**End to End Customer Application Tracing using Zipkin-Azure**

## Overview

Zipkin with Application Insights as backend data store is built for leveraging application performance management features provided by Application Insights along with tracing capabilities provided by Zipkin. This integration makes monitoring and debugging and your distributed systems much easier.

This hands-on lab will guide you through the following features:

1. Instrument a distributed Azure application involving a WCF web service amd a WPF client application.
2. Use Application Insights to query trace data.
3. Use the Zipkin UI to view the trace information.

### About the code

This client application sends arbitrary text to a WCF service and shows the response id generated by WCF service.

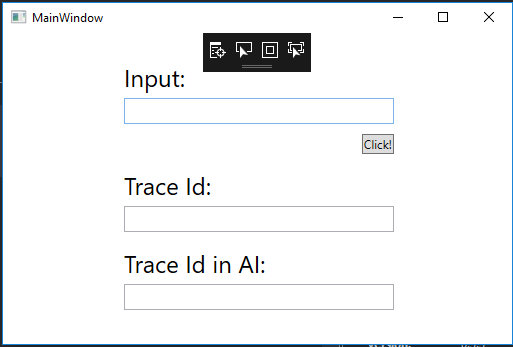
## Scenario 1

In this code challenge, developers instrument a client application interacting with a web service using [zipkin4net](https://github.com/criteo/zipkin4net) library via http. Once they instrument, they can see the interacting services in Zipkin UI and AppInsights UI, and they can also view the logged data in the AppInsights storage.

### Part One

1. To begin, open the “DemoZipkinAppInsights.sln” solution in Visual Studio 2017 and press F5 to compile and launch the WPF, Web Service app on the local machine.

You should be presented with an application that looks like this:



This page is designed to take a random input and pass it on to a **web service(http://localhost:1331/)**.

1. Type any input and press “Click”:

Currently no tracking data is being recorded for this trace ID. We will instrument the services so its path through the system is recorded.

1. Exit the program.

Instrumenting the front-end WPF service:

1. In Visual Studio, click “View” in the top menu and choose “Solution Explorer”.
2. Under the “WpfAppClient” project, expand the “MainWindow.xaml” file and open the “MainWindow.xaml.cs” file. Make the following code changes:
3. In the ***“MainWindow”*** method copy the following line under the comment:

|  |
| --- |
| //Specify Zipkin server URL to transport traces to.  Utilities.InitializeTraceConfig(zipkinServerUrl); |

This initializes and configures tracing.

1. In the “***Button\_Click***” method make the following code changes.
2. Create new trace by copying the following line of code under its comment.

|  |
| --- |
| // Create new Trace.  Trace trace1 = Trace.Create(); |

1. Record the first event by copying following two lines of code under the next comment.

|  |
| --- |
| //Record sr(server receive) and ss(server send) events to acknowledge click event.    Utilities.RecordAnnotation(trace1, Utilities.ZKAnnotations.SR, "WpfAppClient", "GET");  Utilities.RecordAnnotation(trace1, Utilities.ZKAnnotations.SS); |

1. Before calling the WCF Service, create a child trace.

|  |
| --- |
| //Create child trace and record (cs) Client Send event before starting a new Span.  Trace childTrace = trace1.Child();  Utilities.RecordAnnotation(childTrace, Utilities.ZKAnnotations.CS, "WpfAppClient", "GET"); |

1. Add the current childTrace parameter to the SubmitJob call by inserting the highlighted code.

|  |
| --- |
| //Call WCF service with input text  Task<string> response = SubmitJob(inputBox.Text, childTrace); |

1. Copy the following line to the end of the method, to record the response from the server.

|  |
| --- |
| //Record cr (child receive) event to complete current Span.  Utilities.RecordAnnotation(childTrace, Utilities.ZKAnnotations.CR); |

1. In the same file, in the “***SubmitJob”*** method, make the following changes:
2. Add a “trace” parameter of type “Trace” to the “SubmitJob” function by inserting the highlighted code.

SubmitJob(string content, Trace trace)

1. Before calling the WCF service, inject the current trace context into the http headers by copying the following line of code under the comment.

|  |
| --- |
| //Inject trace context into Http headers before calling WCF service.  Utilities.InjectTraceContextIntoHttpRequest(trace, requestMsg); |

1. Type “Ctrl+S” to save the changes to the MainWindow.xaml.cs file.

Instrumenting the back-end WCF service:

1. Click “View” in the top menu and choose “Solution Explorer”.
2. Under the “WcfService1” project, expand “DemoService.svc” and open the “DemoService.svc.cs” file.
3. In the ***“SubmitNewJob”*** method, make the following changes:
4. Initialize tracing by copying the following line under the comment:

|  |
| --- |
| //Specify Zipkin server URL to transport traces to.  Utilities.InitializeTraceConfig(zipkinServerUrl); |

1. Extract the trace context sent by the WPF client app using the following code.

|  |
| --- |
| //Extract trace information from Http headers, if not found create a new Trace without context.  Trace trace = Utilities.ExtractIncomingTrace(request); |

1. Change the code to return the real trace ID.

|  |
| --- |
| //When tracing, comment out these lines.  //app code  //Guid guidId = Guid.NewGuid();  //byte[] byteId = guidId.ToByteArray();  //long longId = BitConverter.ToInt64(byteId, 0);  long longId = trace.CurrentSpan.TraceId; |

Originally, we had no trace ID to show, so we randomly generated one. Now that we have tracing, we will use the actual trace ID.

1. Record the incoming request from the client by copying the following code under comment.

|  |
| --- |
| //Record sr (Server Receive) event to acknowledge receipt from WPF client.  Utilities.RecordAnnotation(trace, Utilities.ZKAnnotations.SR, "WcfService1", "GET");  Utilities.RecordMessage(trace, content); |

1. Record the outgoing response just before the return statement. Copy the code under the specified comment.

|  |
| --- |
| //Record ss (Server Send) event  Utilities.RecordAnnotation(trace, Utilities.ZKAnnotations.SS); |

1. Type “Ctrl+S” to save the changes in DemoService.svc.cs file.

You have finished instrumenting both the client WPF service and the Service WCF service. Now when you run the application again, you will record trace information.

### Part Two

In this section we will run the application again and see the trace information.

Please follow the following instructions:

Run the application:

1. In Solution Explorer, right-click on the solution “DemoZipkinAppInsights” and select

“Rebuild Solution”.

1. Press “F5” to start the applications – WPF UI, WCF service in browser window should appear.
2. Enter any text input (for example “Bah”) and click button.
3. Again enter text input, this time with “error” in the text string (for example “Bah has error”) and click. This will generate an error we will discover in the trace data.
4. Keep this window open to see the traceId values obtained.

Wait about two minutes. This is the latency of the AppInsights storage system. By paying the price of this short latency, you ensure that communicating with AppInsights storage adds only a tiny overhead to your application. After two minutes, you will be able to see the trace information in the Application Insights and ZIpkin UIs.

**Query trace information in Application Insights**:

1. Open a browser window.
2. Go to the following URL.

|  |
| --- |
| https://analytics.applicationinsights.io/subscriptions/52536069-73d6-41ff-9e48-25215221494e/resourcegroups/ZipkinDemoLabAI/components/ZipkinDemoLabAI#/discover/query/results?apptype=web |

1. After the page loads, create a new query window by pressing “+” next to the “Home Page” tab.
2. Enter the following query in the new query window.

|  |
| --- |
| traces  | where customDimensions['traceid'] == '<TraceId in AI>' |

1. Copy the “TraceId in AI” from the client window and replace <TraceId in AI> in the query window with its value.
2. Press Go.
3. Expand the results to see the information stored in AppInsights regarding your trace.

**View trace information in Zipkin UI:**

1. Open a new tab in the web browser.
2. Go to the following URL.

|  |
| --- |
| http://52.229.21.70:9411/ |

1. Click the “Find Traces” button.
2. You should see recent trace information.
3. To view your specific trace, copy the “Trace Id” value in the client window into the “Go to trace” text box on the top right of the Zipkin UI.
4. Hit enter
5. You should see your specific trace. If the text had error, you will see the WCF service bar in red.
6. You can click on JSON and find the message you entered.

In conclusion, you have learned how to instrument a distributed application using zipkin4net client library and view the trace information in Applciation Insights and in Zipkin UI.

Thank you!