
# ROBIN
## Tech Note

Version: 3.6.0 Robin NL
Datum: 24-04-2019

# How-To: The Robin API

# About this Tech Note

This Tech Note applies to the following Robin products:

- Robin SmartView SIP with 1, 2, 4, 6 buttons
- Robin Compact SIP
- Robin ProLine SIP with 1, 2, 4 buttons
- Robin ClassicLine SIP with 1, 2, 4 buttons
- Robin ProLine SIP with Keypad

***Software release 3.6.7 and higher***

# Remote Control

The Robin is controllable via an API (application programming interface). This enables the use of HTTP commands to control the Robin and to retrieve data from the Robin such as: call status, streaming video and version info.

You can use:

1. Directly, with HTTP commands (eg. with cURL)
2. The 'Events' option in the web interface

The API functionality is disabled by default. To enable it, log in to the Robin and enable the API with the 'Enable API interface' option in the menu -System-Security. Our advise is to keep this feature disabled when not using the API.

The API does not have to be enabled to retrieve a MJPEG video stream or JPEG video stream, these can always be accessed.

All executed API calls will be logged on the Robin. The logfile is visible in the menu -System-Logs-.

# Control the Robin with HTTP commands

The Robin SmartView is controllable with HTTP-commands and it is also possible to retrieve data.

A set of commonly used commands are shown below, in these examples with cURL-commands.
To use these commands, use ether the login credentials of the 'admin' user or use an Authentication Token. Create a token in the menu -System-Security- of the Robin.

*Login credentials:* **`curl -u admin:<password>`**

*Authentication token:* **`curl -H 'Sessionid: <generated token>'`**
.

***! Note: cURL is only one of many ways to send HTTP-commands. For more information about cURL, see: http://en.wikipedia.org/wiki/CURL !***

# Access to the MJPEG-video stream

*- Only available on Robin products with built-in camera -*

The Robin with a built-in camera generates a Motion-JPEG video stream that can be retrieved with the following command:

```
curl -u admin:<password> "http://<address>/pbhelper/stream.mjpeg"
```

- <password> = the password for the -admin- account
- <address> = the Robin SV address

***! Note: Multiple simultaneous MJPEG video streams can affect the performance of the Robin negatively. !***

You can also set the preferred resolution and compression level, just add the following to the URL:

- `?width=320&height=240` = resolution for the requested video (320x240 pixels) **- MAX resolution = 720x576**
- `?quality=80` = JPEG quality level (80) **- MAX JPEG quality level = 100**

***! Note: Low resolution and low MJPEG quality will both result in low bandwidth usage. High resolution and high MJPEG quality will both result in high bandwidth usage. !***

# Access to the latest video frame

*- Only available on Robin products with built-in camera -*

The Robin with a built-in camera not only generates a Motion-JPEG video stream but also individual video frames (JPEG). These frames can be retrieved by using the following command:

```
curl -u admin:<password> "http://<address>/camera/frame.jpeg"
```

- <password> = the password for the -admin- account
- <address> = the Robin SV address

You can also set the preferred resolution and compression level, just add the following to the URL:

- `?width=320&height=240` = resolution for the requested video (320x240 pixels) **- MAX resolution = 720x576**
- `?quality=80` = JPEG quality level (80) **- MAX JPEG quality level = 100**

***! Note: Low resolution and low JPEG quality will both result in low bandwidth usage. High resolution and high JPEG quality will both result in high bandwidth usage. !***

# Making calls using HTTP commands

You can control the Robin to make calls, disconnect calls and retrieve the current call status. The Robin will answer to these commands in JSON.

**Command:** `curl -u admin:<password> "http://<address>/api/v1/call _ status"`
**Goal:** Retrieve the call status of the Robin
**Answer:**

```
{
"rv" : {
"status" : "outgoing call to 501"
},
 "ok" : true
}
```

**Command:** `curl -u admin:<password> "http://<address>/api/v1/call _ hangup"`
**Goal:** disconnect the call (call hangup)
**Answer:**

```
{
"ok" : true
}
```

**Command:** `curl -u admin:<password> "http://<address>/api/v1/api _ ping"`
**Goal:** ping the Robin (to check if it responds)
**Answer:**

```
{
 "rv" : {
 "pong" : "pong"
 },
"ok" : true
}
```

***! Note: If the HTTP command to dial an extension contains an extension that is already known in the phone book, the Robin dials that extension and will use the profile that is defined for that extension. !***

# Control the audio mute function using HTTP commands

The audio settings of the Robin can be controled using http commands. The commands control the mute function of the Robin. It mutes speakers of the Robin.
You can mute three types of audio:

- All incoming tones: Mute all tones from an incomming phone call
- All tones: Mute all tones from incomming and outgoing phone calls
- All audio: Mute the speakers of the Robin. It mutes all tones and voice. Use this option to listen-in on the Robin.

The audio mute can also be disabled using one command.
The Robin will answer to these commands in JSON.

**Command:** `curl -u admin:<password> "http://<address>/api/v1/mute_set?mute=tones_incoming"`
**Goal:** mute all tones from an incomming phone call
**Answer:**

```
{
 "rv" : {
 "mute":"tones_incoming"
 },
"ok" : true
}
```

**Command:** `curl -u admin:<password> "http://<address>/api/v1/mute_set?mute=tones_all"`
**Goal:** mute all tones from incomming and outgoing phone calls
**Answer:**

```
{
 "rv" : {
 "mute":"tones_all"
 },
"ok" : true
}
```

**Command:** `curl -u admin:<password> "http://<address>/api/v1/mute_set?mute=all_audio"`
**Goal:** mute the speakers of the Robin. (mutes all tones and voice)
**Answer:**

```
{
 "rv" : {
 "mute":"all_audio"
 },
"ok" : true
}
```

**Command:** `curl -u admin:<password> "http://<address>/api/v1/mute _ set?mute=off"`
**Goal:** disable the audio mute (all sound will be on)
**Answer:**

```
{
 "rv" : {
 "mute":"off"
 },
"ok" : true
}
```

**Command:** `curl -u admin:<password> "http://<address>/api/v1/mute _ get"`
**Goal:** retrieve the audio mute status
**Answer:**

```
{
 "rv" : {
 "mute":"off" / :"all_audio" / :"tones_all" / :"tones_incoming"
 },
"ok" : true
}
```

# Control the door opener using HTTP commands

*- Only available on the Robin SIP and Robin SV -*

You can control the Robin to open and close the door and retrieve the status of the door opener. The Robin will answer to these commands in JSON.

**Command:** `curl -u admin:<password> "http://<address>/api/v1/dooropener_open"`
**Goal:** Open the door and keep it open (close the relay)
**Answer:**
```
{
"ok" : true
}
```

**Command:** `curl -u admin:<password> "http://<address>/api/v1/dooropener_close"`
**Goal:** Close the door (open the relay)
**Answer:**
```
{
"ok" : true
}
```

**Command:** `curl -u admin:<password> "http://<address>/api/v1/dooropener_pulse"`
**Goal:** Open the door and keep it open for the time set in the GUI. After that, close the door
**Answer:**
```
{
"ok" : true
}
```

**Command:** `curl -u admin:<password> "http://<address>/api/v1/dooropener_status"`
**Goal:** Retrieve the status of the door opener
**Answer:**
```
{
"rv" : {
"status" : "Open"
},
 "ok" : true
}
```

# Configure the SIP-configuration using HTTP commands

The SIP configuration of the Robin can be programmed and read using HTTP commands.

The Robin will answer to these commands in JSON.

**Command:** `curl -u admin:<password> "http://<address>/api/v1/get_button_count"`
**Goal:** Retrieve the amount of available buttons on the Robin (1,2,4 or 6) -> Answer=1
**Answer:**
{
"ok":true,"rv":1
}

**Command:** `curl -u admin:<password> "http://<address>/api/v1/get_phonebook"`
**Goal:** Retrieve the phone book data (in this example: no. 1001, 1002 and 1003)
**Answer:**
{"ok":true,"rv":[
{"profile":"---","index":1,"description":"Bria","extension":"1003",
"allow_register":false},
{"profile":"---","index":2,"description":"Yealink_1","extension":"1001",
"allow_register":false},
{"profile":"---","index":3,"description":"Yealink_2","extension":"1002",
"allow_register":false}]}

**Command:** `curl -u admin:<password> "http://<address>/api/v1/add_phonebook_entry?description=Reception&number=101&allow_register=0"`
**Goal:** Add a Phone book entry. In this example: Description=Reception, number=101, Allow register (Peer to Peer) disabled (0)
**Answer:**
{
"ok":true,"rv":[]
}

**Command:** `curl -u admin:<password> "http://<address>/api/v1/delete_phonebook_entry?index=1"`
**Goal:** Remove entry no. 1 from the phone book.
**Answer:**
{
"ok":true,"rv":[]
}

***After removal of a phone book entry, the numbering of the remaining entries will change. Use the 'get_phonebook' command to retrieve the updated entries and numbers.***

**Command:** `curl -u admin:<password> "http://<address>/api/v1/get_callprio_list?button=1"`
**Goal:** Retrieve the call list (in this example: 1001, 1002 en 1003, for button 1)
**Answer:**
{"ok":true,"rv":[
{"name":"Yealink_1","number":"1001"},
{"name":"Yealink_2","number":"1002"},
{"name":"Bria","number":"1003"}
]}

**Command:** `curl -u admin:<password> "http://<address>/api/v1/set_callprio_list?first=name1&second=name2&third=name3&button=2"`
**Goal:** Configure the call list. In this example: First=name1, Second=name2, Third=name3, this is the call list for button no. 2.
**Answer:**
{
"ok":true,"rv":[]
}

**Command:** `curl -u admin:<password> "http://<address>/api/v1/set_sip_config?proto=udp&host=192.168.200.202&port=5060&username=test&password=secret&register=1"`
**Goal:** Configure the SIP settings of the Robin. In this example: SIP protocol=UDP, SIP proxy / Registar=192.168.200.202, SIP proxy port number=5060, Username=test, Password=secret, Register enabled (1).
**Answer:**
{
"ok":true,"rv":[]
}

**Command:** `curl -u admin:<password> "http://<address>/api/v1/get_sip_config"`
**Goal:** Retrieve the SIP configuration from the Robin
**Answer:**
{"ok":true,"rv":
{"proto":"udp","secondary_port":5060,"secondary_host":"ip or hostname",
"host":"192.168.200.202","port":5060,"expires":3600,"register":true,
"password":"secret","username":"test","secondary":false}}

# Upgrade software version using HTTP commands

The software of the Robin can be upgraded using HTTP commands. It takes eight steps to upgrade the Robin.

The Robin will answer to these commands in JSON.

***! Note: Use an interval time of 5 seconds between each command. !***

1. *Check for a new software version:*

   **Command:** `curl -u admin:<password> "http://<address>/api/v1/do _ update"`
   **Goal:** Start the check for a new software version.
   **Answer:** {
   "ok":true,"rv":[]
   }

2. *Check the status during the* **update command***. Continue checking until the output is 'empty':*

   **Command:** `curl -u admin:<password> "http://<address>/api/v1/get _ upgrade _ status"`
   **Goal:** Retrieve the actual status during the update check.
   **Answer:** {
   "ok":true,"rv":"Checking for new versions..." -> busy
   or "ok":true,"rv":"" -> done
   }

3. *Retrieve the software versions. Is the "version_available" identical to the "version_installed"? No update available. Is the "version_available" higher than the "version_installed"? New update available, see step 4:*

   **Command:** `curl -u admin:<password> "http://<address>/api/v1/get _ versions"`
   **Goal:** Retrieve the software versions.
   **Answer:** {
   "ok":true,"rv":{
   "version_available":"dev+3870",
   "version_installed":"dev+3869",
   "version_running":"dev+3869"
   }
   }

4. *Upgrade to the latest version:*

   **Command:** `curl -u admin:<password> "http://<address>/api/v1/do _ upgrade"`
   **Goal:** Start the software upgrade.
   **Answer:** {
   "ok":true,"rv":[]
   }

5. *Check the status during the* **upgrade command**. *Continue checking until the output is 'Ok' and the upgrade is finished:*

   **Command:** `curl -u admin:<password> "http://<address>/api/v1/get _ upgrade _ status"`
   **Goal:** Retrieve the actual status during the software upgrade.
   **Answer:** {
   "ok":true,"rv":"Upgrading..." -> busy
   or "ok":true,"rv":"Ok" -> done
   }

6. *As soon as the upgrade is done, check the status of the upgrade using the '*get_versions' *command. Is the "version_installed" higher than the "version running"? The upgrade was successful and the Robin needs to reboot. See step 7:*

   **Command:** `curl -u admin:<password> "http://<address>/api/v1/get _ versions"`
   **Goal:** Retrieve the software versions.
   **Answer:** {
   "ok":true,"rv":{
   "version_available":"dev+3870",
   "version_installed":"dev+3870",
   "version_running":"dev+3869"
   }
   }

7. *Reboot the Robin, the reboot procedure can take up to one minute:*

   **Command:** `curl -u admin:<password> "http://<address>/api/v1/reboot"`
   **Goal:** Reboor the Robin
   **Answer:** {
   "ok":true,"rv":[]
   }

8. *After the reboot, use the* 'get_versions' *command to check if the upgrade. Is the "version_running" identical to the "version_installed" and is the "version_available" empty? The upgrade went successfully:*

**Command:** `curl -u admin:<password> "http://<address>/api/v1/get _ versions"`
**Goal:** Retrieve the software versions.
**Answer:**

{
"ok":true,"rv":{
"version_available":"",
"version_installed":"dev+3870",
"version_running":"dev+3870"
}
}

# Miscellaneous settings using HTTP commands

Use HTTP commands to make miscellaneous settings.

**Command:** `curl -u admin:<password> "http://<address>/api/v1/reset_defaults"`
**Goal:** Restore the factory settings. The Robin will automatically reboot.
**Answer:**
```
{
"ok":true,"rv":[]
}
```

# Recording feature - HTTP commands

The recording feature of the Robin can be controlled using HTTP commands.

The Robin will answer to these commands in JSON.

**video_list**

The 'video_list' call retrieves a list with all recorded videos. Each entry contains the following fields:

- time: start time of the video (unix timestamp)
- duration: in seconds
- id: video id, a unique string for this video
- url: the url of the videostream
- event_data: the event data of the event that triggered the recording

Arguments:

- button: filters on button number (optional)

**Command:** `curl -u admin:<password> "http://<address>/api/v1/video_list?button=3"`
**Goal:** Retrieve the video list for a specific button - in this case button 3
**Answer:**


```
{
"ok": true,"rv": {"video_list": [
{"call_ids": [
"c3aa4538"
],
"time": 1394113333,
"id": "2",
"duration": 300,
"url" : "/recording/video-2.m3u8",
"event_data": {
"button": 3,
"type": "button"},
{
"call_ids": [
"970fdfe9"
],
"time": 1394113375,
"id": "5",
"duration": 300,
"url" : "/recording/video-5.m3u8",
"event_data": {
"button": 3,
"type": "button"}
]
}
}
```

### Play HLS video

The video file can be accessed via the m3u8 url.

**Command:** `curl -u admin:<password> "http://<address>/recording/video-5.m3u8"`
**Goal:** Retrieve the video - In this case it contans four segments
**Answer:**
```
#EXTM3U
#EXT-X-TARGETDURATION:10
#EXT-X-MEDIA-SEQUENCE:0
#EXTINF:10, no desc
/recording//2014/03/06/14/20140306144219_00000002_10.ts
#EXTINF:10, no desc
/recording//2014/03/06/14/20140306144229_00000003_10.ts
#EXTINF:8, no desc
/recording//2014/03/06/14/20140306144239_00000004_8.ts
#EXTINF:10, no desc
/recording//2014/03/06/14/20140306144247_00000005_10.ts
#EXT-X-ENDLIST
```

### video_del

Remove a specific video.

Arguments:

- id: id of the video (see video_list)

**Command:** `curl -u admin:<password> "http://<address>/api/v1/video _ del?id=6"`
**Goal:** Remove the video with -in this case- id: 6
**Answer:**
```
{
"ok":true,"rv":[]
}
```

**call_log**

Retrieve the call log.

Arguments:

- id: id of a specific call

**Command:** `curl -u admin:<password> "http://<address>/api/v1/call _ log"`
**Goal:** Retrieve the call log
**Answer:**

{
"ok": true,"rv": {"call_log": [
{
"number": "101@10.0.0.26",
"direction": "incoming",
"time": 1386753209,
"id": "55826c80",
"duration": 44
"result":"hangup"
"answered":true},
{
"number": "101@10.0.0.26",
"direction": "incoming",
"time": 1386753970,
"id": "19b46a5f",
"duration": 10
"result":"busy"
"answered":false},
{
"number": "101@10.0.0.26",
"direction": "incoming",
"time": 1386754004,
"id": "483ae77c",
"duration": 3
"result":"error"
"answered":false}
]
}
}

# Control the Robin using Events

The 'Events' method can be used to control functions of the Robin with custom commands. Use this if the direct HTTP commands don't fit your needs.

The first step is to create a 'Source', this will be the activator of the event.
The second step is to create an 'Action' that will follow the 'Source'.

Log on to the Robin web interface and navigate to: -System-Events-, click on 'Add source'.


- Define a name for 'Source', eg. 'Demo'.
- Select as 'Type': Http
- Create a unique Http-path, eg. 'demo'
- Apply settings

The HTTP request to use will be:

**http://<address>/evmgr/<path>**

- <address> = the address of the Robin
- evmgr = event manager
- <path> = the path that is defined in 'Source'

Next, create an ‘Action’ to go with the ‘Source’. To do this, click on ‘Add action’.


- Define a name for this ‘Action’
- As ‘Event’, select the just created ‘Source’
- As ‘Type’, select ‘Call’ (the Robin will start an outgoing call when triggered)
- As ‘Call extension’, select the default extension to dial
- Apply settings

If the following HTTP request is received by the Robin:

**http://<address>/evmgr/demo**

The Robin will dial the default extension that is set in the ‘Action’.