ArduPilot JSON Protocol

This describes the protocol available with the ArduPilot web service on port 80.

Basic Format

Requests are http on port 80. Most replies are in JSON format. Requests are normally made with a POST request, but for the sake of convenience, many examples below are given as GET requests. This allows the examples to be pasted into a browser address bar easily.

MAVLink Protocol

Much of the data that can be requested originates in the MAVLink protocol used by ArduPilot. You can see the definition of the protocol in these two files:
https://github.com/ArduPilot/mavlink/blob/master/message_definitions/v1.0/common.xml
https://github.com/ArduPilot/mavlink/blob/master/message_definitions/v1.0/ardupilotmega.xml
note that the XML files define far more messages than ArduPilot supports, but when looking for the meaning of a field or a description of a message then the XML is useful.

ajax/command.json

Most commands are given via the URL ajax/command.json. You must pass at least one command argument to this URL. For example:

http://192.168.99.1/ajax/command.json?command1=uptime()

that example will return the time since boot of the sonix board in seconds. You may provide more than one command by setting command2, command3 etc.

You can see a table of all commands that can be passed to command.json here:

https://github.com/tridge/companion/blob/pr-web-server/web_server/functions.c#L959

Getting MAVLink messages

One of the most useful commands is to fetch a list of MAVLink messages as a JSON object. For example:

http://192.168.99.1/ajax/command.json?command1=mavlink_message(HEARTBEAT,ATTITUDE,SYS_STATUS,RAW_IMU,SCALED_PRESSURE,GPS_RAW_INT,GPS2_RAW,EKF_STATUS_REPORT,GLOBAL_POSITION_INT,RC_CHANNELS)

that will fetch the 10 listed MAVLink messages as a JSON reply like this:

{"HEARTBEAT" : { "type": 2 ,"autopilot": 3 ,"base_mode": 81 ,"custom_mode": 2 ,"system_status": 3 ,"mavlink_version": 3 },
"ATTITUDE" : { "time_boot_ms": 7506356 ,"roll": -2.801273 ,"pitch": -1.130834 ,"yaw": 0.591847 ,"rollspeed": 0.002925 ,"pitchspeed": -0.000523 ,"yawspeed": 0.000030 },
"SYS_STATUS" : { "onboard_control_sensors_present": 52493327 ,"onboard_control_sensors_enabled": 52476943 ,"onboard_control_sensors_health": 52493323 ,"load": 314 ,"voltage_battery": 4157 ,"current_battery": -1 ,"battery_remaining": -1 ,"drop_rate_comm": 0 ,"errors_comm": 0 ,"errors_count1": 0 ,"errors_count2": 0 ,"errors_count3": 0 ,"errors_count4": 0 },
"RAW_IMU" : { "time_usec": 3211388740 ,"xacc": -905 ,"yacc": 142 ,"zacc": 398 ,"xgyro": 2 ,"ygyro": 0 ,"zgyro": 0 ,"xmag": 0 ,"ymag": 0 ,"zmag": 0 },
"SCALED_PRESSURE" : { "time_boot_ms": 7506356 ,"press_abs": 962.6841 ,"press_diff": 0.942812 ,"temperature": 3892 },
"GPS_RAW_INT" : { "time_usec": 0 ,"fix_type": 0 ,"lat": 0 ,"lon": 0 ,"alt": 0 ,"eph": 9999 ,"epv": 0 ,"vel": 0 ,"cog": 0 ,"satellites_visible": 0 },
"GPS2_RAW" : { "time_usec": 0 ,"fix_type": 0 ,"lat": 0 ,"lon": 0 ,"alt": 0 ,"eph": 0 ,"epv": 0 ,"vel": 0 ,"cog": 0 ,"satellites_visible": 0 ,"dgps_numch": 0 ,"dgps_age": 0 },
"EKF_STATUS_REPORT" : { "flags": 165 ,"velocity_variance": 0.000000 ,"pos_horiz_variance": 0.001130 ,"pos_vert_variance": 0.002447 ,"compass_variance": 0.000000 ,"terrain_alt_variance": 0.000000 },
"GLOBAL_POSITION_INT" : { "time_boot_ms": 7506356 ,"lat": 0 ,"lon": 0 ,"alt": -11240 ,"relative_alt": -11240 ,"vx": 0 ,"vy": 0 ,"vz": -2 ,"hdg": 3391 },
"RC_CHANNELS" : { "time_boot_ms": 7506356 ,"chancount": 0 ,"chan1_raw": 0 ,"chan2_raw": 0 ,"chan3_raw": 0 ,"chan4_raw": 0 ,"chan5_raw": 0 ,"chan6_raw": 0 ,"chan7_raw": 0 ,"chan8_raw": 0 ,"chan9_raw": 0 ,"chan10_raw": 0 ,"chan11_raw": 0 ,"chan12_raw": 0 ,"chan13_raw": 0 ,"chan14_raw": 0 ,"chan15_raw": 0 ,"chan16_raw": 0 ,"chan17_raw": 0 ,"chan18_raw": 0 ,"rssi": 0 }}

Sending MAVLink messages

You can also send arbitrary MAVLink messages using mavlink_message_send(). For example:

http://192.168.99.1/ajax/command.json?command1=mavlink_message_send(PARAM_SET,0,0,SCHED_DEBUG,0)

that would set the parameter SCHED_DEBUG to zero. Note that SCHED_DEBUG is not an important parameter - it just sets a debug level in the ArduPilot scheduler. It is just used as an example (harmless) command.

Filesystem Interaction

You can interact with the vehicles filesystem via JSON command. You can upload files, download files, list files, remove files, rename files, create directories and get disk space and disk label information.
The following examples show how these are done:

- list directory:
  http://192.168.99.1/ajax/command.json?command1=file_listdir(/DATAFLASH)
- create directory:
  http://192.168.99.1/ajax/command.json?command1=file_mkdir(/mydir)
- remove file or directory:
  http://192.168.99.1/ajax/command.json?command1=file_unlink(/mydir)
- rename file or directory
  http://192.168.99.1/ajax/command.json?command1=file_rename(/mydir,/mydir2)
- return disk space and label information
  http://192.168.99.1/ajax/command.json?command1=disk_info()

Downloading files

To download a file, prefix the filename with “fs/”. For example
http://192.168.99.1/fs/DATAFLASH/000002.BIN
a download is plain GET command, with no arguments.

Uploading files

To upload a file, you need to use a multipart/form-data form. Please see
https://github.com/tridge/companion/blob/pr-web-server/web_server/files/upgrade.html
for details.

When uploading a file you must set the “uploadtype” variable to the type of upload. Valid values for uploadtype are:

- “fs” to upload to the filesystem
- “TX” to upload a new transmitter firmware
- “ArduPilot” to upload a new flight board firmware
- “sonix” to upload a new sonix firmware

ArduPilot Parameters

ArduPilot has around 900 user settable parameters. You can list the current parameter settings with this command:
http://192.168.99.1/ajax/command.json?command1=get_param_list()
Documentation for the parameters is here:
http://ardupilot.org/copter/docs/parameters.html
To set a parameter use the PARAM_SET MAVLink message, like this:
http://192.168.99.1/ajax/command.json?command1=mavlink_message_send(PARAM_SET,0,0,ANGLE_MAX,4500)
that would set the maximum lean angle to 45 degrees.

Fast GPS Lock

To supply MGA data for fast GPS lock use the mga_upload() command as a multipart form upload. You need to supply the following variables:

- mga_data: the mga blog (about 5kbytes)
- latitude: the approx latitude in degrees * 1e7
- longitude: the approx longitude in degrees * 1e7
- altitude: the approx altitude above sea level in cm
- utc_time: the current UTC time in seconds since 1970 (unix time)

See
https://github.com/tridge/companion/blob/pr-web-server/web_server/files/js/ublox.js for more details

Firmware Version

To find out the firmware versions use these commands:

- sonix firmware version:
  http://192.168.99.1/ajax/command.json?command1=sonix_version()
- ArduPilot firmware:
  http://192.168.99.1/ajax/command.json?command1=mavlink_message(AUTOPILOT_VERSION)

For the TX firmware version, use the vendor_id field of the AUTOPILOT_VERSION MAVLink message. That is encoded as year<<12 + month<<8 + day. The year starts at 2017.

Security

Security is based on two things:

- the WiFI password for the Sonix
- the Origin header in all HTTP requests

To maintain security you must supply a Origin HTTP header on all requests, like this:

Origin: http://192.168.99.1

this is required to prevent the Sonix from accepting requests from malicious web sites. Otherwise any website could access a users drone by embedding a bit of javascript (or even a creative img tag).
The web server will provide a way for users to select what origins it will accept.

Video Control

You can get a current jpg snapshot with this URL:
http://192.168.99.1/ajax/snapshot.jpg
an MJPG video stream is available as
http://192.168.99.1/ajax/video.mjpg
note that these video methods are not nearly as efficient as the existing H.264 streaming. The old methods still work and should be used when possible (they are not accessible in the browser based interface).

You can ask the sonix to take a photo to microSD using:

http://192.168.99.1/ajax/command.json?command1=take_picture()
you can ask the Sonix to toggle recording using:

http://192.168.99.1/ajax/command.json?command1=toggle_video()

Rebooting

You can reboot the flight board with:
http://192.168.99.1/ajax/command.json?command1=mavlink_command_long_send(MAV_CMD_PREFLIGHT_REBOOT_SHUTDOWN,0,1)
you can reboot the sonix video board with:
http://192.168.99.1/ajax/command.json?command1=reboot_companion()
(note that the sonix is a “companion” computer in ArduPilot nomenclature)

Example Code

The web interface serves as an example of all of the JSON commands that are available. Please see this URL for the current javascript/html code:
https://github.com/tridge/companion/tree/pr-web-server/web_server/files

Parameter Control

To fetch all flight parameters from the vehicle, use this JSON request:
http://192.168.99.1/ajax/command.json?command1=get_param_list()
You can also fetch a list of parameters with a specific set of prefixes by giving them as arguments. For example, to fetch all motor and INS parameters use this:

http://192.168.99.1/ajax/command.json?command1=get_param_list(MOT,INS)
To fetch a single parameter just give it as a single argument:
http://192.168.99.1/ajax/command.json?command1=get_param_list(FENCE_RADIUS)
that will fetch the FENCE_RADIUS parameter.
The reply is a JSON array, like this:
[{ "name" : "FENCE_RADIUS", "value" : 100.0000 }]

To set a parameter send the PARAM_SET message, like this:
http://192.168.99.1/ajax/command.json?command1=mavlink_message_send(PARAM_SET,0,0,FENCE_RADIUS,200)
that will set the FENCE_RADIUS parameter to 200. Note that parameter changes are persistent - they will survive a reboot.

Geo-fence Control

The geo-fence is automatically enabled in GPS modes, and disabled in non-GPS modes. To control the radius of the fence set the FENCE_RADIUS parameter using the above parameter control commands. The FENCE_RADIUS is in meters.
You can also set the maximum fence altitude with the FENCE_MAXALT parameter, also in meters. The FENCE_RADIUS defaults to 100 meters and the FENCE_MAXALT defaults to 50 meters.
The geo-fence is a “tin-can” type of fence, a cylinder centered around home.

Home Position

The home position is used for the RTL location, and also as the center of the geo-fence. To get the home position fetch the HOME_POSITION mavlink message:
http://192.168.99.1/ajax/command.json?command1=mavlink_message(HOME_POSITION)
the message will only be available if the copter has had GPS lock.
The output will look like this:
{"HOME_POSITION" : { "latitude": -353632608 ,"longitude": 1491652351 ,"altitude": 584100 ,"x": 0.000000 ,"y": 0.000000 ,"z": 0.000000 ,"q": [1.000000, 0.000000, 0.000000, 0.000000] ,"approach_x": 0.000000 ,"approach_y": 0.000000 ,"approach_z": 0.000000 ,"_seq" : 145, "_sysid" : 1, "_compid" : 1, "_age" : 8758}}
the key fields are latitude and longitude, which are in degrees times 10^7 (10 million), and altitude, which is in millimeters above sea level.

Setting Home Position

You can also set the home position. This can be used to move the geo-fence center when the pilot moves, by sending the position of their device running the app.
To set home position you should use a DO_SET_HOME command with a frame type of MAV_FRAME_GLOBAL_RELATIVE_ALT. The advantage of using this command is you can

set the relative_alt to zero, which will not change the height of home, making it safe to use even when the altitude estimate of the device and the drone are different.
To send that command use the following:
http://192.168.99.1/ajax/command.json?command1=mavlink_message_send(COMMAND_INT,0,0,3,179,1,0,0,0,0,0,-353600000,1491600000,0)
or written out more clearly with named constants it would be:

- MAV_CMD_DO_SET_HOME=179
- MAV_FRAME_GLOBAL_RELATIVE_ALT=3
- POS_SCALE_INTEGER=10000000
- NEW_LATITUDE=-35.36
- NEW_LONGITUDE=149.16
- ALTITUDE_CHANGE=0
- http://192.168.99.1/ajax/command.json?command1=mavlink_message_send(COMMAND_INT,0,0,MAV_FRAME_GLOBAL_RELATIVE_ALT,MAV_CMD_DO_SET_HOME,,1,0,0,0,0,0,NEW_LATITUDE*POS_SCALE_INTEGER,NEW_LONGITUDE*POS_SCALE_INTEGER,ALTITUDE_CHANGE)

Note that the units are not the same as for fetching the HOME_POSITION. The units for latitude and longitude are in degrees times 10^7 (which is what POS_SCALE_INTEGER is for). The units for the relative altitude is meters.
If you send a DO_SET_HOME like the above when the copter does not yet have full GPS lock then it will be ignored and no change will be made to HOME.

Flight Mode

To get the current flight mode use HEARTBEAT.custom_mode field. The values are:

- STABILIZE = 0, // manual airframe angle with manual throttle
- ACRO = 1, // manual body-frame angular rate with manual throttle
- ALT_HOLD = 2, // manual airframe angle with automatic throttle
- AUTO = 3, // fully automatic waypoint control using mission commands
- GUIDED = 4, // fully automatic fly to coordinate or fly at velocity/direction using GCS immediate commands
- LOITER = 5, // automatic horizontal acceleration with automatic throttle
- RTL = 6, // automatic return to launching point
- CIRCLE = 7, // automatic circular flight with automatic throttle
- LAND = 9, // automatic landing with horizontal position control
- DRIFT = 11, // semi-automous position, yaw and throttle control
- SPORT = 13, // manual earth-frame angular rate control with manual throttle
- FLIP = 14, // automatically flip the vehicle on the roll axis
- AUTOTUNE = 15, // automatically tune the vehicle's roll and pitch gains

- POSHOLD = 16, // automatic position hold with manual override, with automatic throttle
- BRAKE = 17, // full-brake using inertial/GPS system, no pilot input
- THROW = 18, // throw to launch mode using inertial/GPS system, no pilot input
- AVOID_ADSB = 19, // automatic avoidance of obstacles in the macro scale - e.g. full-sized aircraft
- GUIDED_NOGPS = 20, // guided mode but only accepts attitude and altitude

To set the flight mode use the set_mode mavlink command, like this:
http://192.168.99.1/ajax/command.json?command1=mavlink_message_send(SET_MODE,0,1,6)
that sets mode 6, which is RTL.

Photo and Video Events

To detect the user pressing photo or video buttons, the app can look for the NAMED_VALUE_INT mavlink message:
http://192.168.99.1/ajax/command.json?command1=mavlink_message(NAMED_VALUE_INT)

that message will look like this:

"NAMED_VALUE_INT" : { "time_boot_ms": 161985 ,"name": "SNAPSHOT" ,"value": 1 ,"_seq" : 72, "_sysid" : 1, "_compid" : 1, "_age" : 51704}}

the 'name' field will be one of the following:

- "SNAPSHOT" for when the user has pressed the photo button
- "VIDEOTOG" for when the user has pressed the video button (it means "Video Toggle")
- "WIFIRESET" if the user has triggered a WiFi password reset ("turtle" reset)

you can use the time_boot_ms field (time since boot in milliseconds of the flight board) to detect if it is a new message.
The above message can be used in combination with the uart_data.is_recording_video flag to know if video should currently be recorded.

You can tell the TX to beep to notify of video recording by the app by sending a VNOTIFY message like this:
http://192.168.99.1/ajax/command.json?command1=mavlink_message_send(NAMED_VALUE_INT,0,VNOTIFY,1)
the same VNOTIFY message is used for photos and videos. For videos you should send the VNOTIFY at more than 1Hz (preferably 2Hz to account for some network lag) when recording video using the app video button.
You don't need to send VNOTIFY messages if the photo or video is triggered using the TX buttons.

Sending Arbitrary Tunes

You can play arbitrary RTTTL tunes on the TX. You can test this feature using this page:
http://192.168.99.1/devtools.html
See that page for documentation on the tune string format. To send a tune, use this command:
http://192.168.99.1/ajax/command.json?command1=play_tx_tune(:d=8,o=6,b=480:a,d7,c7,a,d7,c7)
that will send the first part of the TX startup tune.

Getting and Setting WiFi Info

You can fetch the current WiFI SSID, password etc like this
http://192.168.99.1/ajax/command.json?command1=get_ssid_info()
That will return JSON data like this:
{ "ssid": "SKYVIPERGPS_14843B", "password" : "vipergps", "channel" : 9, "authmode" : 3 }
To set the SSID, password, authmode and channel, use a command like this:
http://192.168.99.1/ajax/command.json?command1=set_ssid(SKYVIPER_TEST1,vipergps,3,7)
where 7 is the channel, and authmode 3 is WPA2
Due to limitations in the parsing of json commands you won't be able to support a comma in the password. It is probably best to limit the character set to upper and lower case letters and digits, plus a small set of other characters such as #, @, and *