OpenStreetCam Metadata 2.0 - Public Documentation

Metadata format 2.0

Metadata file structure

```
METADATA:2.0

HEADER

ALIAS:functionAlias;functionName;currentFunctionVersion;minimumCompatibleVersion

...

BODY

timestamp:functionAlias:parameters

...

END
```

Explanations

The new format will have 2 parts: header and body. They are explicitly separated by the keywords "HEADER", "BODY" and "END"

HEADER

- Before "HEADER" row we should have Metadata version separated by colon ":" from "METADATA" keyword as below

  `METADATA:2.0`

- Each row should contain function aliases for used functions (metadata fields groups) along with the current function version number and minimum compatible function version number.
- Please note that ALIAS keyword is separated by **colon ":"** and the rest of the elements by **semicolon ";"**

- Header **example**:

> *METADATA:2.0*
> *HEADER*
> *ALIAS:dm;DEVICEMOTION;1;1*
> *ALIAS:g;GPS;1;1*
> *ALIAS:o;OBD;1;1*
> *ALIAS:p;PRESSURE;1;1*
> *ALIAS:c;COMPAS;1;1*
> *ALIAS:f;PHOTO;1;1*
> *ALIAS:dev;DEVICE;1;1*

BODY

- Each row should contain timestamp (see timestamps below), the function alias defined in header and the parameters corresponding to the function (as described in the Functions table below)
  - **sensor timestamp** = the timestamp that was sent from OS
  - **adjusted timestamp** = the timestamp relative to last device restart + the timestamp of restart relative to Time Zero (1970). This can differ from sensor timestamp due to imprecisions caused by rest
  - **receiving timestamp** = the timestamp when the sensor information was received. Error can be larger than 0.04 seconds.
  - **writing timestamp** = the timestamp at the moment of writing in metadata file (this can differ from receiving timestamp due to multiple threads handling)
- Functions that don't have a timestamp (eg: DEVICE) should use writing timestamp
- Please note that functionAlias is separated by **colons ":"** on both sides
- Missing fields values will be marked by empty strings. (We are not proud of this since it might lead to lines that will contain sequences like ;;;; which is bad and Santa will skip us ☹)

- Body (partial) **example**:

> *BODY*
>
> *1539963940.6919626:dev:iOS;iOS;12;iPhone10,6;2.4;8*
>
> *1539963940.6919627:ra:3.997098299279162e-05;-0.0010819015199709063;-0.003815018909946694;0.002487492747604847;0.0014815751928836107;0.02194952964782715;-0.0038150073*
>
> *1539963940.7321248:ra:3.8036573396457776e-05;-0.0010073952388624202;-0.0038958323145342295;-0.0003156054299324751;-6.135005969554186e-05;0.019828319549560547;-0.0038958*

END

Having "END" is a confirmation that the metadata file was successfully written.

FULL METADATA FILE EXAMPLE

```
METADATA:2.0

HEADER

ALIAS:dm;DEVICEMOTION;1;1

ALIAS:g;GPS;1;1

ALIAS:o;OBD;1;1

ALIAS:p;PRESSURE;1;1

ALIAS:c;COMPAS;1;1

ALIAS:f;PHOTO;1;1

ALIAS:dev;DEVICE;1;1

BODY

1539963940.6919626:dev:iOS;iOS;12;iPhone10,6;2.4;8

1539963940.6919627:ra:3.997098299279162e-05;-0.0010819015199709063;-0.003815018909946694;0.002487492747604847;0.0014815751928836107;0.02194952964

1539963940.7321248:ra:3.8036573396457776e-05;-0.0010073952388624202;-0.0038958323145342295;-0.0003156054299324751;-6.135005969554186e-05;0.019828

END
```

Functions == metadata fields groups				
Function name	**Recommended function alias**	**Function version**	**Function Timestamp used on body**	**Fields (sensors) from function**
PHOTO	f	1	receiving	videoIndex;frameIndex;gpsTimestamp;latitude;longitude;horizontalAccuracy;GPSspeed;compasTimestamp;compass;obdTimestamp;OBDspeed
GPS	g	1	sensor	latitude;longitude;altitude;horizontalAccuracy;verticalAccuracy;GPSspeed
ACCELERATION	a	1		accelerationX;accelerationY;accelerationZ
COMPASS	c	1	sensor	compass
OBD	o	1	receiving	OBDspeed
PRESSURE	p	1	sensor	pressure
ATTITUDE	y	1		yaw;pitch;roll
GRAVITY	x	1		gravityX;gravityY;gravityZ
DEVICE	d	1	writing	platformName;osRawName;osVersion;deviceRawName;appVersion;appBuildNumber;recordingType
DEVICEMOTION	m	1	adjusted	yaw;pitch;roll;accelerationX;accelerationY;accelerationZ;gravityX;gravityY;gravityZ
CAMERA	cam	1	writing	vFOV;vZF;aperture
EXIF	exif	1	writing	fNumber;fLen

Metadata fields

Metadata fields			
Data field	**Abbreviation**	**Unit and/or example**	**Details**

version	ver	2.0.1	Metadata format version. If it is missing 2.0 is the version.
timestamp	ts	fractions of seconds eg.: 1477518763.871	The interval between the date object and 00:00:00 UTC on 1 January 1970. This property’s value is negative if the date object is earlier than 00:00:00 UTC on 1 January 1970.
Device and app details			
platform	platform	eg.: iOS	The platform from which the track was recorded: iOS, Android, Waylens, GarminVIRB, GoPRO, ..., Other
OS raw name	osRawName	eg: OxygenOS	Custom version of operating system. Default OS name (platform) will be used if there is no customized version. Ex: iOS, Yun OS, MIUI, Paranoid Android
OS version	os	eg.: 12.1	The OS version from the device from which the track was recorded.
device raw name	device	eg.: iPhone10,6	The raw name of the device. Eg: iPhone10,3 for iPhone X.
app version	appVersion	eg: 2.4	App version with X.Y or X.Y.Z format. Eg: 2.4, 2.4.1
app build number	appBuildNumber	natural number eg: 2	Build number for app version.
recording type	recordingType	string eg.: video	The type of recording: video, photo, vdb
Device sensors			
longitude	lon	degrees ex.: -123.0655579	The longitude in degrees. Measurements are relative to the zero meridian, with positive values extending east of the meridian and negative values extending west of the meridian.
latitude	lat	degrees ex.: 49.27723752	The latitude in degrees. Positive values indicate latitudes north of the equator. Negative values indicate latitudes south of the equator.
elevation	elv	meters ex.: 40.0	The altitude measured in meters. Positive values indicate altitudes above sea level. Negative values indicate altitudes below sea level.
horizontal accuracy	hAccu	meters ex.: 7.0	The radius of uncertainty for the location, measured in meters. The location’s latitude and longitude identify the center of the circle, and this value indicates the radius of that circle. A negative value indicates that the location’s latitude and longitude are invalid.
vertical accuracy	vAccu	m	The accuracy (**from GPS**) of the altitude value in meters. The value in the altitude property could be plus or minus the value indicated by this property. A negative value indicates that the altitude value is invalid.
GPS speed	GPSs	m/s ex.: 8.73	The instantaneous speed of the device in meters per second. This value reflects the instantaneous speed of the device in the direction of its current heading. A negative value indicates an invalid speed. Because the actual speed can change many times between the delivery of subsequent location events, you should use this property for informational purposes only.

yaw	yaw	radians	rotation of the device around vector through the body of the phone, interval is [-PI,PI] The yaw of the device, in radians. A yaw is a rotation around an axis that runs vertically through the device. It is perpendicular to the body of the device, with its origin at the center of gravity and directed toward the bottom of the device.
pitch	pitch	radians	rotation of the device around vector along phone left to right, interval is [-PI,PI] The pitch of the device, in radians. A pitch is a rotation around a lateral axis that passes through the device from side to side.
roll	roll	radians	rotation of the device around vector along phone body bottom to top, interval is [-PI,PI] The roll of the device, in radians. A roll is a rotation around a longitudinal axis that passes through the device from its top to bottom.
acceleration X	accX	G	linear acceleration in Gs along the vector from the phones left to right The acceleration that the user is giving to the device. X-axis acceleration in G's (gravitational force).
acceleration Y	accY	G	linear acceleration in Gs along the vector from the phones bottom to top Y-axis acceleration in G's (gravitational force).
acceleration Z	accZ	G	linear acceleration in Gs along the vector from the phones back to front Z-axis acceleration in G's (gravitational force).
pressure	pres	kPa	The recorded pressure, in **kiloPascals**.
compass	comp	degrees eg.: 215	The heading (measured **clockwise** in degrees) **relative to true north**. The value in this property represents the heading relative to the geographic North Pole. The value 0 means the device is pointed toward true north, 90 means it is pointed due east, 180 means it is pointed due south, and so on. A negative value indicates that the heading could not be determined.
video index	vIndex	global video index, interval [0;), ex.: 24	Index of the Video relative to the whole trip, Integer
frame index	fIndex	global frame index, interval [0;), ex.: 2153	Index of the current frame relative to the whole trip. Integer
gravity X	gX	G, gravity on the X vector	The gravity acceleration vector expressed in the device's reference frame. X-axis acceleration in G's (gravitational force).
gravity Y	gY	G, gravity on the Y vector	Y-axis acceleration in G's (gravitational force).
gravity Z	gZ	G, gravity on the Z vector	Z-axis acceleration in G's (gravitational force).

OBD2 speed	OBDs	km/h eg.: 42	km/h, Integer
Camera parameters - iOS			
videoFieldOfView	vFOV	float	Field of view in degrees. If field of view is unknown, a value of 0 is returned.
videoZoomFactor	vZF	float	A value that controls the cropping and enlargement of images captured by the device. This value is a multiplier. For example, a value of `2.0` doubles the size of an image's subject (and halves the field of view).
lensAperture	aperture		Aperture of the device.
fNumber	fNumber		According to wikipedia: The f-number *N* or f# is given by: *N* = / where is the focal length, and is the diameter of the entrance pupil
focalLength	fLen		According to wikipedia: The **focal length** of an optical system is a measure of how strongly the system converges or diverges light. For an optical system in air, it is the distance over which initially collimated (parallel) rays are brought to a focus.
intrinsicParams	intrinsicPram	[f_x, f_y, c_x, c_y, s]	The values `fx` and `fy` are the pixel focal length, and are identical for square pixels. The values `ox` and `oy` are the offsets of the principal point from the top-left corner of the image frame. All values are expressed in pixels. s is the squee
intrinsicMatrixReferenceDimensions	intrinsicRefDim	(width, height)	The image dimensions to which the camera's intrinsic matrix values are relative. The `intrinsicMatrix` property measures focal length and principal point offset in pixels, but those values are meaningful only in the context of an image of this size.
pixelSize	pxSize	mm	The size, in millimeters, of one image pixel.
extrinsicMatrix	extrinsicMat	matrix_float4x3	The extrinsic matrix consists of a unitless 3x3 rotation matrix (`R`) on the left and a translation (`t`) 3x1 column vector on the right. The translation vector's units are millimeters. $$\left[R \mid t\right] = \left[\begin{array}{ccc\|c} r_{1,1} & r_{1,2} & r_{1,3} & t_1 \\ r_{2,1} & r_{2,2} & r_{2,3} & t_2 \\ r_{3,1} & r_{3,2} & r_{3,3} & t_3 \end{array}\right]$$ The camera's pose is expressed with respect to a reference camera (camera-to-world view). If the rotation matrix is an identity matrix, then this camera is the reference camera. (Note that a `matrix_float4x3` matrix is column major with 3 rows and 4 columns.)

lensDistortionLookupTable	distortTable		A map of floating-point values describing radial distortions imparted by the camera lens, for use in rectifying camera images. Images captured by a camera are geometrically warped by small imperfections in the lens. In order to project from the 2D image plane back into the 3D world, the images must be distortion corrected, or made rectilinear. Lens distortion is modeled using a one-dimensional lookup table of 32-bit float values evenly distributed along a radius from the center of the distortion to a corner, with each value representing a magnification of the radius. This model assumes symmetrical lens distortion.
inverseLensDistortionLookupTable	inverseDistortTable		A map of floating-point values describing radial distortions for use in reapplying camera geometry to a rectified image. If you've rectified an image by removing the distortions characterized by the `lensDistortionLookupTable` p roperty, and now wish to go back to a geometrically distorted image (for example, to render visual effects into the camera image or perform computer vision tasks such as scene reconstruction), use this inverse lookup table.
lensDistortionCenter	distortCenter	(x, y)	The offset of the distortion center of the camera lens from the top-left corner of the image. Due to geometric distortions in the image, the center of the distortion may not be equal to the optical center (principal point) of the lens. When making an image rectilinear, the distortion center should be used rather than the optical center of the image.
lensPoseRotation	lensRotation	float[] (x, y, z, w)	
lensPoseTranslation	lensTranslation	float[] (x, y, z)	
scalarCropRegion	scalarCropReg	Rect[int, int, int, int]	