Telemetry Guidance

Overview

Arcade projects will emit telemetry metadata that allows us to more easily classify where failures occur. This metadata will follow a prescribed format outlined below. The metadata will be written to Azure DevOp's Timeline API so that it can be gathered later and moved to our Engineering database or otherwise analyzed.

Telemetry format

```
(NETCORE_ENGINEERING_TELEMETRY=[telemetry category]) [Message]
```

Example

```
(NETCORE_ENGINEERING_TELEMETRY=Publish) Publishing failed with
exit code 1.
```

Writing to the Timeline API

Data can be written into Azure DevOp's Timeline API by writing error or warning messages to the console in the prescribed Azure DevOps format.

Example

```
echo "##vso[task.logissue type=error](NETCORE_ENGINEERING_TELEMETRY=Publish)
Publishing failed with exit code 1."
```

Arcade support for writing categorized telemetry

Arcade will provide support for writing telemetry both in scripts (Powershell and bash) and in Arcade's MSBuild logger.

Arcade script support

Arcade scripts will support a `Write-PipelineTelemetryError` function that can be called and will provide properly formatted error message output.

Repos with custom scripts can add telemetry categorization by using Arcade's logging functions which are available in their repo via dependency flow of the `eng/common` scripts.

Please note: for scripts that call the logging functions to properly report back to AzDO, you must set `$ci=$true`.

Arcade MSBuild logger support

Arcade's MSBuild logger looks for an `NETCORE_ENGINEERING_TELEMETRY` static or global property. If present, then error output will be decorated in the expected telemetry format.

This will allow us to add telemetry properties into MSBuild projects which will then decorate the Timeline API results with the expected categorization.

Project properties are useful to set a generic categorization for a project. This model works well for Arcade because Arcade directly invokes MSBuild on a project and we can specify a categorization when that invocation occurs. For dynamic categorization (specifying a category to be set when a target is executing), you can use MSBuilds "Telemetry" task.

Example If we add...

```xml
<PropertyGroup>
  <NETCORE_ENGINEERING_TELEMETRY>Build</NETCORE_ENGINEERING_TELEMETRY>
</PropertyGroup>

<Target Name="Build" />

<Target Name="MyCustomBuildTarget"
        AfterTargets="Build">
    <Telemetry EventName="NETCORE_ENGINEERING_TELEMETRY" EventData="Category=Custom" />
</Target>
```

into a project file, then any CI failures in the "Build" target will be categorized as "Build". Any failure in "MyCustomBuildTarget" will be categorized as "Custom" because of the "Telemetry" task.

Arcade will use a combination of the above techniques to enable categorization for anyone using Arcade.

Logging categories

We are not intending to be proscriptive are hard-lined about a specific set of categories that a repo must use when sending telemetry. Initially, however, we will modify Arcade to categorize "Restore", "Build", "Test", "Sign", and "InitializeToolset" (Arcade script) changes.

Example chart

This is a completely made up example of a possible way we could surface this data.

Category chart

Repos may provide their own categories or use existing reports as examples of good category names.

Was this helpful? 👍 👎