API

CamON Live implements an API that allows a better integration with clients or services.

Streams

Still image

`http://<ipaddress>:<port>/video/jpeg`

Returns a single JPEG image.

Note: for compatibility, the following aliases are also supported:

`/jpeg`

`/video.jpg`

`/live/jpeg`

`/video/video.jpg`

MJPEG

`http://<ipaddress>:<port>/video/mjpeg`

`http://<ipaddress>:<port>/video/mjpeg?fps=<fps>`

Plays the MJPEG video stream.

`fps` (double) - the desired framerate in frames per seconds

Note: for compatibility, the following aliases are also supported:

`/mjpeg`

`/video.mjpg`

`/mjpg/video.mjpg`

`/live/mjpeg`

`/live/0/mjpeg.jpg`

`/live/0/mjpeg.sdp`

`/video/video.mjpg`

RTSP

`rtsp://<ipaddress>:<port>/video/h264`

Plays the AAC/H264 audio/video stream.

Note: for compatibility, the following aliases are also supported:

`/h264`

```
/H264
/live
/live.sdp
/live/h264
/live/0/h264.sdp
```

HLS

`http://<ipaddress>:<port>/video/live.m3u8`

Plays the HLS live playlist, with AAC audio and H264 video contents.

To embed the stream with Video.js, use:
<source src=\"http://<ipaddress>:<port>/video/live.m3u8\" type=\"application/vnd.apple.mpegurl m3u8\"/>

Status request

It is possible to query the device for some internal status by sending a GET request to the server. The result will be a JSON object containing all the related information.

Status

`http://<ipaddress>:<port>/status`

Returns the overall device's status.

Note: if the headers contain *gps-mode: fine* or *gps-mode: coarse*, or the query contains *gps-mode=fine* or *gps-mode=coarse*, the location mode will be changed accordingly.

`model` (string) - the device model

`serial` (string) - the device serial number

`id` (string) - the device ID

`version` (string) - the app version

`location` (object) - the location object

- `latitude` (double) - the latitude, in degrees
- `longitude` (double) - the longitude, in degrees
- `time` (long) - the UTC time, in milliseconds since January 1, 1970
- `accuracy` (float) - the estimated accuracy, in meters
- `provider` (string) - the name of the provider that generated the fix

`connections` (array) - the list of active connections

- `client_address` (string) - the host name of the connected client or its IP address

- `MJPEG_stream` (boolean) - whether this client is streaming MJPEG video
- `H264_stream` (boolean) - whether this client is streaming H264 video
- `AAC_stream` (boolean) - whether this client is streaming AAC audio
- `RTSP_session_ID` (string) - the RTSP session ID, when streaming H264/AAC

`streams` (int) - the number of active streams

`WiFi` (boolean) - whether a local network is the active connection; it could be WiFi, but also Ethernet, Bluetooth or USB tethering

`mobile` (boolean) - whether the mobile data is the active connection

`mobile_type` (string) - the mobile network type, one of TelephonyManager.NETWORK_TYPE_* names

`H264` (boolean) - whether the H264 video stream is available

`audio` (boolean) - whether the AAC audio stream is available

Sensors

`http://<ipaddress>:<port>/sensors`

Returns the last known sensors's status.

Note: if the headers contain *sensors: trigger*, or the query contains *sensors=trigger*, the device's sensors acquisition will be triggered.

`battery` (object) - the battery object

- `connection` (string) - the battery power source, one of BatteryManager.BATTERY_PLUGGED_* names
- `level` (float) - the battery level

`torch` (boolean) - whether the torch is turned on

`temperature` (float) - ambient air temperature in °C (9999 if not available)

`humidity` (float) - ambient relative humidity in % (9999 if not available)

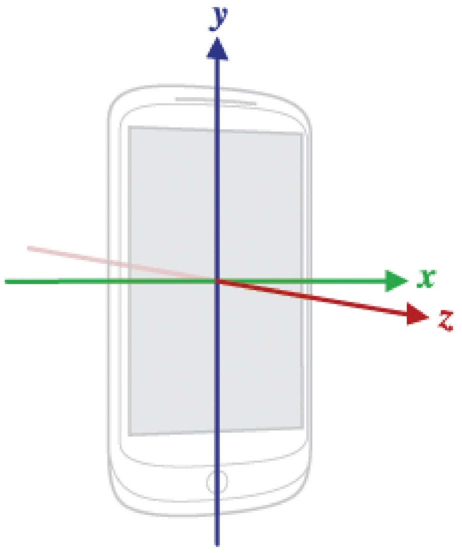
`pressure` (float) - ambient air pressure in hPa or mbar (9999 if not available)

`light` (float) - Illuminance in lx (9999 if not available)

`gravity` (array) - the gravity vector in m/s^2 (9999 if not available)

- `x` (float) - force of gravity along the x axis
- `y` (float) - force of gravity along the y axis
- `z` (float) - force of gravity along the z axis

The coordinate system is defined relative to the device's screen when the device is held in its default orientation:



X axis is horizontal and points to the right

Y axis is vertical and points up

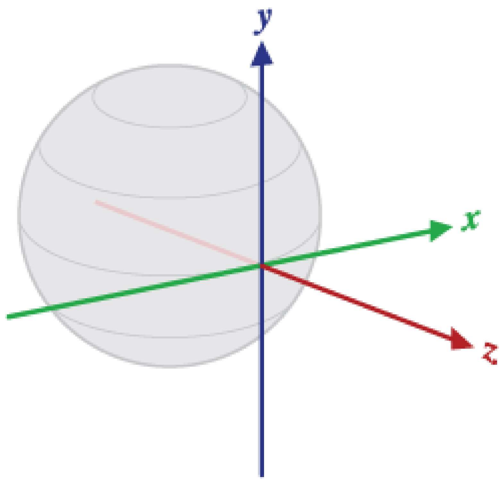
Z axis points toward the outside of the screen face

rotation (array) - a vector that represents the orientation of the device as a combination of an angle and an axis, in which the device has rotated through an angle θ around an axis (x, y, or z), unitless (9999 if not available)

- x (float) - rotation vector component along the x axis (x * sin(θ/2))
- y (float) - rotation vector component along the y axis (y * sin(θ/2))
- z (float) - rotation vector component along the z axis (z * sin(θ/2))

The magnitude of the rotation vector is equal to sin(θ/2), and the direction of the rotation vector is equal to the direction of the axis of rotation.

The reference coordinate system is defined as follows:



X is tangential to the ground at the device's current location and points approximately East

Y is tangential to the ground at the device's current location and points toward the geomagnetic North Pole

Z points toward the sky and is perpendicular to the ground plane

The device's x, y, and z axes are defined in the same way as the gravity vector.

Motion history

`http://<ipaddress>:<port>/motion`

Returns the latest motion events.

`motion_events` (array) - the array of the latest known motion events

- `state` (boolean) - true when a motion event starts
- `timestamp` (long) - the UTC timestamp, in milliseconds since January 1, 1970

Video and audio

`http://<ipaddress>:<port>/video`

`http://<ipaddress>:<port>/audio`

The available video and audio sources.

`streams` (array) - the list of available streams

- `url` (string) - the stream URL
- `mime` (string) - the MIME type
- `available` (boolean) - whether the stream is available
- `parameters` (array) - the list of the available query parameters, if any
    - `name` (string) - the name of the parameter
    - `min` (float) - the minimum value
    - `max` (float) - the maximum value

Control commands

It is possible to control the app remotely by sending POST requests to the URL `http://<ipaddress>:<port>/control`

The request's body should be plain text, specifying one or more commands to execute; the headers should contain *Content-Type: text/plain*.

Commands can be specified once per row, in the form *command=parameters*; rows may be either \n or \r\n terminated.

It is also possible to use a GET request and specify one or more commands in the query string, using the same form *command=parameters* for each.

Camera

`camera=id [r:<width>x<height>]`

Select a camera and sets its resolution.

If the resolution is not specified, the current one will be kept. The selected resolution may differ from the desired one, depending on the camera capabilities.

`id` (int) - the id of the camera to activate

`width` (int) - the desired resolution width in pixel

`height` (int) - the desired resolution height in pixel

Incremental zoom

`zoom=<[+,-]delta>`

Increments or decrements the camera zoom.

`delta` (int) - the desired zoom change, in percent

Absolute zoom

`zoom=<magnification>`

Sets the camera zoom to the specified magnification factor.

`magnification` (int) - the desired magnification, in 1/100x

Autofocus

`autofocus=<x>,<y>`

`autofocus=start`

Performs both autofocus and light compensation.

`x,y` (int,int) - the position of the focus/metering window, in uniform coordinates (-1000 to 1000)

`"start"` - equals to *autofocus=0,0*

Torch control

`torch=<state>`

Controls the torch associated to the active camera.

`state` (string) - either *on*, *off* or *toggle*

Video sync

`video-sync=send`

Requests the transmission of an H264 IDR frame.

Delay

`delay=<time>`

Waits for the specified time.

Can be used to synchronize other commands in the same request:
zoom=+200
delay=250
autofocus=start

`time` (int) - time to wait, in milliseconds

Motion Detection

`motion=<state> [size:<size>] [sens:<sensitivity>]`

Controls the motion detection feature.

`state` (string) - the state of the motion detector, either *on* or *off*

`size` (string) - the size of the objects to detect, either *small*, *medium* or *large*

`sensitivity` (string) - the sensitivity of the detection algorithm, either *high*, *medium* or *low*

GPS Mode

`gps-mode=<mode>`

Changes the location update mode.

`mode` (string) - either *fine* or *coarse*

Sensors

`sensors=trigger`

Triggers the device's sensors acquisition, that will stop after a timeout if no other requests are sent.

Live Streaming

`live=<state>`

Controls the live streaming state.

`state` (string) - either *play* or *stop*

Vibration

`vibrate=<duration>`

Activate vibration for the specified duration, if supported by the device.

`duration` (int) - how long to vibrate, in milliseconds

Alarm

`alarm=play [v:<volume>] [r:<repetitions>]`

Activate vibration for the specified duration, if supported by the device.

`volume` (int) - the sound volume, in percent, default is 100%

`repetitions` (int) - how many times to repeat the sound, default is 0

App Control

`app=<state>`

Controls the state of the app.

`state` (string) - either *bg* (app in background), *fg* (app in foreground) or *close* (close the app)