Camera Network Services API

Version 0.3

Version History		
0.1	6/10/2015	Initial document creation
0.2	6/16/2015	Add Video Streaming documentation.
0.3	6/25/2015	Add version 3 of the JSON API

Table of Contents

Legal Information

The use of this API is subject to the terms of the *End User License Agreement*. The EULA can be found at http://developer.garmin.com/virb/api/

Supported Devices

Unless otherwise specified, it can be assumed that this document supports at least the following devices:

VirbX VirbXE

mDNS Documentation

VirbX/XE devices will advertise mDNS services for their supported application protocols. The devices will advertise mDNS services when connected to or hosting a WiFi network depending on the networking capabilities of the device.

Services

Service Name	Protocol
_garmin-virb._tcp	This service name represents the JSON service over HTTP protocol. See the JSON Documentation section for using the JSON application protocol.

Additional Services

The VirbX/XE may advertise additional mDNS services that use the “_garmin-" prefix. These services and underlying application protocols will be included in this documentation as they become externally available.

HTTP Documentation

The VirbX/XE devices host an HTTP server that can be used to issue JSON commands or to download files from URL's retrieved from JSON commands such as mediaList (3).

The HTTP server is hosted on the standard HTTP port (80). The mDNS service can be used to determine the devices IP address.

Services

Service Name	Description
/virb	HTTP POST requests made to this address will be processed as JSON commands. Refer to the JSON Documentation section for more information about the available JSON commands.
URL provided by JSON	URLs provided by JSON responses can be used to download the designated file.

Additional Services

As additional HTTP services become publicly available this documentation will be updated to reflect the changes.

Live Preview Stream Documentation

The Live Preview Stream feature on the VirbX/XE provides a wireless video and audio stream preview of what can be or is being recorded by the camera to another device.

The Live Preview Stream has two media streams, an H.264 video stream and an AAC audio stream. The Live Preview Stream uses a set of technologies known as RTSP and RTP to control and transport these streams.

RTSP

The RTSP (Real-time Streaming Protocol, *RFC 2326*) protocol, is used to control the Live Preview Stream. Clients can discover the Live Preview Stream URL (which is used for RTSP) by issuing the livePreview (3) JSON command with streamType “rtp”. This will return the Live Preview Stream URL. See the JSON Documentation section for details on issuing commands.

Session Description

When issuing the RTSP DESCRIBE request, clients will receive a response with the Live Preview Stream session description in the form of an SDP (Session Description Protocol, *RFC 2327*) message. The Live Preview Stream is a presentation composed of two media URLs and a single aggregate control URL.

Stream Setup

Once receiving the session description, clients should issue a SETUP request for each media stream to be included in the session with the stream’s respective media URL. Valid session configurations are:

Session Configuration	Number of Streams
Video Only	1
Audio Only	1
Synchronized Audio and Video	1

NOTE: each stream requires a separate SETUP command and appropriate media URL. The Aggregate Control URL is not used in the SETUP request.

Stream Control

The aggregate control URL is used to PLAY and TEARDOWN the entire session. The session is composed of the streams that have been setup in the session with the SETUP request. Clients issue a single PLAY and TEARDOWN command using the aggregate control URL to control the entire session (and thus all streams within the session).

Stream Instances

The VirbX/XE only support a single instance of each audio and video stream. This means multiple clients cannot each open an RTSP session and setup the same stream within their session. If a client encounters this limitation, RTSP requests will fail with the status code 453 (Not Enough Bandwidth). RTSP requests will continue to fail until the outstanding instance is closed.

NOTE: The limitation is a single instance of each stream and not on the number of RTSP clients. This limitation allows for more obscure configurations like one client consuming the video stream while another client is consuming the audio stream.

Stream Availability

The Live Preview Stream feature may not always be available depending on the configuration of the camera video mode or during transitional states such as stopping or starting a recording. When the Live Preview Stream is not available due to one of these conditions, RTSP requests will fail with the status code 503 (Service Unavailable).

If encountering the 503 status code due to a transition between record states, the best practice is to retry the request in a few seconds (1-5).

If encountering the 503 status code due to configured video modes that don't support the Live Preview Stream, resolve the configuration issue and then retry the request. Clients can learn whether the Live Preview Stream is available by issuing the livePreview (3) JSON command, which will contain a result of 0 if the stream is unavailable.

Stream Halts

In certain conditions, the VirbX/XE may halt the Live Preview Stream. This happens when starting or stopping a recording. When this happens, the RTSP server will issue an RTCP packet containing a Goodbye message. The best practice for handling a stream halt is to issue a TEARDOWN request and reissue the RTSP requests to setup and play the desired streams.

Another case to note is if the stream halts after an RTSP session is setup, but before any RTP or RTCP data is sent, no Goodbye message will be sent by the device. This behavior is compliant with RFC 3350, but leaves the client in a situation where the RTSP session is valid and the RTP session has been halted. RTSP has no mechanism for a server to notify a client when the session needs to be torn down. In order to provide an indication to the client, the device will close the RTSP TCP connection in this case. Clients can use this information to trigger a TEARDOWN and then setup the session again. Clients can also handle this situation by implementing an RTP data timeout, where if no RTP data has been received for a fixed amount of time, treat the session as dead and issue a TEARDOWN followed by setting up the stream again.

RTP

The video and audio streams are transported via RTP (Real-time Transport Protocol, *RFC 3550*). The VirbX/XE will communicate RTP transport information in the standard manner of a “Transport” field within the RTSP SETUP response message for each stream. Clients will start to receive RTP data once a PLAY command has been issued for the session.

Video Format

The VirbX/XE uses H.264 for the underlying encoded video. H.264 data is sent over RTP using RTP Payload Format for H.264 Video (*RFC 6184*). Virb devices will operate in packetization-mode 1 (Non-interleaved) and send the following packet types for H.264 over RTP:

- Single NAL Unit
- Single-Time Aggregation Packet type A (STAP-A)
- Fragmentation Unit type A (FU-A)

NOTE: The VirbX/XE does not communicate the profile-level-id and sprop-parameter-sets via SDP because the encoder is already generating them in-band with the H.264 NALUs. These parameters are sent as NALUs with every I-frame.

Audio Format

The VirbX/XE uses AAC for the underlying audio encoding. AAC data is sent over RTP using RTP Payload Format for Transport of MPEG-4 Elementary Streams (*RFC 3640*). AAC streams use the Low Complexity profile and are sent using the High Bit-rate AAC mode (*Section 3.3.6, RFC 3640*).

The VirbX/XE supports the following audio configurations:

Sample Rate	Channels
48Khz	2
48Khz	1
16Khz	2
16Khz	1
8Khz	2
8Khz	1

Depending on the audio source the camera is using for a microphone, it will select one of the above mentioned configurations.

RTCP

As part of the RTP protocol, the VirbX/XE uses RTCP to track clients that are consuming RTP streams. RTP receivers must send out RTCP Receiver Reports in a timely manner or risk being timed out from the session. The cameras, following the guidance listed in (*Section 6.2.1, RFC 3550*), will timeout a client after it has missed 5 reporting intervals, which is about 25 seconds in the case of a single client.

JSON Documentation

The JSON server allows remote applications to retrieve status information and update device settings over WIFI. JSON commands can be sent to the unit using HTTP post messages with a formatted JSON command message.

NOTE: Command version numbers are found inside the () next to the command name. Always use the latest command version supported by your device.

HTTP Post message

Example JSON message sent using cURL. Message captured using Wireshark.

NOTE: For these examples the IP address is known. Use the mDNS service to determine the devices IP address. Information on the mDNS service can be found in the mDNS Documentation.

cURL Command

```
curl --data "{\"command\":\"status\"}" http://192.168.0.1/virb
```

Captured Message

```
POST /virb HTTP/1.1
User-Agent: curl/7.38.0
Host: 192.168.0.1
Accept: */*
Content-Length: 20
Content-Type: application/x-www-form-urlencoded

{"command":"status"}
```

Captured Response

```
HTTP/1.1 200 OK
Content-Type: application/json
Connection: close
Content-Length: 366

{"apiMin":3,"apiMax":3,"totalSpace":15653888,"availableSpace":6648512,
"gpsAccuracy":8,"gpsLatitude":38.856570,"gpsLongitude":-94.799157,
"speed":0.000000,"antSensor":0,"btSensor":0,"btHeadset":0,
"wifiSensor":0,"recordingTimeRemaining":73889,"photosRemaining":1400,
"photoCount":0,"recordingTime":0,"wifiMode":"AP","state":"idle",
"lastMediaEventTime":285361,"result":1}
```


Error Codes

All JSON commands will return with "result": 1 if the command was successfully completed, and "result": 0 when the command was unable to be completed successfully.

Most failure cases can be avoided by checking the current state of the system with **{"command":"status"}**, **{"command":"features"}**, or other system status/state commands.

Some command are not able to "fail" and will always return "result": 1. Best practice is to check the "result" parameter for each command issued to verify that the command succeeded.

Available Features

This section contains a list of all the available "features" that can be returned or used by the features (3) or updateFeature (3) commands.

NOTE: Features and feature options may not be available depending on the model, and system configuration.

Static Features:

Feature	Summary
previewWhileRecording	Static feature used to signal that video preview is available while recording on the device.
locateCamera	Static feature used to signal that the device supports the locate (3) and found (3) commands.

System Settings:

Feature	Summary	Options
friendlyName	The user specified camera name. The "enabled" attribute indicates whether or not the attribute has been set.	User specified name
gps	Enable/Disable GPS	on off
language	Language for the device. The list of available languages is dependent on the language files on the device.	English ...
microphoneAdvanced	Configure the microphone.	Off Internal External
powerSave	Enable/Disable the auto off feature.	1 = Enable 0 = Disable
recordingLED	Enable/Disable the recording LED.	1 = Enable 0 = Disable

Feature	Summary	Options
rotation2	Set the current device orientation.	Auto Up Down
tones	Enable/Disable system tones.	1 = Enable 0 = Disable
units	Set the device units.	Statute Metric
wowlan	Enable/Disable wake on wireless.	1 = Enable 0 = Disable

Video Settings:

Feature	Summary	Options
autoRecord	Enable auto recording.	off whenMoving
fieldOfView	Set video field of view.	wide zoom ultraZoom
stabilization	Enable/Disable video stabilization.	1 = Enable 0 = Disable
timeLapseRate	Set the video timelapse rate in seconds.	1_2 1 2 5 10 30 60
videoFormat	Sets the desired video format.	NTSC PAL
videoLensCorrection	Enable/Disable video lens correction.	1 = Enable 0 = Disable
videoLoop	Configure the video loop duration in seconds.	0 5 10 20 30
videoMode	Configure the video mode.	Tall

Feature	Summary	Options
		HD Video Slow-Mo Timelapse
videoModeFps	Configure the video frames per second.	240 200 120 100 60 50 48 30 25 24
videoModeLightBoost	Configure the video mode light boost.	on off
videoMode-PlaybackSpeed	Configure the video mode playback speed.	1/1 1/2 1/4 1/8
videoModeRes	Configure the video mode resolution.	1920x1440 1920x1080 1280x960 1280x720 848x480

Pro-Mode Video Settings:

Feature	Summary	Options
videoColorCurve	Set video color curve.	vivid flat
videoExposureBias	Set video exposure bias.	-2.0 -1.7 -1.3 -1.0 -0.7 -0.3 +0.0 +0.3

Feature	Summary	Options
		+0.7 +1.0 +1.3 +1.7 +2.0
videoExposureLock	Enable/Disable video exposure lock.	1 = Enable 0 = Disable
videoISOLimit	Set the video ISO limit.	6400 1600 400
videoProSettingEn	Enable/Disable video pro-mode settings.	1 = Enable 0 = Disable
videoSharpness	Set the video sharpness.	low medium high
videoWhiteBalance	Set the video white balance.	AUTO 2800K 4000K 5000K 6500K 7500K

Photo Settings:

Feature	Summary	Options
burstRate	Configure photo burst mode rate.	3Fps1Second 5Fps1Second 10Fps1Second 20Fps1Second 30Fps1Second 5Fps2Seconds 10Fps2Seconds 15Fps2Seconds 10Fps3Seconds 5Fps6Seconds
dataStamp	Enable/Disable photo data stamp.	1 = Enable 0 = Disable
imageSize	Set desired image size.	4000x3000

Feature	Summary	Options
		3008x2256
photoLensCorrection	Enable/Disable photo lens correction.	1 = Enable 0 = Disable
photoMode	Set desired photo mode.	Single Burst Timelapse Ext. Timelapse
photoTimeLapseRate	Configure photo timelapse rate. 0.5s is not supported on VirbX	0.5s 1s 2s 5s 10s 30s 60s
selfTimer	Configure photo self timer. Delay is in seconds.	Off 2 5 10 30 60

Available Sensors

This section contains a list of all the available "sensors" that can be returned or used by the sensors (3) command.

NOTE: Sensors may not be available depending on the model, and connected sensors.

Local Sensors:

Sensor Name	Units	Data Type	Description
InternalAccelG	Instantaneous G force	double	The overall G forces acting on the unit
InternalAccelX	Instantaneous G force	double	The G forces acting on the units X axis
InternalAccelY	Instantaneous G force	double	The G forces acting on the units Y axis
InternalAccelZ	Instantaneous G force	double	The G forces acting on the units Z axis
InternalGyroX	Degrees/Second	double	The rotational speed of the unit along its X axis
InternalGyroY	Degrees/Second	double	The rotational speed of the unit along its Y axis
InternalGyroZ	Degrees/Second	double	The rotational speed of the unit along its Z axis

ANT Sensors:

Sensor Name	Units	Data Type	Description
Cadence	RPM	int	Users cadence
FootSpeed	Meters/Second	double	Users speed detected by Foot Pod
HeartRate	BPM	int	Users heart rate
Temperature	DegreesCelsius	double	Reported temperature
WheelSpeed	Meters/Second	double	Users speed detected by wheel sensor

Commands

The following pages contain a list of the available JSON commands. Each command will have a Format, Command Member, Description, Response Objects, and Example Success and Failure Responses.

NOTE: Most commands use quotes for their parameters. Commands that don't will provide that information in the description.

NOTE: All commands are case sensitive.

deleteFile (3)

Format:

{"command":"deleteFile","files":[<file>]}

{"command":"deleteFile","files":[<file1>, <file2>, ...]}

Command Members:

Objects	Values	Description
files[]	files_object	Array of files to be deleted
files_object		URL of the media to be deleted

Description:

Deletes the selected video and picture files from the device.

Response Objects:

Objects	Members	Description
result		Result of the requested command. 1 = Success. 0 = Failure.

Example Success Response:

```
{
"result": 1
}
```

Example Failure Response:

```
{
"result": 0
}
```


deviceInfo (3)

Format:

{"command":"deviceInfo"}

Command Members:

No command members are use with this command.

Description:

Retrieves status information about the camera.

Response Objects:

Objects	Members	Description
deviceInfo[]	deviceInfo_object	Array containing the device information.
deviceInfo_object	deviceId	Device ID.
	firmware	Firmware version number. Effectively a subkey of "model". Integer value representing major/minor version, ie "123" = 1.23.
	model	Model of the unit.
	partNumber	The software part number for the system software.
	type	Hardcoded to "primary".
result		Result of the requested command. 1 = Success. 0 = Failure.

Example Success Response:

```
{
"deviceInfo": [
    {
    "model": "VIRBXE",
    "firmware": 212,
    "type": "primary",
    "partNumber": "006-B2172-00",
    "deviceId": 3899144409
    }
    ],
"result": 1
}
```


Example Failure Response:

```
{
"result": 0
}
```


features (3)

Format:

{"command":"features"}

Command Members:

No command members are use with this command.

Description:

Get current features and status.

Features are only returned if they are available. Some Response Objects are not returned depending on the type of feature and if the feature is enabled.

Features can be updated by issuing a updateFeature (3) command.

A complete list of all features and options can be found in the Available Features section.

Response Objects:

Objects	Members	Description
features[]	features_object	Array of features
features_object	enabled	Is the feature enabled (1) or disabled (0). Type 1 and 2 only.
	feature	Name of the feature.
	options[]	Array of strings representing the currently available options for the feature. This list can change based on current device state. Type 1 only.
	optionSummaries[]	Array of strings that provide a summary for each of the available options. Type 1 only.
	type	Type of feature. 0 = static feature, 1 = Feature contains list of values to be selected, 2 = Feature can only be set On (1)/Off (0)
	value	Current feature value. Type 1 and 2 only.
result		Result of the requested command. 1 = Success. 0 = Failure.

Example Success Response:

```
{
"features": [
    {
    "type": 0,
    "feature": "previewWhileRecording"
    },
    {
    "type": 1,
    "feature": "units",
    "enabled": 1,
```



```
    "value": "Statute",
    "options": [
        "Statute",
        "Metric"
        ]
    },
    {
      "type": 2,
      "feature": "powerSave",
      "enabled": 1,
      "value": "0"
    },
    ...
    ],
"result": 1
}
```

Example Failure Response:

```
{
"result": 0
}
```


found (3)

Format:

{"command":"found"}

Command Members:

No command members are use with this command.

Description:

Causes camera to stop beeping and flashing.

Response Objects:

Objects	Members	Description
result		Result of the requested command. 1 = Success. 0 = Failure.

Example Success Response:

```
{
"result": 1
}
```

Example Failure Response:

```
{
"result": 0
}
```


livePreview (3)

Format:

{"command":"livePreview","streamType": <stream_type> }

Command Members:

Objects	Values	Description
streamType	"rtp"	Requested video stream type.

Description:

Returns the URL for streaming live video from the camera.

NOTE: Not all modes support preview while recording.

Response Objects:

Objects	Members	Description
url		The URL for streaming live video from the camera.
result		Result of the requested command. 1 = Success. 0 = Failure.

Example Success Response:

```
{
    "result": 1,
    "url": "rtsp://192.168.0.1/livePreviewStream"
}
```

Example Failure Response:

```
{
    "result": 0
}
```


locate (3)

Format:

{"command":"locate"}

Command Members:

No command members are use with this command.

Description:

Causes camera to beep/flash led so it can be located.

Response Objects:

Objects	Members	Description
result		Result of the requested command. 1 = Success. 0 = Failure.

Example Success Response:

```
{
"result": 1
}
```

Example Failure Response:

```
{
"result": 0
}
```


mediaDirList (3)

Format:

{"command":"mediaDirList"}

Command Members:

No command members are use with this command.

Description:

Returns an array of media directories currently on the camera.

Response Objects:

Objects	Members	Description
mediaDirs[]	mediaDirs_object	Array of directory objects
mediaDirs_object	date	Creation date/time specified in UTC seconds since the UNIX Epoch.
	path	Directory path.
	type	Directory type.
result		Result of the requested command. 1 = Success. 0 = Failure.

Example Success Response:

```
{
"mediaDirs": [
    {
    "type": "mediadirectory",
    "path": "D:/DCIM/100_VIRB/",
    "date": 568101658
    },
    {
    "type": "mediadirectory",
    "path": "D:/DCIM/101_VIRB/",
    "date": 804175802
    }
    ],
"result": 1
}
```


Example Failure Response:

```
{
"result": 0
}
```


mediaList (3)

Format:

{"command":"mediaList"}

{"command":"mediaList", "path": <path> }

Command Members:

Objects	Values	Description
path		Path is optional. Not specifying it will search all directories on the SD card, which may be time-consuming.

Description:

Returns an array of videos and pictures currently on the camera at a given path.

If "path" key is missing in the request all the files in all the folders under DCIM will be listed.

A list of available paths can be found by issuing a mediaDirList (3) command.

NOTE: Split video files are denoted by a filename-x.MP4, where x denotes the split segment order.

Response Objects:

Objects	Members	Description
media[]	media_object	Array of media objects.
media_object	data	Creation-date/time of the main file for the media list entry, specified in UTC second in the UNIX Epoch.
	fileSize	Size of the specified media in bytes.
	fitURL	URL to download the coresponding fit file. Use the *FIT SDK* to parse this file.
	lowResVideoPath	URL to download a low resolution video file version.
	name	Name of specified media.
	thumbURL	URL to download specified media thumbnail. thumbURL is not available for video files.
	type	Type of specified media.
	url	URL to download specified media.
result		Result of the requested command. 1 = Success. 0 = Failure.

Example Success Response:

```
{
  "media": [
    {
      "type": "photo",
      "url": "http://192.168.0.1/media/photo/DCIM/104_VIRB/
VIRB0002.jpg",
```



```
      "thumbUrl": "http://192.168.0.1/media/thumb/photo/DCIM/104_VIRB/
VIRB0002.BMP",
      "fitURL": "http://192.168.0.1/media/fit/Garmin/
GMetrix/2015-07-06-15-00-16.fit",
      "name": "VIRB0002.jpg",
      "fileSize": 4550847,
      "date": 1436293508
    },
    {
      "type": "video",
      "url": "http://192.168.0.1/media/video/DCIM/104_VIRB/
VIRB0003.MP4",
      "lowResVideoPath": "http://192.168.0.1/media/lowres/video/
DCIM/104_VIRB/VIRB0003t.GLV",
      "fitURL": "http://192.168.0.1/media/fit/Garmin/
GMetrix/2015-07-06-15-00-16.fit",
      "name": "VIRB0003.MP4",
      "fileSize": 2831155200,
      "date": 1436293510
    },
    {
      "type": "video",
      "url": "http://192.168.0.1/media/video/DCIM/104_VIRB/
VIRB0003-2.MP4",
      "lowResVideoPath": "http://192.168.0.1/media/lowres/video/
DCIM/104_VIRB/VIRB0003t-2.GLV",
      "fitURL": "http://192.168.0.1/media/fit/Garmin/
GMetrix/2015-07-06-15-00-16.fit",
      "name": "VIRB0003-2.MP4",
      "fileSize": 2831155200,
      "date": 1436294412
    },
    ...
  ],
  "result": 1
}
```


Example Failure Response:


```
{
"result": 0
}
```


sensors (3)

Format:

{"command":"sensors"}

{"command":"sensors","names":[<sensor>]}

{"command":"sensors","names":[<sensor1>, <sensor2>, ...]}

Command Members:

A list of available sensor name members can be found by issuing {"command":"sensors"} and parsing the response of available sensors. The complete list of sensors can be found in the Available Sensors section.

Description:

Retrieves information from the connected sensors. The list of available sensors can change at any time since only sensors that are currently available will be returned. This allows the possibility of an empty sensors array ("sensors":[]) being returned when no sensors are available.

A {"command":"sensors"} request will always return all of the available sensors and their respective data.

A {"command":"sensors","names":[...]} with sensor members will return a list of available sensors out of those requested along with their respective data.

Response Objects:

Objects	Members	Description
sensors[]	sensors_object	Array of available sensors.
sensors_object	data	Data value of the sensor. Not reported when has_data is 0
	data_type	Determines the parsing format of the sensor data. Potential formats are: double, and int. Not reported when has_data is 0
	has_data	Determines if the sensor is just connected (0) or is reporting data (1). If not present assume value is 1.
	name	Name of the sensor.
	type	Determines if the sensor is a local ("LOCAL"), ANT ("ANT"), or bluetooth ("BT") sensor.
	units	The base unit of the sensor data.
result		Result of the requested command. 1 = Success. 0 = Failure.

Example Success Response:

```
{
"sensors": [
    {
    "name": "InternalGyroX",
    "type": "LOCAL",
    "units": "Degrees/Second",
    "data_type": "double",
```



```
    "data": "0.731707"
    },
    {
      "name": "Cadence",
      "type": "ANT",
      "has_data": "0",
      "units": "RPM"
    },
    {
      "name": "HeartRate",
      "type": "ANT",
      "has_data": "1",
      "units": "BPM",
      "data_type": "int",
      "data": "65"
    },
    ...
    ],
"result": 1
}
```


Example Failure Response:


```
{
"result": 0
}
```


snapPicture (3)

Format:

{"command":"snapPicture"}

Command Members:

No command members are use with this command.

Description:

Signals camera to take a still photo.

This command honors the device photo mode.

NOTE: The system does not respond to JSON commands while taking photos in Extended Timelapse mode. Do not expect a response if issuing snapPicture in Extended Timelapse mode.

NOTE: In Timelapse and Burst modes only the first photo is returned. Issue a mediaList (3) command to obtain the URL's for the remaining photos.

NOTE: Use the stopStillRecording (3) command to stop taking photos while in Timelapse mode.

Response Objects:

Objects	Members	Description
media	name	Name of the specified media.
	thumbUrl	URL to download the specified media thumbnail.
	type	Type of media taken (should always be photo).
	url	URL to download the specified media.
result		Result of the requested command. 1 = Success. 0 = Failure.

Example Success Response:

```
{
"result": 1,
"media": {
    "type": "photo",
    "url": "http://192.168.0.1/media/photo/DCIM/101_VIRB/
VIRB0002.jpg",
    "thumbUrl": "http://192.168.0.1/media/thumb/photo/DCIM/101_VIRB/
VIRB0002.jpg",
    "name": "VIRB0002.jpg"
    }
}
```


Example Failure Response:

```
{
"result": 0
}
```


startRecording (3)

Format:

{"command":"startRecording"}

Command Members:

No command members are use with this command.

Description:

Causes camera to begin recording a video.

NOTE: Not all video modes support live preview while recording.

Response Objects:

Objects	Members	Description
result		Result of the requested command. 1 = Success. 0 = Failure.

Example Success Response:

```
{
    "result":1
}
```

Example Failure Response:

```
{
    "result":0
}
```


status (3)

Format:

{"command":"status"}

Command Members:

No command members are use with this command.

Description:

Retrieves status information about the camera.

Response Objects:

Objects	Members	Description
antSensor		Is an ANT+ sensor attached? 0 means false, 1 means true, not present means software may have it but doesn't support this JSON API
apiMax		Maximum JSON API revision supported
apiMin		Minimum JSON API revision supported
availableSpace		Storage space remaining in kilobytes
batteryLevel		If present, estimated battery capacity percentage (+/- 10%). If not present, no battery available or failed to read battery capacity.
btHeadset		Is a bluetooth headset attached? 0 means false, 1 means true, not present means software may have it but doesn't support this JSON API.
btSensor		Is a bluetooth sensor attached? 0 means false, 1 means true, not present means software may have it but doesn't support this JSON API.
gpsAccuracy		Accuracy in meters. Not present if no sensor is currently reporting GNSS.
gpsLatitude		Latitude in degrees. Not present if no sensor is currently reporting GNSS.
gpsLongitude		Longitude in degrees. Not present if no sensor is currently reporting GNSS.
photoCount		The number of photos taken during the current burst or self timer repeat session
photosRemaining		Number of pictures that can be taken.
recordingTime		Amount of time in seconds that the video has been recording. Only specified if "state" = "recording"
recordingTimeRemaining		Total remaining time in seconds. If missing, unit was unable to calculate it (most likely card missing).
speed		Best estimate of speed we currently have in meters / second. Not present if no sensor is currently reporting speed.

Objects	Members	Description
state		Current state of the camera. Possible values include: "recording", "idle", "stillRecording", and "other"
totalSpace		Total storage space in kilobytes
wifiMode		Current WIFI state. Possible values include: "STA" /"AP"/"P2PCL"/"P2PGO"/"IBSS"/"INVALID"
wifiSensor		Is a WiFi sensor attached? 0 means false, 1 means true, not present means software may have it but doesn't support this JSON API.
wifiSignalStrength		Percentage estimate of WiFi signal strength. Currently only specified if WiFi is in "station" mode. This should not be relied upon to tell if the unit is in AP or station mode.
result		Result of the requested command. 1 = Success. 0 = Failure.

Example Success Response:

```
{
    "apiMin": 3,
    "apiMax": 3,
    "totalSpace": 15653888,
    "availableSpace": 7206560,
    "gpsAccuracy": 9,
    "gpsLatitude": 38.85656,
    "gpsLongitude": -94.799157,
    "speed": 0,
    "antSensor": 0,
    "btSensor": 0,
    "btHeadset": 0,
    "wifiSensor": 0,
    "recordingTimeRemaining": 2085,
    "photosRemaining": 1519,
    "photoCount": 0,
    "recordingTime": 0,
    "wifiMode": "AP",
    "state": "idle",
    "lastMediaEventTime": 12874,
    "result": 1
}
```

Example Failure Response:

```
{
"result": 0
}
```


stopRecording (3)

Format:

{"command":"stopRecording"}

Command Members:

No command members are use with this command.

Description:

Causes camera to stop recording the video.

Response Objects:

Objects	Members	Description
result		Result of the requested command. 1 = Success. 0 = Failure.

Example Success Response:

```
{
"result":1,
}
```

Example Failure Response:

```
{
    "result":0
}
```


stopStillRecording (3)

Format:

{"command":"stopStillRecording"}

Command Members:

No command members are use with this command.

Description:

Causes camera to stop recording still images.

Response Objects:

Objects	Members	Description
result		Result of the requested command. 1 = Success. 0 = Failure.

Example Success Response:

```
{
"result":1
}
```

Example Failure Response:

```
{
"result":0
}
```


updateFeature (3)

Format:

{"command":"updateFeature","feature": <feature> ,"value": <value> }

Command Members:

Objects	Values	Description
feature		Name of the feature.
value		Value to set the feature to. The available values are dependent on the feature to be updated.

Description:

Sets the value of a feature and retrieves the current values/options for the system.

Features are only returned if they are available. Some Response Objects are not returned depending on the type of feature and if the feature is enabled.

For a list of available features and values issue a features (3) command.

A complete list of all features and options can be found in the Available Features section.

Response Objects:

Objects	Members	Description
features[]	features_object	Array of features
features_object	enabled	Is the feature enabled (1) or disabled (0). Type 1 and 2 only.
	feature	Name of the feature.
	options[]	Array of strings representing the currently available options for the feature. This list can change based on current device state. Type 1 only.
	optionSummaries[]	Array of strings that provide a summary for each of the available options. Type 1 only.
	type	Type of feature. 0 = static feature, 1 = Feature contains list of values to be selected, 2 = Feature can only be set On (1)/Off (0)
	value	Current feature value. Type 1 and 2 only.
result		Result of the requested command. 1 = Success. 0 = Failure.

Example Success Response:

```
{
"features": [
    {
    "type": 0,
    "feature": "previewWhileRecording"
    },
```



```
    {
    "type": 1,
    "feature": "units",
    "enabled": 1,
    "value": "Statute",
    "options": [
        "Statute",
        "Metric"
        ]
    },
    ...
    ],
"result": 1
}
```

Example Failure Response:

```
{
"features": [
    {
    "type": 0,
    "feature": "previewWhileRecording"
    },
    {
    "type": 1,
    "feature": "units",
    "enabled": 1,
    "value": "Statute",
    "options": [
        "Statute",
        "Metric"
        ]
    },
    ...
    ],
"result": 0
}
```


Contact us

Questions and concerns regarding this documentation can be addressed by sending an email to `<virb.developer@garmin.com>`

FAQ's

A list of frequently asked questions can be found at http://developer.garmin.com/virb/help/

