Common concepts

This page contains concepts that may be shared between different implementations of the instrumentation library: purpose, organization, message format, etc. There may be different implementations of the instrumentation library for different targets, e.g. Android, iOS, Unity.

Instrumentation library contents

The instrumentation library will be used by app developers to provide measurement points from within their own apps. These measurement points are necessary to compute some measurements. For instance, “playback started” and “playback finished” are measurement points which are used to compute the “playback memory usage” measurement, and possibly others. Measurements in turn can be used to compute KPIs or other aggregated data.

The measurement points are associated with a particular app feature. For instance, the two measurement points mentioned above would belong to a “media file playback” feature. The features themselves belong to a particular app use case, which define the general category of the app, such as “virtual reality"or “content distribution streaming services”. Some features don’t belong to any particular use case and are associated with a “common” use case.

The package/class hierarchy of the instrumentation library should provide a clear path to the appropriate measuerement, so that the user is able to navigate it to find the appropriate method/function to call. In addition, the relationship between a feature and its measurement points is expected to be clear enough, so that there is no need to add measurements to the hierarchy of the library (which have a many-to-many relationship with measurement points).

The general structure would then be:

- App instrumentation library
    - Use cases
        - Features
            - Measurement points

Instrumentation messages

A call to one of the methods/functions of the instrumentation library will produce a message with all the relevant contents for that measurement point . The way of getting the message from the mobile device to TAP depends on the actual library implementation. The contents and format of the message do not.

Contents

The messages produced by the instrumentation library for any measurement point should have the following contents:

- Timestamp: the time on the device when the message was produced
- Measurement point: which measurement point is reported in the message. Each measurement point is reported in a single message (although a single measurement point may have multiple values). A measurement point is identified together with the use case and feature to which it belongs.
- Value: the value (or values) of the measurement point, as given by the caller

Message format

The instrumentation library will produce messages in text format with a well-defined format. This format shall be flexible enough, so that new messages can be produced without altering the format or breaking the existing parsers.

The general format of the text messages is as follows:

```
<timestamp>\t<use_case>\t<feature>\t<measurement_point>\t<value>
```

Note that \t denotes a tab character.

The variables used in the format are:

- <timestamp>: the timestamp from the message
    - The timestamp should be in the UTC timezone
    - The timestamp is formatted using the following ISO 8601 representation: <date>T<time>, where

        - <date>=YYYY-MM-DD, with YYYY, MM, DD as zero-padded year, month and day, respectively
        - <time>=hh:mm:ss.sss<tz>, with hh, mm, ss, sss as zero-paded hour, minute, second, and millisecond, respectively.
            - <tz> is the timezone indicator, which can either be Z for UTC, or ±[hh] to indicate an offset in hours from UTC
- <use_case>: the "canonical id" of the use case
- <feature>: the "canonical id" of the feature
- <measurement_point>: the "canonical id" of the measurement point
- <value>: the actual value or values reported for the measurement point
    - The actual format of the value depends on the type used in the measurement point

The "canonical ids" of the use case, feature and measurement point must be composed exclusibely of alphanumeric characters and the underscore, starting with a letter character i.e. "[a-zA-Z][a-zA-Z0-9_]*".

The message text is trimmed so that no extra spacing are located before or after the contents.

Value format

A measurement point may have zero or more arguments, which together form the value of the measurement point.

If a measurement point has more than one argument, the <value> contains a tab-separated list of each of the argument values. There shall be no extra spaces around the commas. For instance: 1\ttrue\t2.0.

The format of each argument depends on its type. The following types are supported as arguments of a measurement point:

- boolean
    - true and false values are formatted as true and false, respectively
- int
    - integers are formatted using base 10
- double
    - floating point values are formatted using an IEEE 754 compatible format
- string
    - strings are placed between double quotes ("), e.g. "Hello, World!"
    - quotes inside strings must be escaped with \, e.g. "Hello, \"World\"!"
    - new-line characters should be escaped as well, as \n and \r, e.g. "line1\nline2"

Custom measurement points

The library will provide means to app developers to provide additional measurement points. These measurement points shall be parsed and stored alongside the rest of the measurement points, but they will not be included as part of any standard measurement computation.

To distinguish them from regular measurement points, all these measurement points will be organized into a new use case, called "Custom". It's still expected that the message will contain a feature and measurement point, but their actual values and meaning are left to the user.