *and* production servers. Unlike Trace.WriteLine() or Debug.WriteLine(), with the BizTalk CAT I.F. you can enable tracing on a production server with only a negligible impact on performance (when tracing to a file).

2. Software Requirements

The BizTalk CAT Instrumentation Framework Controller requires .NET 3.0 SP1 or newer.

The BizTalk CAT I.F. itself requires .NET 3.5, which is not common on BizTalk Servers prior to BizTalk 2009.

For development machines, the recommended minimum is Windows XP SP3, but Windows Vista or Windows 7 is ideal. Windows XP SP2 users may need this hotfix: http://support.microsoft.com/kb/940322

The exporter's solution and project files are in Visual Studio 2008 format.

3. User Interface

Here's a screenshot of the user interface:

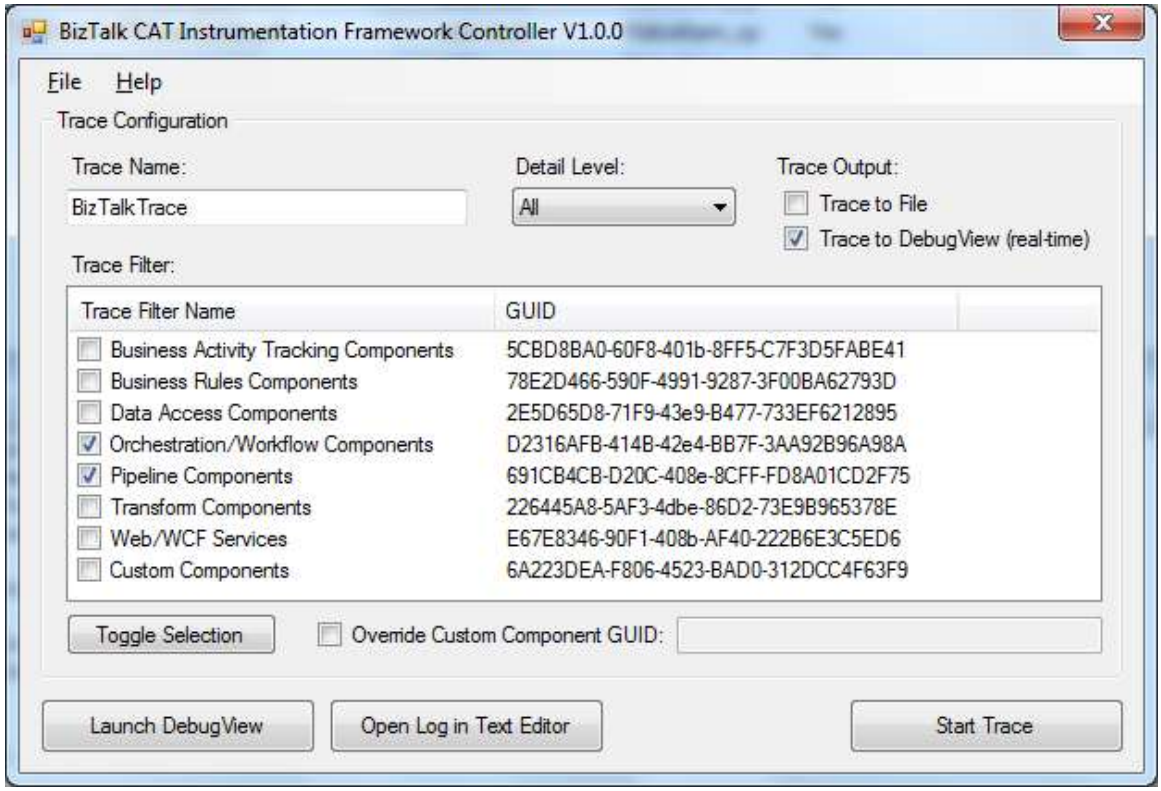

Figure 1 - User Interface

Before starting a trace, enter a name, select the desired detail level and trace output options and select which categories to trace. Click "Start Trace" to begin data capture.

Most of the options are self-explanatory, with a few clarifications:

"Trace to File" is the highest performance option and writes all trace data to a binary file. When the trace is stopped, the binary trace data is converted to a human-readable text file. The "Open Log in Text Editor" button will open that text file after a trace has been completed with "Trace to File" checked. Keep in mind that the log is circular, so after it fills up the oldest events will be overwritten.

"Trace to DebugView" causes all trace events to route through the OutputDebugString() Windows API function, which DebugView monitors (as can other debuggers like Visual Studio). This can cause a noticeable increase in CPU utilization, so this option is recommended for development machines only, or when performance is not critical.

"Override Custom Component GUID" is used to limit Custom Component tracing to a specific .NET class, which must be annotated with the Guid attribute. Please see the BizTalk CAT team's blog entitled "How to Support Component-Level Instrumentation

Using BizTalk CAT Instrumentation Framework." The value that you would enter in this text box is the GUID from your class's Guid attribute.

The Controller uses two Windows utilities behind the scenes: tracelog.exe and tracefmt.exe. If you are having any trouble enabling logging, the Options dialog box has an option to log the command-line program output to Debug.txt, located next to the Controller's EXE.

While a trace is running, you'll find it listed in the Performance Monitor's Data Collector Sets/Event Trace Sessions node on newer versions of Windows (Vista or newer).

4. The Author

Thomas F. Abraham, founder of IllumiTech Consulting LLC, has been the project owner and sole developer for the Deployment Framework for BizTalk and other BizTalk open-source projects since mid-2008.

He has an extensive background in software development, architecture, configuration management and systems engineering, helping to build high-performance, mission-critical applications for companies including Nasdaq, Best Buy and Wells Fargo. Over the last 16-plus years, Thomas has worked with technologies ranging from C/C++ to BizTalk Server to Exchange and .NET, was the lead author of the book "Visual Basic .NET Solutions Toolkit" from Wrox Press and a presenter at the 2006 SOA & Business Process Conference in Redmond, WA.

He holds a number of Microsoft certifications, including MCPD, MCSD, MCT (inactive) and TS for both BizTalk 2004 and 2006.

His blog is located at http://www.tfabraham.com.