Codebook | TIME Study Raw Data and Log File

TIME study website

6/30/23: This codebook is currently being improved as more researchers use the TIME data. If information in the codebook is not clear, or if you feel information is missing, including key variables that might simplify your data analysis, you are encouraged to use the Google Docs comment feature to insert your question(s) or comment(s) directly in this document. Alternatively, you can use the project forum. Thank you for your help! - The TIME Study Team.

Table of Contents

About the TIME Study

The TIME study used real-time mobile technologies to collect intensive longitudinal data examining differences in micro-temporal processes underlying the adoption and maintenance of physical activity, low sedentary time, and sufficient sleep duration.

To learn about TIME Study goals, methods, and data, please begin by reviewing the TIME Study website.

About this Codebook

This codebook includes information about raw data and log files from TIME Study. This document is for **PUBLIC** users. The TIME app records data and logs of each hour into four folders: data, data-watch, logs, logs-watch. The files are organized by participants, dates, and hours.

The first page has a glossary that helps readers direct to the page for particular raw data files. Each file page consists of one summary section and one metadata column section. The summary section has the general description about data source and nature of the data. The metadata column section includes description, format, and value options for each column in the intermediate data file.
This is one of several codebooks for the TIME Study. Many projects will require combining data from multiple codebooks. The TIME Study website has links to all codebooks.
Researchers using these data should ask questions or report problems on the project forum.

About Data Generation

The raw data and log files are recorded by the TIME app installed on smartphones and smartwatches. The original Android app code can be found in this repository (needs internal access). Researchers using these data should ask questions or report problems on the project forum.

Abbreviations and Terminology

Please review the Table of Abbreviations and Special Terminology.

Raw Data and Log Files

Data files (phone)

1. Acceleration.csv
2. ActivityDetected.csv
3. AppEventCounts.csv
4. GPS.csv
5. LightSensorStats.csv
6. NotificationCounts.csv
7. ProximitySensorManagerService.csv
8. StepCounterService.csv
9. AmbientPressManagerService.csv

Data-watch files (watch)

10. Raw watch accelerometer baf file
11. Watch Battery Event File
12. har_activity.csv
13. har_intensity.csv
14. har_posture.csv
15. mims.csv

Logs files (phone)

16. AccelSampling.log.csv
17. AccelerationManagerService.log.csv
18. AlwaysOnService.log.csv
19. AppEventsCountsService.log.csv
20. AppUsageStatsService.log.csv
21. AppsInstalledStats.log.csv
22. BackupMinuteService.log.csv
23. BatteryManager.log.csv
24. ConnectivityManager.log.csv
25. DataTransfer.log.csv
26. DatabaseManager.log.csv
27. DetectedActivityIntentService.log.csv
28. EnvironmentSensorsManager.log.csv
29. FilebaseCleaningService.log.csv
30. LocalAlarmManager.log.csv

31. LocationManagerService.log.csv
32. LocationManagerService2.log.csv
33. MemoryManager.log.csv
34. MicroTUploadManagerService.log.csv
35. MicroTUploadManagerServiceNotes.log.csv
36. MinuteService.log.csv
37. NotificationLoggerService.log.csv
38. NotificationManager.log.csv
39. PermissionManager.log.csv
40. Phone-SurveyManager.log.csv
41. PowerManager.log.csv
42. PromptResponses.log.csv
43. StepCounterService.log.csv
44. StorageManager.log.csv
45. SuperBurstScheduleManager.log.csv
46. SurveyManager.log.csv
47. SystemBroadcastReceiver.log.csv
48. UploadManager-upload.err.log
49. UploadManager-upload.log.csv
50. Util.log.csv
51. WatchUploadManagerService.log.csv
52. WearDataListenerService.log.csv
53. WearNoteSender.log.csv
54. Zipper.log.csv

Logs-watch files (watch)

55. Watch-AccelSampling.log.csv
56. Watch-DozeAlarmManager.log.csv
57. Watch-FileProcessingService.log.csv
58. Watch-MiscLogger.log.csv
59. Watch-NotificationsHandler.log.csv
60. Watch-PhoneCommunicator.log.csv
61. Watch-PowerConnectionReceiver.log.csv
62. Watch-PowerConnectionReceiverNotes.log.csv
63. Watch-PromptScheduler.log.csv
64. Watch-SamplingRate.log.csv
65. Watch-SensorManagerService.log.csv
66. Watch-TimeStudyArbitrater.log.csv

67. Watch-TimeStudyWearListenerService.log.csv
68. Watch-TimeStudyWearListenerServiceNotes.log.csv
69. Watch-UtilityMethods.log.csv
70. Watch-WWAlwaysOnService.log.csv
71. Watch-WearListenerService.log.csv
72. Watch-WearableMessageTransferService.log.csv
73. Watch-WearableWakefulService.log.

Acceleration.csv

Summary

This data file is created by the class AccelerationManagerService to record raw accelerometer readings from motion sensors on smartphones. The sampling rate is 50 readings per second. For more details about the data source, please read the motion sensor TYPE_ACCELEROMETER used in Android, and the description of acceleration data can be found in the developer documentation.

Metadata Columns

[Col 0] - Logging Time

Descriptions: Use differential encoding with timestamp. The number of the first row is the logging timestamp and the value of the following row is milliseconds difference from the first row.

Format: Timestamp for the first row and integer for the following rows.

[Col 1] - X_ACCELERATION_METERS_PER_SECOND_SQUARED

Descriptions: Acceleration force along the x axis (including gravity) with the unit of measure being m/s2, divided by 1 g (gravitational acceleration, 9.8 m/s2)

Format: Float

[Col 2] - Y_ACCELERATION_METERS_PER_SECOND_SQUARED

Descriptions: Acceleration force along the y axis (including gravity) with the unit of measure being m/s2, divided by 1 g (gravitational acceleration, 9.8 m/s2)

Format: Float

[Col 3] - Z_ACCELERATION_METERS_PER_SECOND_SQUARED

Descriptions: Acceleration force along the z axis (including gravity) with the unit of measure being m/s2, divided by 1 g (gravitational acceleration, 9.8 m/s2)

Format: Float

Note

ActivityDetected.csv

Summary

This data file is created by the class DetectedActivityIntentService to record activity updates from Google's Activity Recognition Sampling API. The confidence level is a value from 0 to 100 indicating the likelihood that the user is performing the activity.

The activities are detected by periodically waking up the device and reading **short bursts** of sensor data. It only makes use of low power sensors in order to keep the power usage to a minimum. For example, it can detect if the user is currently on foot, in a car, on a bicycle or still. See DetectedActivity for more details.

The detailed description for all detected activities and confidence level can be found in Google's online documentation.

Metadata Columns

[Col 0] - Writing Time

Descriptions: Writing time

Format: String

[Col 1] - Confidence Level of IN_VEHICLE

Descriptions: the confidence level of the device is in a vehicle, such as a car.

Format: 0-100

[Col 2] - Confidence Level of ON_BICYCLE

Descriptions: the confidence level of the device is on a bicycle.

Format: 0-100

[Col 3] - Confidence Level of ON_FOOT

Descriptions: the confidence level of the device is on a user who is walking or running.

Format: 0-100

[Col 4] - Confidence Level of RUNNING

Descriptions: the confidence level of the device is on a user who is running.

Format: 0-100

[Col 5] - Confidence Level of STILL

Descriptions: the confidence level of the device is still (not moving).

Format: 0-100

[Col 6] - Confidence Level of TILTING

Descriptions: the confidence level of the device angle relative to gravity changed significantly.

Format: 0-100

[Col 7] - Confidence Level of WALKING

Descriptions: the confidence level of the device is on a user who is walking.

Format: 0-100

[Col 8] - Confidence Level of UNKNOWN

Descriptions: the confidence level of unable to detect the current activity.

Format: 0-100

Notes

- Missing data related issues
    - In our TIME app, we controlled the detectionIntervalMillis parameter to be Long.MAX_VALUE, which means the app only monitors the results requested by other clients without consuming additional power.
    - “To conserve battery, activity reporting may stop when the device is 'STILL' for an extended period of time. It will resume once the device moves again. This only happens on devices that support the Sensor.TYPE_SIGNIFICANT_MOTION hardware.”

AppEventCounts.csv

Summary

This data file is created by the class AppEventsCountsService to record app usage events for the past hour on smartphones. For more details about the app usage events, please check the Android documentation.

Metadata Columns

[Col 0] - Logging Time

Descriptions: logging time (human readable)

Format: string

[Col 1] - Logging Time less one hour

Descriptions: Logging Time less one hour

Format: string

[Col 2] - Logging Time

Descriptions: logging time (timestamp)

Format: string

[Col 3] - Package Name

Descriptions: the package name of the source of this event.

Format: string

[Col 3] - Usage Event Timestamp

Descriptions: timestamp of this event.

Format: string

[Col 3] - Event Type

Descriptions: event type. For more details about the app usage events, please check the Android documentation.

Format: string

Notes

GPS.csv

Summary

This data file is created by the class LocationManagerService to record smartphone location data every minute.

Metadata Columns

[Col 0] - Logging Timestamp

Descriptions: logging timestamp

Format: string

[Col 1] - Detection Timestamp

Descriptions: the timestamp of location being detected

Format: string

[Col 2] - Latitude

Descriptions: Latitude, in degrees

Format: numerical

[Col 3] - Longitude

Descriptions: longitude, in degrees

Format: numerical

[Col 4] - Accuracy

Descriptions: estimated horizontal accuracy of this location, radial, in meters. We define horizontal accuracy as the radius of 68% confidence. In other words, if you draw a circle centered at this location's latitude and longitude, and with a radius equal to the accuracy, then there is a 68% probability that the true location is inside the circle. This accuracy estimation is only concerned with horizontal accuracy, and does not indicate the accuracy of bearing, velocity or altitude if those are included in this Location. If this location does not have a horizontal accuracy, then 0.0 is returned.

Format: numerical

[Col 5] - Location Provider

Descriptions: the name of the provider that generated this fix

Format: string

[Col 6] - Speed

Descriptions: the speed if it is available, in meters/second over ground. If this location does not have a speed then 0.0 is returned.

Format: numerical

[Col 7] - Altitude

Descriptions: the altitude if available, in meters above the WGS 84 reference ellipsoid. If this location does not have an altitude then 0.0 is returned.

Format: numerical

[Col 8] - Bearing

Descriptions: the bearing, in degrees. Bearing is the horizontal direction of travel of this device, and is not related to the device orientation. It is guaranteed to be in the range (0.0, 360.0] if the device has a bearing. If this location does not have a bearing then 0.0 is returned.

Format: numerical

Notes

LightSensorStats.csv

Summary

This data file is created by the class EnvironmentSensorsManager to record light sensor data on smartphones. For more details about the sensor data, please check the Android documentation.

Metadata Columns

[Col 0] - Logging Time

Descriptions: logging time

Format: string

[Col 1] - Raw Sensor Value

Descriptions: Ambient light level in SI lux units

Format: numerical

[Col 2] - Maximum Sensor Value

Descriptions: maximum range of the sensor in the sensor's unit

Format: numerical

Notes

NotificationCounts.csv

Summary

This data file is created by the class NotificationLoggerService to record notification packages and priority on smartphones.

Metadata Columns

[Col 0] - Logging Time

Descriptions: logging time

Format: string

[Col 1] - Post Time

Descriptions: the time the notification was posted

Format: timestamp

[Col 2] - Notification ID

Descriptions: notification ID

Format: numerical

[Col 3] - Package Name

Descriptions: The package that the notification belong to

Format: string

[Col 4] - Notification ID

Descriptions: Relative priority for this notification

Format: string

Notes

ProximitySensorManagerService.csv

Summary

This data file is created by the class EnvironmentSensorsManager to record proximity sensor data on smartphones. For more details about the sensor data, please check the Android documentation.

Metadata Columns

[Col 0] - Logging Time

Descriptions: logging time

Format: string

[Col 1] - Raw Sensor Value

Descriptions: Proximity sensor distance measured in centimeters. Some proximity sensors only support a binary near or far measurement. In this case, the sensor should report its maximum range value in the far state and a lesser value in the near state.

Format: numerical

[Col 2] - Maximum Sensor Value

Descriptions: maximum range of the sensor in the sensor's unit

Format: numerical

Notes

StepCounterService.csv

Summary

This data file is created by the class UploadManagerService to record step count tracked by smartphone sensors.

Metadata Columns

[Col 0] - Logging Time

Descriptions: logging time

Format: string

[Col 1] - Steps in This Hour

Descriptions: steps in this hour.

Format: numerical

[Col 2] - Steps

Descriptions: Steps since last reboot

Format: numerical

Notes

AmbientPressManagerService.csv

Summary

This data file is created by the class EnvironmentSensorsManager to record pressure sensor data on smartphones. For more details about the sensor data, please check the Android documentation.

Metadata Columns

[Col 0] - Logging Time

Descriptions: logging time

Format: string

[Col 1] - Raw Sensor Value

Descriptions: Atmospheric pressure in hPa (millibar)

Format: numerical

[Col 2] - Maximum Sensor Value

Descriptions: maximum range of the sensor in the sensor's unit

Format: numerical

Notes

Raw Watch Accelerometer Baf File

Summary

The original watch accelerometer data is stored in the data-watch folder on the cluster (AndroidWearWatch-AccelerationCalibrated-NA.020000000000-AccelerationCalibrated.<date_time>.sensor.baf). The script to decrypt and read this file can be found in the bitbucket repo (lines 20-53). The decrypted data frame contains a single line per sensor reading from the watch accelerometer, including logging timestamp and acceleration on xyz dimensions. For more details about the data source, please read the motion sensor TYPE_ACCELEROMETER used in Android, and the description of acceleration data can be found in the developer documentation.

Metadata Columns

[Col 0] - HEADER_TIME_STAMP

Descriptions: logging time of readings from sensor.

Format: datetime object

[Col 1] - X_ACCELERATION_METERS_PER_SECOND_SQUARED

Descriptions: Acceleration force along the x axis (including gravity) with the unit of measure being m/s2, divided by 1 g (gravitational acceleration, 9.8 m/s2)

Format: Float

[Col 2] - Y_ACCELERATION_METERS_PER_SECOND_SQUARED

Descriptions: Acceleration force along the y axis (including gravity) with the unit of measure being m/s2, divided by 1 g (gravitational acceleration, 9.8 m/s2)

Format: Float

[Col 3] - Z_ACCELERATION_METERS_PER_SECOND_SQUARED

Descriptions: Acceleration force along the z axis (including gravity) with the unit of measure being m/s2, divided by 1 g (gravitational acceleration, 9.8 m/s2)

Format: Float

Notes

1. There are several types of situations that have caused missing data:
    a. If the watch battery level is low (below 10%), no logs would be recorded.
    b. If the watch memory is full, no logs would be recorded.
    c. Sometimes the sensor got unregistered

   d. a third party app, loss of connection followed by data backlog, or just google play bloatware

2. A bug before July 2020 where the watch data transfer missed some sensor files. This has been resolved in some late July or early August release.

3. A bug in the Fossil Android code was found around April 2022 that led to the watch having the completely wrong time and date for some dates. With the wrong system time, watch accelerometer data was written to the folder of a past date. There are two situations that have been causing the wrong system time on the watch:

   a. "Automatic date & time" was off or disconnection between watch and phone. This leads to the watch not being able to sync time with the phone.

   b. The time-only mode is turned on. Smartwatches using Wear OS will activate time-only mode automatically when the battery is low.

   This bug was replicated by one researcher: when the watch battery drained and it went into TIME-only mode and then eventually died (not connected, with auto date/time off) and then was charged, it displayed a completely wrong time AND date (in the past). It continued to have this time until it was able to connect with the phone after the auto date/time was switched back on.

   Until May 1st 2022, this bug has impacted data from 28 participants, about 700-100 hours across 86 days. Since there is no way to recover the correct system time, and the watch logs (including prompt responses) were corrupted, data from the impacted dates have been excluded from the dataset.

4. Data corruption.
   a. The code to save the watch accelerometry data in binary format had an error, improperly converting the raw accelerometer value for the x axis into bytes. In byte representation, a floating-point number is represented by four bytes in a specific order. The bytes should have been stored in order as byte[0], byte[1], byte[2] and byte[3]. However, the raw accelerometer value on the x axis was stored as byte[0], byte[1], byte[3], byte[1]. The fourth byte was saved to the wrong position, and the third byte was never properly recorded.
   b. Because the third byte was never recorded, we cannot simply resave the bytes in the proper order. byte[2] and byte[3] contain the least significant bits of the four-byte floating point number. This is why the error was never caught when visualizing data. Real-world experiments showed that **the impact of the error is very small (hour-level aggregation difference (MIMS-units) < 0.2%)**. In fact, the error was only caught when the code was being reviewed to be incorporated into a new project. It was never apparent from data visualization of the raw accelerometer output, which occurred on a regular basis.
   c. More details for this issue can be found in Understanding the Impact of Sensor Bug .

Watch Battery Event File

Summary

The watch battery event file is created by WearableWakefulService. It is stored in the data-watch folder with the name of Battery.020000000000-Battery.<date_time>.event.csv.

Metadata Columns

[Col 0] - HEADER_TIME_STAMP

Descriptions: timestamp of creating the header, should be later than start time

Format: string

[Col 1] - START_TIME

Descriptions: logging time

Format: string

[Col 2] - STOP_TIME

Descriptions: no data was not logged, should always be null.

Format: null

[Col 3] - BATTERY_LEVEL

Descriptions:

Format: numerical

[Col 4] - BATTERY_CHARGING

Descriptions:

Format: boolean

Notes

har_activity.csv

Summary

This file records activity classification results computed by the MUSS algorithm on the raw watch accelerometer data. Please find more details about this file in the intermediate data documentation.

har_intensity.csv

Summary

This file records activity intensity results computed by the MUSS algorithm on the raw watch accelerometer data. Please find more details about this file in the intermediate data documentation.

har_posture.csv

Summary

This file records posture results computed by the MUSS algorithm on the raw watch accelerometer data. Please find more details about this file in the intermediate data documentation.

mims.csv

Summary

This file records mims values computed on the raw watch accelerometer data. Please find more details about this file in the intermediate data documentation.

AccelSampling.log.csv

Summary

This data file is created by the class AccelerationManagerService to record 10-second-window area under the curve values for the three axises computed from raw accel data collected from smartphones.

Metadata Columns

[Col 0] - Logging Time

Descriptions: logging time

Format: String

[Col 1] - Logging Type

Descriptions: logging type

Format: String

[Col 2] - Participant's Visualizer ID

Descriptions: Participant's visualizer ID

Format: String

[Col 3] - Class Name

Descriptions: class name

Format: String

[Col 4] - Area Under the Curve Value

Descriptions: logs of Area Under the Curve Value

Format: String

Value: timestamp, x-axis area under the curve, y-axis area under the curve, z-axis area under the curve

Notes

AccelerationManagerService.log.csv

Summary

This file records logs from the class AccelerationManagerService. It is originally created under /.microt/logs/ on smartphone. The class DetectedActivityIntentService is called by MinuteService => ActivityDetectionService. The raw accel value is written in Acceleration.csv. The values of area under the curve are computed and write the results in AccelSampling.csv.

Metadata Columns

[Col 0] - Logging Time

Descriptions: logging time

Format: String

[Col 1] - Logging Type

Descriptions: logging type

Format: String

[Col 2] - Participant's Visualizer ID

Descriptions: Participant's visualizer ID

Format: String

[Col 3] - Class Name

Descriptions: class name

Format: String

[Col 4] - Logs

Descriptions: class logs

Format: String

Value:

- "File is created!" means a new Acceleration.csv is created for the current hour on device under /.MICROT/data/;
- "File already exists!" means the Acceleration.csv file has been created already;
- "acctualPoints: " means the number of accelerometer data points actually collected in the current hour. Ideally there should be 180,000 data points collected each hour.

- “expectedPoints: " means the expected number of accelerometer data points should have been collected so far for the current hour.
- "dataLossPercent: " means the percentage of data loss of accelerometer data points so far for the current hour.
-

Notes

AlwaysOnService.log.csv

Summary

This file records logs from the class AlwaysOnService. This class register broadcast receivers and starts minute services. The system broadcast intents are recorded in SystemBroadcastReceiver.log.csv.

Metadata Columns

[Col 0] - Logging Time

Descriptions: logging time

Format: String

[Col 1] - Logging Type

Descriptions: logging type

Format: String

[Col 2] - Participant's Visualizer ID

Descriptions: Participant's visualizer ID

Format: String

[Col 3] - Class Name

Descriptions: class name

Format: String

[Col 4] - Logs

Descriptions: class logs

Format: String

Value: logs of registering broadcast receivers

Notes

AppEventsCountsService.log.csv

Summary

This file records logs from the class AppEventsCountsService. It is originally created under /.microt/logs/ on smartphone. The class AppEventsCountsService is called by MinuteService. The app usage events are written in AppEventCounts.csv.

Metadata Columns

[Col 0] - Logging Time

Descriptions: logging time

Format: String

[Col 1] - Logging Type

Descriptions: logging type

Format: String

[Col 2] - Participant's Visualizer ID

Descriptions: Participant's visualizer ID

Format: String

[Col 3] - Class Name

Descriptions: class name

Format: String

[Col 4] - Logs

Descriptions: class logs

Format: String

Notes

AppUsageStatsService.log.csv

Summary

This file records logs from the class AppUsageStatsService. It is originally created under /.microt/logs/ on smartphone. The class AppUsageStatsService is called by MinuteService. Specifically, the service calls `getTotalTimeInForeground` to get the total time this package spent in the foreground (When in the foreground, the user is actively interacting with the app). The total minutes of the app being in foreground for the past day are written in AppUsageStats.csv.

Metadata Columns

[Col 0] - Logging Time

Descriptions: logging time

Format: String

[Col 1] - Logging Type

Descriptions: logging type

Format: String

[Col 2] - Participant's Visualizer ID

Descriptions: Participant's visualizer ID

Format: String

[Col 3] - Class Name

Descriptions: class name

Format: String

[Col 4] - Logs

Descriptions: class logs

Format: String

Notes

AppsInstalledStats.log.csv

Summary

This file records logs from the class AppsInstalledStats. It is originally created under /.microt/logs/ on smartphone. The class AppsInstalledStats is called by MinuteService. The information about the installed apps are written in AppsInstalledStats.csv.

Metadata Columns

[Col 0] - Logging Time

Descriptions: logging time

Format: String

[Col 1] - Logging Type

Descriptions: logging type

Format: String

[Col 2] - Participant’s Visualizer ID

Descriptions: Participant’s visualizer ID

Format: String

[Col 3] - Class Name

Descriptions: class name

Format: String

[Col 4] - Logs

Descriptions: class logs

Format: String

Notes

BackupMinuteService.log.csv

Summary

This file records logs from the class BackupMinuteService. It is originally created under /.microt/logs/ on smartphone. The class BackupMinuteService is called by MinuteService to check if the current thread is the main thread.

Metadata Columns

[Col 0] - Logging Time

Descriptions: logging time

Format: String

[Col 1] - Logging Type

Descriptions: logging type

Format: String

[Col 2] - Participant's Visualizer ID

Descriptions: Participant's visualizer ID

Format: String

[Col 3] - Class Name

Descriptions: class name

Format: String

[Col 4] - Logs

Descriptions: class logs

Format: String

Notes

BatteryManager.log.csv

Summary

This file records logs from the class BatteryManager. It is originally created under /.microt/logs/ on smartphone. The class BackupMinuteService is called by MinuteService to check the battery level, charging status, and power source in charging.

Metadata Columns

[Col 0] - Logging Time

Descriptions: logging time

Format: String

[Col 1] - Logging Type

Descriptions: logging type

Format: String

[Col 2] - Participant's Visualizer ID

Descriptions: Participant's visualizer ID

Format: String

[Col 3] - Class Name

Descriptions: class name

Format: String

[Col 4] - Logs

Descriptions: smartphone battery level, charging status, power source (AC charger, USB port, wireless)

Format: String

Notes

ConnectivityManager.log.csv

Summary

This file records logs from the class ConnectivityManager. It is originally created under /.microt/logs/ on smartphone. The class ConnectivityManager called by MinuteService to check data connection.

Metadata Columns

[Col 0] - Logging Time

Descriptions: logging time

Format: String

[Col 1] - Logging Type

Descriptions: logging type

Format: String

[Col 2] - Participant's Visualizer ID

Descriptions: Participant's visualizer ID

Format: String

[Col 3] - Class Name

Descriptions: class name

Format: String

[Col 4] - Logs

Descriptions: data connection status, data connection (wifi, mobile)

Format: String

Notes

DataTransfer.log.csv

Summary

This file records logs from the class DataTransfer. It is originally created under /.microt/logs/ on smartphone. The class DataTransfer is called to check watch connection status.

Metadata Columns

[Col 0] - Logging Time

Descriptions: logging time

Format: String

[Col 1] - Logging Type

Descriptions: logging type

Format: String

[Col 2] - Logs

Descriptions: watch connection status

Format: String

Notes

DatabaseManager.log.csv

Summary

This file records logs from the class DatabaseManager. It is originally created under /.microt/logs/ on smartphone. The class DatabaseManager is called to fetch data and write notes on firebase.

Metadata Columns

[Col 0] - Logging Time

Descriptions: logging time

Format: String

[Col 1] - Logging Type

Descriptions: logging type

Format: String

[Col 2] - Participant's Visualizer ID

Descriptions: Participant's visualizer ID

Format: String

[Col 3] - Class Name

Descriptions: class name

Format: String

[Col 4] - Logs

Descriptions: logs

Format: String

Notes

DetectedActivityIntentService.log.csv

Summary

This file records logs from the class DetectedActivityIntentService. It is originally created under /.microt/logs/ on smartphone. The class DetectedActivityIntentService is called by MinuteService => ActivityDetectionService to request activity updates and write results in ActivityDetected.csv.

Metadata Columns

[Col 0] - Logging Time

Descriptions: logging time

Format: String

[Col 1] - Logging Type

Descriptions: logging type

Format: String

[Col 2] - Participant's Visualizer ID

Descriptions: Participant's visualizer ID

Format: String

[Col 3] - Class Name

Descriptions: class name

Format: String

[Col 4] - ActivityDetected.csv File Exit or Not

Descriptions: Check if the ActivityDetected.csv has been created for the current hour on device under /.microt/data/.

Format: String

Value: "File is created!" means a new file is created; "File already exists!" means this file has been created already.

Notes

EnvironmentSensorsManager.log.csv

Summary

This file records logs from the class EnvironmentSensorsManager. It is originally created under /.microt/logs/ on smartphone. The class EnvironmentSensorsManager is called by MinuteService to request ambient sensor data and write results in LightSensorStats.csv, ProximitySensorManagerService.csv, AmbientPressManagerService.csv, AmbientTempManagerService.csv, and AmbientHumidManagerService.csv.**Metadata Columns**

[Col 0] - Logging Time

Descriptions: logging time

Format: String

[Col 1] - Logging Type

Descriptions: logging type

Format: String

[Col 2] - Participant's Visualizer ID

Descriptions: Participant's visualizer ID

Format: String

[Col 3] - Class Name

Descriptions: class name

Format: String

[Col 4] - Logs

Descriptions: logs

Format: String

Notes

FilebaseCleaningService.log.csv

Summary

This file records logs from the class FilebaseCleaningService. It is originally created under /.microt/logs/ on smartphone. The class FilebaseCleaningService is called every day to delete data and log files that are created for more than two weeks and have been uploaded to Firebase.

Metadata Columns

[Col 0] - Logging Time

Descriptions: logging time

Format: String

[Col 1] - Logging Type

Descriptions: logging type

Format: String

[Col 2] - Participant's Visualizer ID

Descriptions: Participant's visualizer ID

Format: String

[Col 3] - Class Name

Descriptions: class name

Format: String

[Col 4] - Logs

Descriptions: logs

Format: String

Notes

LocalAlarmManager.log.csv

Summary

This file records logs from the class LocalAlarmManager. It is originally created under /.microt/logs/ on smartphone. The class LocalAlarmManager is called by AlwaysOnService to set prompt alarm.

Metadata Columns

[Col 0] - Logging Time

Descriptions: logging time

Format: String

[Col 1] - Logging Type

Descriptions: logging type

Format: String

[Col 2] - Participant's Visualizer ID

Descriptions: Participant's visualizer ID

Format: String

[Col 3] - Class Name

Descriptions: class name

Format: String

[Col 4] - Logs

Descriptions: logs

Format: String

Notes

LocationManagerService.log.csv

Summary

This file records logs from the class LocationManagerService. It is originally created under /.microt/logs/ on smartphone. The class LocationManagerService is called by MinuteService to request location data every minute. The logs record information about the connection status to Google location API and time out. The raw location data are written in GPS.csv. Find more details about the Google Location API in the documentation.

Metadata Columns

[Col 0] - Logging Time

Descriptions: logging time

Format: String

[Col 1] - Logging Type

Descriptions: logging type

Format: String

[Col 2] - Participant's Visualizer ID

Descriptions: Participant's visualizer ID

Format: String

[Col 3] - Class Name

Descriptions: class name

Format: String

[Col 4] - Logs

Descriptions: logs

Format: String

Notes

LocationManagerService2.log.csv

Summary

This file records logs from the class LocationManagerService2. It is originally created under /.microt/logs/ on smartphone. Similar to LocationManagerService, the class LocationManagerService2 is called by MinuteService to request location data with longer time out limit when the last location update is 15 minutes away.

Metadata Columns

[Col 0] - Logging Time

Descriptions: logging time

Format: String

[Col 1] - Logging Type

Descriptions: logging type

Format: String

[Col 2] - Participant's Visualizer ID

Descriptions: Participant's visualizer ID

Format: String

[Col 3] - Class Name

Descriptions: class name

Format: String

[Col 4] - Logs

Descriptions: logs

Format: String

Notes

MemoryManager.log.csv

Summary

This file records logs from the class MemoryManager. It is originally created under /.microt/logs/ on smartphone. The class MemoryManager is called by MinuteService to record smartphone memory usage for the current moment.

Metadata Columns

[Col 0] - Logging Time

Descriptions: logging time

Format: String

[Col 1] - Logging Type

Descriptions: logging type

Format: String

[Col 2] - Participant's Visualizer ID

Descriptions: Participant's visualizer ID

Format: String

[Col 3] - Class Name

Descriptions: class name

Format: String

[Col 4] - Logs

Descriptions: logs

Format: String

Notes

MicroTUploadManagerService.log.csv

Summary

This file records logs from the class UploadManagerService. It is originally created under /.microt/logs/ on smartphone. The class UploadManagerService is called by MinuteService to zip and upload data and log files.

Metadata Columns

[Col 0] - Logging Time

Descriptions: logging time

Format: String

[Col 1] - Logging Type

Descriptions: logging type

Format: String

[Col 2] - Participant's Visualizer ID

Descriptions: Participant's visualizer ID

Format: String

[Col 3] - Class Name

Descriptions: class name

Format: String

[Col 4] - Logs

Descriptions: logs

Format: String

Notes

MicroTUploadManagerServiceNotes.log.csv

Summary

This file records logs from the class UploadManagerService. It is originally created under /.microt/logs/ on smartphone. In addition to MicroTUploadManagerService.log.csv, the class UploadManagerService is also called by MinuteService to record DND status, audio profile, study start and end date, next wake and sleep time, current wake time, and EOD completion status.

Metadata Columns

[Col 0] - Logging Time

Descriptions: logging time

Format: String

[Col 1] - Logging Type

Descriptions: logging type

Format: String

[Col 2] - Participant's Visualizer ID

Descriptions: Participant's visualizer ID

Format: String

[Col 3] - Class Name

Descriptions: class name

Format: String

[Col 4] - Logs

Descriptions: logs

Format: String

Notes

MinuteService.log.csv

Summary

This file records logs from the class MinuteService. It is originally created under /.microt/logs/ on smartphone. The class MinuteService is also called by AlwaysOnService to record all kind of information for every minute, sending notifications, schedule prompting, and start services.

Metadata Columns

[Col 0] - Logging Time

Descriptions: logging time

Format: String

[Col 1] - Logging Type

Descriptions: logging type

Format: String

[Col 2] - Participant’s Visualizer ID

Descriptions: Participant’s visualizer ID

Format: String

[Col 3] - Class Name

Descriptions: class name

Format: String

[Col 4] - Logs

Descriptions: logs

Format: String

Notes

NotificationLoggerService.log.csv

Summary

This file records logs from the class NotificationLoggerService. It is originally created under /.microt/logs/ on smartphone. The class NotificationLoggerService is called by MinuteService. The package name of the notification and notification priority is written in NotificationCounts.csv.

Metadata Columns

[Col 0] - Logging Time

Descriptions: logging time

Format: String

[Col 1] - Logging Type

Descriptions: logging type

Format: String

[Col 2] - Participant's Visualizer ID

Descriptions: Participant's visualizer ID

Format: String

[Col 3] - Class Name

Descriptions: class name

Format: String

[Col 4] - Logs

Descriptions: logs

Format: String

Notes

NotificationManager.log.csv

Summary

This file records logs from the class NotificationManager. It is originally created under /.microt/logs/ on smartphones. The class NotificationManager is called by SurveyManager to send system status notifications and survey notifications to the user's device.

Metadata Columns

[Col 0] - Logging Time

Descriptions: logging time

Format: String

[Col 1] - Logging Type

Descriptions: logging type

Format: String

[Col 2] - Participant's Visualizer ID

Descriptions: Participant's visualizer ID

Format: String

[Col 3] - Class Name

Descriptions: class name

Format: String

[Col 4] - Logs

Descriptions: logs

Format: String

Notes

PermissionManager.log.csv

Summary

This file records logs from the class PermissionManager. It is originally created under /.microt/logs/ on smartphone. The class PermissionManager is called by SetupActivity => WocketsActivity to request permissions from user.

Metadata Columns

[Col 0] - Logging Time

Descriptions: logging time

Format: String

[Col 1] - Logging Type

Descriptions: logging type

Format: String

[Col 2] - Participant's Visualizer ID

Descriptions: Participant's visualizer ID

Format: String

[Col 3] - Class Name

Descriptions: class name

Format: String

[Col 4] - Logs

Descriptions: logs

Format: String

Notes

Phone-SurveyManager.log.csv

Summary

This file records logs from the class SurveyManager. It is originally created under /.microt/logs/ on smartphone. The class SurveyManager is called by MinuteService to launch scheduled surveys.

Metadata Columns

[Col 0] - Logging Time

Descriptions: logging time

Format: String

[Col 1] - Logging Type

Descriptions: logging type

Format: String

[Col 2] - Participant’s Visualizer ID

Descriptions: Participant’s visualizer ID

Format: String

[Col 3] - Class Name

Descriptions: class name

Format: String

[Col 4] - Logs

Descriptions: logs

Format: String

Notes

PowerManager.log.csv

Summary

This file records logs from the class SurveyManager. It is originally created under /.microt/logs/ on smartphone. The class SurveyManager is called by MinuteService to check if the device is in idle mode, power save mode, interactive, and sustained performance mode. Please find more information about the mode in Android's documentation.

Metadata Columns

[Col 0] - Logging Time

Descriptions: logging time

Format: String

[Col 1] - Logging Type

Descriptions: logging type

Format: String

[Col 2] - Participant's Visualizer ID

Descriptions: Participant's visualizer ID

Format: String

[Col 3] - Class Name

Descriptions: class name

Format: String

[Col 4] - Logs

Descriptions: logs

Format: String

Notes

PromptResponses.log.csv

Summary

This file records metadata about survey prompting and survey responses. The intermediate file “phone_promptresponse_clean_<date>.csv” adds a header to the original PromptResponses.log.csv. Please find details for each column in the documentation of the intermediate file.

StepCounterService.log.csv

Summary

This file records logs from the class StepCounterService. It is originally created under /.microt/logs/ on smartphone. The class StepCounterService is called by MinuteService to record steps taken by the user. This service uses TYPE_STEP_COUNTER to keep track of the steps.

Metadata Columns

[Col 0] - Logging Time

Descriptions: logging time

Format: String

[Col 1] - Logging Type

Descriptions: logging type

Format: String

[Col 2] - Participant's Visualizer ID

Descriptions: Participant's visualizer ID

Format: String

[Col 3] - Class Name

Descriptions: class name

Format: String

[Col 4] - Logs

Descriptions: logs

Format: String

Notes

StorageManager.log.csv

Summary

This file records logs from the class StorageManager. It is originally created under /.microt/logs/ on smartphone. The class StorageManager is called by MinuteService to record storage status of the file system (external and internal storage) on the user's smartphone. The unit of storage is megabyte (MB).

Metadata Columns

[Col 0] - Logging Time

Descriptions: logging time

Format: String

[Col 1] - Logging Type

Descriptions: logging type

Format: String

[Col 2] - Participant's Visualizer ID

Descriptions: Participant's visualizer ID

Format: String

[Col 3] - Class Name

Descriptions: class name

Format: String

[Col 4] - Logs

Descriptions: logs

Format: String

Notes

SuperBurstScheduleManager.log.csv

Summary

This file records logs from the class SuperBurstScheduleManager. It is originally created under /.microt/logs/ on smartphone. The class SuperBurstScheduleManager is called by MinuteService to schedule the burst mode periods. There are 2 burst modes every 30 days. Each burst mode is 4 days long. Snozzing, before each burst mode the user has the option to snooze the burst by 2 days, only once. Variables given,

- days_bw_bursts: Days between burst mode, this variable defines the days between the 2 bursts modes in a month, the minimum value of days_bw_bursts is 3.
- days_before_first_burst: No. of days before the first burst mode in a month.
- burst_enabled: Bolean to check if burst mode is enabled.
- remaining_days_after_burst: this will be automatically calculated.

The sum of days_before_first_burst+4+days_bw_bursts+4+remaining_days_after_burst = 30 always, cannot be more or less, has to be taken care of. These variables will be stored in the DataManager. Burst mode is from 4 AM to 4 AM.

Metadata Columns

[Col 0] - Logging Time

Descriptions: logging time

Format: String

[Col 1] - Logging Type

Descriptions: logging type

Format: String

[Col 2] - Participant's Visualizer ID

Descriptions: Participant's visualizer ID

Format: String

[Col 3] - Class Name

Descriptions: class name

Format: String

[Col 4] - Logs

Descriptions: today is burst or not and yesterday is burst or not

Format: String

Values:

- "Today is not burst" means today is not burst day;
- "Today is not burst but yesterday was" means today is not burst but yesterday was;
- "Today is not burst and yesterday was and before 4 AM" means today is not burst but yesterday was and the the current time is before 4 AM;
- "Today is not burst and yesterday was and not before 4 AM" means today is not burst but yesterday was and the the current time is after 4 AM;
- "Today is burst and yesterday was not burst" means today is burst and yesterday was not burst;
- "Today is burst and yesterday was not burst and after 4 AM" indicates the burst mode today and yesterday, and the current time is after 4 AM;
- "Today is burst and yesterday was not burst and not after 4 AM" indicates the burst mode today and yesterday, and the current time is not after 4 AM.

Notes

SurveyManager.log.csv

Summary

This file records logs from the class SurveyManager. It is originally created under /.microt/logs/ on smartphone. The class SurveyManager is called by MinuteService to launch scheduled surveys.

Metadata Columns

[Col 0] - Logging Time

Descriptions: logging time

Format: String

[Col 1] - Logging Type

Descriptions: logging type

Format: String

[Col 2] - Participant’s Visualizer ID

Descriptions: Participant’s visualizer ID

Format: String

[Col 3] - Class Name

Descriptions: class name

Format: String

[Col 4] - Logs

Descriptions: logs

Format: String

Notes

SystemBroadcastReceiver.log.csv

This file records logs from the class AlwaysOnService. The logs records system broadcast intents on smartphones.

Metadata Columns

[Col 0] - Logging Time

Descriptions: logging time

Format: String

[Col 1] - Logging Type

Descriptions: logging type

Format: String

[Col 2] - Participant's Visualizer ID

Descriptions: Participant's visualizer ID

Format: String

[Col 3] - Class Name

Descriptions: class name

Format: String

[Col 4] - Logs

Descriptions: class logs

Format: String

Value: system broadcast intents (time tick, battery changed, configuration changed, screen on/off, user present, user unlocked, power connected/disconnected, shutdown, boot completed, reboot completed, headset connected, airplane mode, dnd mode)

Notes

UploadManager-upload.err.log

This file records logs from the class UploadManager. The class SurveyManager is called by MinuteService => UploadManagerService to upload files. The logs record errors in the file uploading process.

Metadata Columns

[Col 0] - Logging Time

Descriptions: logging time

Format: String

[Col 1] - Logging Type

Descriptions: logging type

Format: String

[Col 2] - Participant's Visualizer ID

Descriptions: Participant's visualizer ID

Format: String

[Col 3] - Class Name

Descriptions: class name

Format: String

[Col 4] - Logs

Descriptions: class logs

Format: String

Notes

UploadManager-upload.log.csv

This file records logs from the class UploadManager. The class SurveyManager is called by MinuteService => UploadManagerService to upload files. The logs record file uploading status.

Metadata Columns

[Col 0] - Logging Time

Descriptions: logging time

Format: String

[Col 1] - Logging Type

Descriptions: logging type

Format: String

[Col 2] - Participant's Visualizer ID

Descriptions: Participant's visualizer ID

Format: String

[Col 3] - Class Name

Descriptions: class name

Format: String

[Col 4] - Logs

Descriptions: class logs

Format: String

Notes

Util.log.csv

This file records logs from the class Util. The class Util is called by MinuteService to set minute service alarms.

Metadata Columns

[Col 0] - Logging Time

Descriptions: logging time

Format: String

[Col 1] - Logging Type

Descriptions: logging type

Format: String

[Col 2] - Participant’s Visualizer ID

Descriptions: Participant’s visualizer ID

Format: String

[Col 3] - Class Name

Descriptions: class name

Format: String

[Col 4] - Logs

Descriptions: class logs

Format: String

Notes

WatchUploadManagerService.log.csv

This file records logs from the class WatchUploadManagerService. It is originally created under /.microt/logs/ on smartphones. The class WatchUploadManagerService is called by MinuteService and UploadManagerService to encrypt and upload watch data and log files to Firebase.

Metadata Columns

[Col 0] - Logging Time

Descriptions: logging time

Format: String

[Col 1] - Logging Type

Descriptions: logging type

Format: String

[Col 2] - Participant's Visualizer ID

Descriptions: Participant's visualizer ID

Format: String

[Col 3] - Class Name

Descriptions: class name

Format: String

[Col 4] - Logs

Descriptions: class logs

Format: String

Notes

WearDataListenerService.log.csv

This file records logs from the class WearDataListenerService. It is originally created under /.microt/logs/ on smartphones. The class WearDataListenerService is called to listen to messages from the phone to trigger uEMA prompting.

Metadata Columns

[Col 0] - Logging Time

Descriptions: logging time

Format: String

[Col 1] - Logging Type

Descriptions: logging type

Format: String

[Col 2] - Participant's Visualizer ID

Descriptions: Participant's visualizer ID

Format: String

[Col 3] - Class Name

Descriptions: class name

Format: String

[Col 4] - Logs

Descriptions: class logs

Format: String

Notes

WearNoteSender.log.csv

This file records logs from the class WearNoteSender. It is originally created under /.microt/logs/ on smartphones. The class WearNoteSender is called by many services to send note to the watch and send the config file from phone to watch.

Metadata Columns

[Col 0] - Logging Time

Descriptions: logging time

Format: String

[Col 1] - Logging Type

Descriptions: logging type

Format: String

[Col 2] - Participant’s Visualizer ID

Descriptions: Participant’s visualizer ID

Format: String

[Col 3] - Class Name

Descriptions: class name

Format: String

[Col 4] - Logs

Descriptions: class logs

Format: String

Notes

Zipper.log.csv

Summary

This file records logs from the class Zipper. It is originally created under /.microt/logs/ on smartphones. The class Zipper is called by many services to zip files and the errors are logged.

Metadata Columns

[Col 0] - Logging Time

Descriptions: logging time

Format: String

[Col 1] - Logging Type

Descriptions: logging type

Format: String

[Col 2] - Participant's Visualizer ID

Descriptions: Participant's visualizer ID

Format: String

[Col 3] - Class Name

Descriptions: class name

Format: String

[Col 4] - Logs

Descriptions: class logs

Format: String

Notes

Watch-AccelSampling.log.csv

Summary

This data file is created by the class SensorManagerService to record 10-second-window area under the curve values for the three axises computed from raw accel data collected from smartwatches.

Metadata Columns

[Col 0] - Logging Time

Descriptions: logging time

Format: String

[Col 1] - Logging Type

Descriptions: logging type

Format: String

[Col 2] - Participant's Visualizer ID

Descriptions: Participant's visualizer ID

Format: String

[Col 3] - Class Name

Descriptions: class name

Format: String

[Col 4] - Area Under the Curve Value

Descriptions: logs of Area Under the Curve Value

Format: String

Value: sensor event detection timestamp, actual sample count, area under x-axis, area under y-axis, area under z-axis.

Notes

Watch-DozeAlarmManager.log.csv

Summary

This file records logs from the class DozeAlarmManager. It is originally created under /.microt/logs-watch/ on smartwatch. The class DozeAlarmManager is called by PromptScheduler to to set doze alarms on smartwatches.

Metadata Columns

[Col 0] - Logging Time

Descriptions: logging time

Format: String

[Col 1] - Logging Type

Descriptions: logging type

Format: String

[Col 2] - Participant's Visualizer ID

Descriptions: Participant's visualizer ID

Format: String

[Col 3] - Class Name

Descriptions: class name

Format: String

[Col 4] - Logs

Descriptions: logs

Format: String

Notes

Watch-FileProcessingService.log.csv

Summary

This file records logs from the class FileProcessingService. It is originally created under /.microt/logs-watch/ on smartwatch. The class FileProcessingService is called to transfer data and log files from smartwatches to smartphones. The logs record battery level, storage, and RAM size check as the conditions of launching the transferring task. The thresholds of launching the transferring task are battery >= 20, storage >= 205 MB, and RAM >= 100 MB on smartwatches.

Metadata Columns

[Col 0] - Logging Time

Descriptions: logging time

Format: String

[Col 1] - Logging Type

Descriptions: logging type

Format: String

[Col 2] - Participant's Visualizer ID

Descriptions: Participant's visualizer ID

Format: String

[Col 3] - Class Name

Descriptions: class name

Format: String

[Col 4] - Logs

Descriptions: logs

Format: String

Notes

Watch-MiscLogger.log.csv

Summary

This file records logs from the class MiscLogger. It is originally created under /.microt/logs-watch/ on smartwatch. The class MiscLogger is called by AlwaysOnService to record app release versions, OS version, Android wear version, DND status, and notification channel status from smartwatches.

Metadata Columns

[Col 0] - Logging Time

Descriptions: logging time

Format: String

[Col 1] - Logging Type

Descriptions: logging type

Format: String

[Col 2] - Participant's Visualizer ID

Descriptions: Participant's visualizer ID

Format: String

[Col 3] - Class Name

Descriptions: class name

Format: String

[Col 4] - Logs

Descriptions: logs

Format: String

Notes

Watch-NotificationsHandler.log.csv

Summary

This file records logs from the class NotificationsHandler. It is originally created under /.microt/logs-watch/ on smartwatch. The class NotificationsHandler is called to manage notifications prompted on smartwatches.

Metadata Columns

[Col 0] - Logging Time

Descriptions: logging time

Format: String

[Col 1] - Logging Type

Descriptions: logging type

Format: String

[Col 2] - Participant's Visualizer ID

Descriptions: Participant's visualizer ID

Format: String

[Col 3] - Class Name

Descriptions: class name

Format: String

[Col 4] - Logs

Descriptions: logs

Format: String

Notes

Watch-PhoneCommunicator.log.csv

Summary

This file records logs from the class PhoneCommunicator. It is originally created under /.microt/logs-watch/ on smartwatch. The class PhoneCommunicator is called to send message from smartwatches to smartphones. The logs record the sent message.

Metadata Columns

[Col 0] - Logging Time

Descriptions: logging time

Format: String

[Col 1] - Logging Type

Descriptions: logging type

Format: String

[Col 2] - Participant's Visualizer ID

Descriptions: Participant's visualizer ID

Format: String

[Col 3] - Class Name

Descriptions: class name

Format: String

[Col 4] - Logs

Descriptions: logs

Format: String

Notes

Watch-PowerConnectionReceiver.log.csv

This file records logs from the class PowerConnectionReceiver. It is originally created under /.microt/logs-watch/ on smartwatches. The class PowerConnectionReceiver is called by AlwaysOnService to record system broadcast intents such as charging status, watch power save mode, shutdown and booting events of smartwatches.

__**Metadata Columns**__

[Col 0] - Logging Time

Descriptions: logging time

Format: String

[Col 1] - Logging Type

Descriptions: logging type

Format: String

[Col 2] - Participant's Visualizer ID

Descriptions: Participant's visualizer ID

Format: String

[Col 3] - Class Name

Descriptions: class name

Format: String

[Col 4] - Logs

Descriptions: class logs

Format: String

__**Notes**__

Watch-PowerConnectionReceiverNotes.log.csv

This file records logs from the class PowerConnectionReceiver. It is originally created under /.microt/logs-watch/ on smartwatches. The class PowerConnectionReceiver is called by AlwaysOnService to record system broadcast intents such as charging status, watch power save mode, shutdown and booting events of smartwatches.

Metadata Columns

[Col 0] - Logging Time

Descriptions: logging time

Format: String

[Col 1] - Logging Type

Descriptions: logging type

Format: String

[Col 2] - Participant's Visualizer ID

Descriptions: Participant's visualizer ID

Format: String

[Col 3] - Class Name

Descriptions: class name

Format: String

[Col 4] - Logs

Descriptions: class logs

Format: String

Notes

Watch-PromptScheduler.log.csv

Summary

This file records logs from the class PromptScheduler. It is originally created under /.microt/logs-watch/ on smartwatch. The class PromptScheduler is called by AlwaysOnService_to schedule prompts on smartwatches.

Metadata Columns

[Col 0] - Logging Time

Descriptions: logging time

Format: String

[Col 1] - Logging Type

Descriptions: logging type

Format: String

[Col 2] - Participant's Visualizer ID

Descriptions: Participant's visualizer ID

Format: String

[Col 3] - Class Name

Descriptions: class name

Format: String

[Col 4] - Logs

Descriptions: logs

Format: String

Notes

Watch-SamplingRate.log.csv

Summary

This data file is created by the class SensorManagerService to record smartwatch accelerometer sampling rate. The unit of measurement is HZ.

Metadata Columns

[Col 0] - Logging Time

Descriptions: logging time

Format: String

[Col 1] - Logging Type

Descriptions: logging type

Format: String

[Col 2] - Participant's Visualizer ID

Descriptions: Participant's visualizer ID

Format: String

[Col 3] - Class Name

Descriptions: class name

Format: String

[Col 4] - Logs

Descriptions: the timestamp of the latest watch accelerometer detection, watch accelerometer sampling rate

Format: Numerical, HZ

Notes

Watch-SensorManagerService.log.csv

Summary

This file records logs from the class SensorManagerService. It is originally created under /.microt/logs-watch/ on smartwatch. The class SensorManagerService is called to observe and record accelerometer activity changes on smartwatches. The raw accel value is written in Raw Watch Accelerometer Baf File. The values of area under the curve are computed and write the results in Watch-AccelSampling.csv.

Metadata Columns

[Col 0] - Logging Time

Descriptions: logging time

Format: String

[Col 1] - Logging Type

Descriptions: logging type

Format: String

[Col 2] - Participant's Visualizer ID

Descriptions: Participant's visualizer ID

Format: String

[Col 3] - Class Name

Descriptions: class name

Format: String

[Col 4] - Logs

Descriptions: logs

Notes

Watch-TimeStudyArbitrater.log.csv

Summary

This file records logs from the class TimeStudyArbitrater. It is originally created under /.microt/logs-watch/ on smartwatch. The class TimeStudyArbitrater is used as a callback which has been invoked from BluetoothSensorService in WocketsLib. Contains arbitrary functions (mostly power related) which did not fit anywhere else in the codebase but need to be performed frequently.

Metadata Columns

[Col 0] - Logging Time

Descriptions: logging time

Format: String

[Col 1] - Logging Type

Descriptions: logging type

Format: String

[Col 2] - Participant's Visualizer ID

Descriptions: Participant's visualizer ID

Format: String

[Col 3] - Class Name

Descriptions: class name

Format: String

[Col 4] - Logs

Descriptions: logs

Format: String

Notes

Watch-TimeStudyWearListenerService.log.csv

Summary

This file records logs from the class TimeStudyWearListenerService. It is originally created under /.microt/logs-watch/ on smartwatch. The class TimeStudyWearListenerService is called to send messages from smartwatches to smartphones.

Metadata Columns

[Col 0] - Logging Time

Descriptions: logging time

Format: String

[Col 1] - Logging Type

Descriptions: logging type

Format: String

[Col 2] - Participant's Visualizer ID

Descriptions: Participant's visualizer ID

Format: String

[Col 3] - Class Name

Descriptions: class name

Format: String

[Col 4] - Logs

Descriptions: logs

Format: String

Notes

Watch-TimeStudyWearListenerServiceNotes.log.csv

Summary

This file records logs from the class TimeStudyWearListenerService. It is originally created under /.microt/logs-watch/ on smartwatch. The class TimeStudyWearListenerService is called to send messages from smartwatches to smartphones.

Metadata Columns

[Col 0] - Logging Time

Descriptions: logging time

Format: String

[Col 1] - Logging Type

Descriptions: logging type

Format: String

[Col 2] - Participant's Visualizer ID

Descriptions: Participant's visualizer ID

Format: String

[Col 3] - Class Name

Descriptions: class name

Format: String

[Col 4] - Logs

Descriptions: logs

Format: String

Notes

Watch-UtilityMethods.log.csv

Summary

This file records logs from the class UtilityMethods. It is originally created under /.microt/logs-watch/ on smartwatch. The class UtilityMethods contains utility functions called by various services on smartwatches.

Metadata Columns

[Col 0] - Logging Time

Descriptions: logging time

Format: String

[Col 1] - Logging Type

Descriptions: logging type

Format: String

[Col 2] - Participant's Visualizer ID

Descriptions: Participant's visualizer ID

Format: String

[Col 3] - Class Name

Descriptions: class name

Format: String

[Col 4] - Logs

Descriptions: logs

Format: String

Notes

Watch-WWAlwaysOnService.log.csv

Summary

This file records logs from the class AlwaysOnService. It is originally created under /.microt/logs-watch/ on smartwatch. The class AlwaysOnService is for performing tasks defined in at scheduled intervals

Metadata Columns

[Col 0] - Logging Time

Descriptions: logging time

Format: String

[Col 1] - Logging Type

Descriptions: logging type

Format: String

[Col 2] - Participant's Visualizer ID

Descriptions: Participant's visualizer ID

Format: String

[Col 3] - Class Name

Descriptions: class name

Format: String

[Col 4] - Logs

Descriptions: logs

Format: String

Notes

Watch-WearListenerService.log.csv

Summary

This file records logs from the class WearListenerService. It is originally created under /.microt/logs-watch/ on smartwatch. The class WearListenerService is called to receive messages from smartphones.

Metadata Columns

[Col 0] - Logging Time

Descriptions: logging time

Format: String

[Col 1] - Logging Type

Descriptions: logging type

Format: String

[Col 2] - Participant's Visualizer ID

Descriptions: Participant's visualizer ID

Format: String

[Col 3] - Class Name

Descriptions: class name

Format: String

[Col 4] - Logs

Descriptions: logs

Format: String

Notes

Watch-WearableMessageTransferService.log.csv

Summary

This file records logs from the class WearableMessageTransferService. It is originally created under /.microt/logs-watch/ on smartwatch.

Metadata Columns

[Col 0] - Logging Time

Descriptions: logging time

Format: String

[Col 1] - Logging Type

Descriptions: logging type

Format: String

[Col 2] - Participant's Visualizer ID

Descriptions: Participant's visualizer ID

Format: String

[Col 3] - Class Name

Descriptions: class name

Format: String

[Col 4] - Logs

Descriptions: logs

Format: String

Notes

Watch-WearableWakefulService.log.csv

Summary

This file records logs from the class WearableWakefulService. It is originally created under /.microt/logs-watch/ on smartwatch.

Metadata Columns

[Col 0] - Logging Time

Descriptions: logging time

Format: String

[Col 1] - Logging Type

Descriptions: logging type

Format: String

[Col 2] - Participant's Visualizer ID

Descriptions: Participant's visualizer ID

Format: String

[Col 3] - Class Name

Descriptions: class name

Format: String

[Col 4] - Logs

Descriptions: logs

Format: String

Notes