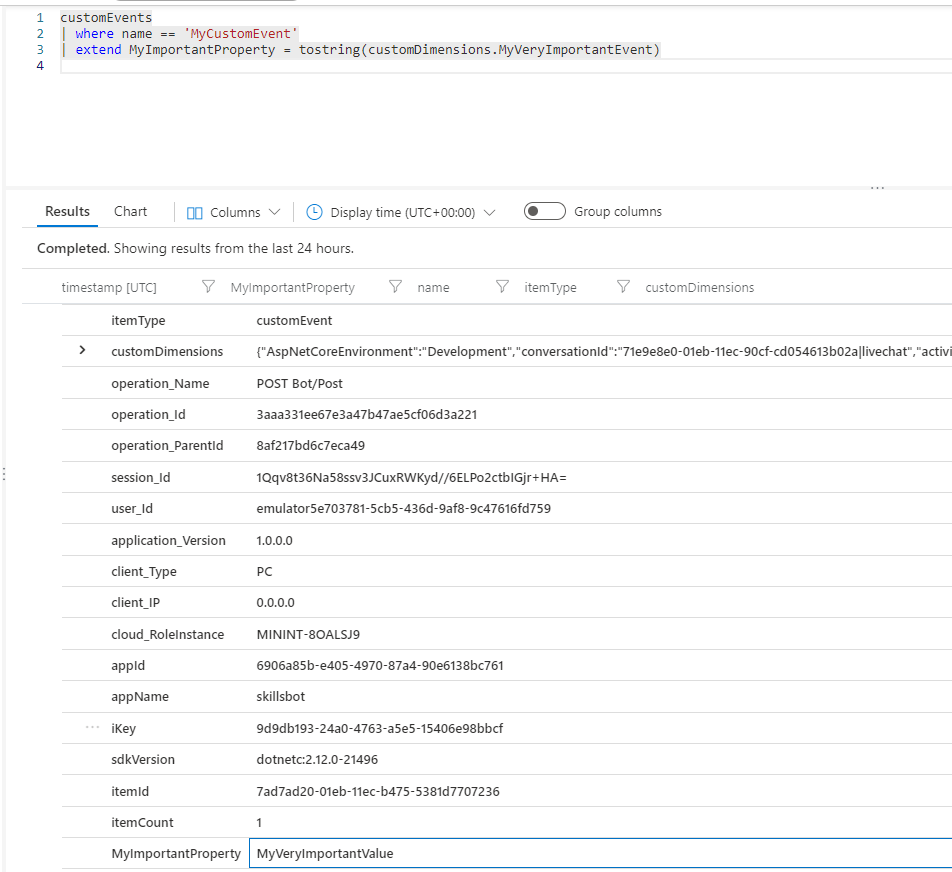
Hi Hiro and Steven,

First, I now fully appreciate your comment during our call about what’s the “right” way to setup telemetry. I’m going to package the information into various sections to make it easier for you to ingest this information dump.

Setting up Telemetry

I think it’s important to talk about what you get out of the box with telemetry. Some of this I’m sure you already know, but I’m going to be slightly pedantic just in case. The App Insights telemetry primarily focuses on metrics around the health of your app. This is primarily focused on calls, responses, and errors. This can vary with your logging level, but this is a reasonable guideline.

As far as I’ve been able to tell it’s pretty good at saying something is going right or wrong. It’s not good at saying why something is going right or wrong. To help remedy this you can add custom events to get better information about what’s happening at each spot in your app. I have an example of this in the **Telemetry**  project on line 50 in **EchoBot.cs**. App Insights allows you to create and log custom events with various payloads to make it easier for you to understand what is (or isn’t) happening on your bot. To make the sample work all you’ll need is your **ApplicationInsights InstrumentationKey**. If you run the bot locally, and enter the message “important” you’ll be able to see that event in App Insights under Custom Events:



You can use this outline to instrument your app at whatever point you so see fit and give yourself whatever logs best meet your needs. Now one other thing you mentioned during our call was the need to be able to reconstruct a conversation based on a conversationID. The easiest way I found to do this was by adding an Azure Table and saving the activities to that table.

**Azure Table**

I found Azure Tables to be the easiest/lightest weight solution to this particular storage problem. There isn’t any reason you couldn’t use a standard database, but I wanted to make it as easy as possible. I should also add maybe this is possible with App Insights, but I think there’s value in having the activity when trying to diagnose issues. If you look at the **Table** project you’ll see the sample for how this works. You can follow the steps in [this](https://docs.microsoft.com/en-us/azure/cosmos-db/table/tutorial-develop-table-dotnet) article to create the table. The only caveat is on step 5 in the **Create an Azure Cosmos DB Table API account** you’ll select the “Table” option from the list of ~5-6 choices since the screenshot is out of date. Once the table is created go to **Data Explorer -> Your Table -> Edit Entity**. Add a string column for **Activity** and add a string column for **Notes**.

**A screenshot of a computer

Description automatically generated**

After that you can skip to the **Configure your storage connection string** and save the connection string and table name somewhere.

If you go to the bot you’ll need to update 2 things. First, you’ll need to update **appsettings.json** with your connection string. Second, you’ll need to update the **\_nameOfTable** variable on line 28 in **TableBot.cs** to be the name of your table. If you look down on line 53 you’ll see the LogActivity function. This function shows one way you could save the activities, based on conversationID to make it very easy to reconstruct some conversation. The **Notes** property could be filled with the ChannelId, or any other information you’d be interested in saving.

Let me share some technical information about other things worth understanding in this sample. The **TableHelper.cs** file has a number of other functions that you might be interested in. I only wired up add since I wanted to make it easy to follow. If you wanted to change the properties of the **ActivityEntity** you can look in the **Model** folder. You could add more properties here. If you do, you’ll need to reflect those changes in the Table’s Entity definition so everything continues to pipe through.

If you wanted to see what the activities look like in Azure here’s an example:

Graphical user interface, text, application

Description automatically generated

In wrapping up, if you want to better instrument your bot for telemetry, I’d recommend adding custom events to give yourself the information you’re looking for. If you want to easily be able to reconstruct any conversation I’d recommend setting up an Azure Table (or DB) and saving your activities there.