OVERVIEW

This document specifies the external API of Bird Home Automation products. The interface provides the functionality for IP communicating with Bird Home Automation products via LAN (Local Area Network), handled by the built-in servers in Bird Home Automation products.

COPYRIGHT NOTICE

This document is copyright protected and is the property of Bird Home Automation GmbH and may not be copied, reproduced or distributed in any way without the prior written consent of Bird Home Automation GmbH.

BETA AND SUPPORT NOTICE

This API is currently in beta status and is subject to change without prior notice. This API is not part of the product purchase and thus comes without any kind of support and without any kind of warranty.

LICENSE AGREEMENT

This License Agreement (“License”) is a legal agreement between you (either individual or an entity) and Bird Home Automation GmbH (“Bird Home Automation”). By using the INTERFACE and INTERFACE DESCRIPTION (each defined below), whether in whole or in part, you agree to be bound by the terms of this License.

1. GRANT OF LICENSE

Bird Home Automation hereby grants to you the right to use BIRD HOME AUTOMATION LAN API (“INTERFACE”) and the written specification of the INTERFACE (the “INTERFACE DESCRIPTION”) for the sole and limited purpose of creating, manufacturing and developing a solution that integrates any unit or portion included in the product range of Bird Home Automation network products, as defined by Bird Home Automation at its discretion (an “Bird Home Automation Product”) and to market, sell and distribute any such solution.

2. COPYRIGHT

The INTERFACE and the INTERFACE DESCRIPTION are owned by Bird Home Automation and are protected by copyright laws and international treaty provisions. Any use of the INTERFACE and/or the INTERFACE DESCRIPTION outside the limited purpose set forth in Section 1 above is strictly prohibited.

3. RESTRICTIONS ON USE

You have no rights with respect to the INTERFACE, INTERFACE DESCRIPTION or
any portions thereof and shall not use the INTERFACE, INTERFACE DESCRIPTION or any portion thereof except as expressly set forth herein. You may not reverse engineer, decompile, or disassemble the INTERFACE except to the extent required to obtain interoperability with other independently created computer programs as permitted by mandatory law.

4. THIRD PARTY RIGHTS

You agree that you are fully responsible for your own conduct while using the INTERFACE and integrating any Bird Home Automation Products into your solution and the consequences thereof. Bird Home Automation Products may be combined with a virtually infinite number of potential solutions. Consequently, you recognize that (i) other third parties may claim to own patents or copyrights that could cover certain solutions which integrate Bird Home Automation products, or which result from the combination of Bird Home Automation products and additional technology or solutions and (ii) you are responsible for ensuring that any solution which integrates with an Bird Home Automation Product, or a combination of a solution and an Bird Home Automation product, does not infringe upon or misappropriate any intellectual property or personal right of any third party.

5. TERMINATION

This License is effective until terminated. Your rights under this License will terminate automatically without notice from Bird Home Automation if you fail to comply with any term(s) of this License. Upon the termination of this License, you shall cease all use and disposition of the INTERFACE and/or THE INTERFACE DESCRIPTION whether for the purpose set forth in Section 1 above or not.

6. REPRESENTATIONS AND WARRANTIES; DISCLAIMER

6.1. You represent and warrant that (i) any solution created, manufactured and/or developed by you which integrates an Bird Home Automation Product shall not infringe or otherwise violate any third party rights, including but not limited to third party intellectual property rights; and (ii) your use of the INTERFACE and INTERFACE DESCRIPTION will comply with all applicable foreign and domestic laws, rules and regulations.

6.2. YOUR USE OF THE INTERFACE IS AT YOUR SOLE RISK. THE INTERFACE AND THE INTERFACE DESCRIPTION ARE DELIVERED FREE OF CHARGE AND "AS IS" WITHOUT WARRANTY OF ANY KIND. THE ENTIRE RISK AS TO THE USE, RESULTS AND PERFORMANCE OF THE INTERFACE AND THE INTERFACE DESCRIPTION IS ASSUMED BY THE USER/YOU. BIRD HOME AUTOMATION DISCLAIMS ALL WARRANTIES, WHETHER EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, TITLE, NON-INFRINGEMENT AND PRODUCT LIABILITY, OR ANY WARRANTY ARISING OUT OF ANY PROPOSAL, SPECIFICATION OR SAMPLE WITH RESPECT TO THE INTERFACE AND THE INTERFACE DESCRIPTION.

Without limiting the generality of the foregoing, you acknowledge and agree that Bird Home Automation does not make any representation or warranty that the integration of Bird Home Automation Products into your solution does not infringe any third party rights. You are solely responsible for any intellectual property infringement claims that are based on or relate to solutions created, manufactured and distributed by you which integrate Bird Home Automation Products. Bird Home Automation is unaware.
of the details regarding your particular solution, has not conducted any investigation relating to potential third party rights issues relating to your solution and does not accept any responsibility or liability with respect thereto.

6.3. THIS LICENSE DOES NOT CONVEY ANY LICENSE TO THIRD PARTY INTELLECTUAL PROPERTY. YOU ARE SOLELY RESPONSIBLE FOR (I) EXAMINING WHETHER THE INTERFACE AND THE INTERFACE DESCRIPTION ARE ENCUMBERED BY OR INFRINGES UPON A RIGHT HELD BY A THIRD PARTY AND (II) ANY INTELLECTUAL PROPERTY INFRINGEMENT CLAIMS THAT ARISE OUT OF OR RELATE TO SOLUTIONS CREATED, MANUFACTURED AND DISTRIBUTED BY YOU WHICH INTEGRATE BIRD HOME AUTOMATION PRODUCTS.

**DOCUMENT HISTORY**

<table>
<thead>
<tr>
<th>Version</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.16</td>
<td>Add information about Digest Authentication. Describe “test” parameter for notification requests. Change example to show HTTPS support at notifications. Update web links at AEC/ANR hint.</td>
</tr>
<tr>
<td>0.17</td>
<td>History images are stored in the cloud.</td>
</tr>
<tr>
<td>0.18</td>
<td>Added document history. Add DoorBird D21x to compatible devices. Describe permission check and RC 204 for several requests: live video stream, live image, live audio transmit/receive, open door, light on, history image. Remove outdated chapters for notification, monitor and check requests. Add information about the “API operator” permission and where it's necessary. Add chapters for UDP events, favorites, schedules and peripherals. Extend info.cgi chapter (MAC address, device type and relays). Extend description of history.image requests (permission check). Include information about the SIP API. SIP API now handles the “API operator” permission, property “autocall_doorbell_url” is now deprecated.</td>
</tr>
<tr>
<td>0.19</td>
<td>Add generic information about the integration scheme. Describe more response codes from favorites.cgi and schedules.cgi.</td>
</tr>
<tr>
<td>0.20</td>
<td>Add monitor.cgi chapter.</td>
</tr>
<tr>
<td>0.21</td>
<td>Add more examples to the “schedule management” chapter.</td>
</tr>
<tr>
<td>0.22</td>
<td>Updated app url schemes for preventing full-screen on landscape mode.</td>
</tr>
<tr>
<td>0.23</td>
<td>Add restart.cgi chapter. Change some doorbird.com links to HTTPS.</td>
</tr>
<tr>
<td>Task Number</td>
<td>Description</td>
</tr>
<tr>
<td>-------------</td>
<td>-------------</td>
</tr>
<tr>
<td>0.24</td>
<td>Fix description and example of time intervals for weekly schedules at the “schedules management” chapter.</td>
</tr>
<tr>
<td>0.25</td>
<td>Add “encryption” chapter for the HTTP interface.</td>
</tr>
<tr>
<td>0.26</td>
<td>Add more information to “EVENT MONITORING (UDP BROADCASTS)”. Specify SSL certificate handling for HTTPS URL favorites.</td>
</tr>
<tr>
<td>0.27</td>
<td>Add example of gstreamer for audio-transmit.cgi.</td>
</tr>
<tr>
<td>0.28</td>
<td>Remove input parameter for schedules.</td>
</tr>
<tr>
<td>0.29</td>
<td>Add DoorBird D11x to compatible devices and new path for rtsp with 720p and 1080p.</td>
</tr>
<tr>
<td>0.30</td>
<td>Corrected watch-always time to 1 minute and updated SIP section with current screenshot.</td>
</tr>
<tr>
<td>0.31</td>
<td>Added fingerprint to schedule-input.</td>
</tr>
<tr>
<td>0.32</td>
<td>Include “ring_time_limit” and “call_time_limit” information for the SIP API.</td>
</tr>
<tr>
<td>0.33</td>
<td>Added curl example call to audio-transmit.cgi documentation.</td>
</tr>
<tr>
<td>0.34</td>
<td>Replaced deprecated event monitoring by documentation of new version - “EVENT MONITORING (UDP BROADCASTS)”, added new information about rate limits for wrong authentications. Added rate limit information in section “CONCURRENT CONNECTIONS AND RATE LIMITS”.</td>
</tr>
<tr>
<td>0.35</td>
<td>Fixed example packets for local UDP notifications (version 2).</td>
</tr>
<tr>
<td>0.36</td>
<td>Added information about key length of UDP notifications</td>
</tr>
</tbody>
</table>

### COMPATIBLE DEVICES

<table>
<thead>
<tr>
<th>Device</th>
<th>Hardware version</th>
<th>Firmware version</th>
</tr>
</thead>
<tbody>
<tr>
<td>DoorBird Video Door Station D10x</td>
<td>1.00 and above</td>
<td>000099 and above</td>
</tr>
<tr>
<td>DoorBird Video Door Station D20x</td>
<td>1.00 and above</td>
<td>000099 and above</td>
</tr>
<tr>
<td>DoorBird Video Door Station D21x</td>
<td>1.00 and above</td>
<td>000108 and above</td>
</tr>
<tr>
<td>BirdGuard B10x</td>
<td>1.00 and above</td>
<td>000099 and above</td>
</tr>
<tr>
<td>DoorBird Video Door Station D11x</td>
<td>1.00 and above</td>
<td>000130 and above</td>
</tr>
</tbody>
</table>

Please check also the Firmware Change Log at [https://www.doorbird.com/changelog](https://www.doorbird.com/changelog) and check the descriptions of firmware updates.

For reasons of brevity, we use the terms “device” to refer to our “DoorBird Video Door Station” and “BirdGuard” products and “mobile device” to refer to a smartphone or
tablet.

GENERAL INFORMATION

PRIVACY STATEMENT
Please observe any relevant country-specific statutory regulations concerning the use of surveillance components and security surveillance applicable at the installation site and within your third-party solution.

LOCATING DEVICES IN YOUR LAN
You can locate devices and obtain its IP address within your LAN using the Apple Bonjour Protocol, see e.g.

- DoorBird App → Settings → Administration → Search
- Online tool https://www.doorbird.com/checkonline
- Apple Bonjour command line tool “dns-sd”, e.g. “dns-sd.exe -B _http._tcp local”. Please see the Apple Bonjour documentation for further information on this topic

CONCURRENT CONNECTIONS AND RATE LIMITS
The device handles via this third-party API a maximum of 1 concurrent connection per second for API access.

The API has also a limit for all connection if wrong authentication is used. It will block the IP address or the whole user from the system for 1 minute after extensive use of wrong credentials. This can be seen by an HTTP response code 423.

Please keep in mind that the device is a Video Door Station, which handles in general - like all commercially relevant door stations - only one simultaneous audio/video call for live communication. You get a status code “503” (Busy) if another user already took the call. In that case you can notify the user with a message dialog on your GUI, e.g. “Line busy” and additionally preview one still image (LIVE IMAGE REQUEST).

INTEGRATION SCHEME

The following is a professional integration scheme and process flow. You should realize your integration like this.
The mobile devices (e.g. smartphone, tablet, smart-home panel) listen for UDP Broadcast messages within the LAN from the door station. If a valid UDP Broadcast message (valid: fits to the call-button number) is received from the door station by the mobile devices, then the mobile devices play a "ding-dong" notification sound and offer the End-user the possibility to take the call upon user-interaction by e.g. pressing a button on the mobile devices.

The mobile devices MUST NOT fetch the video and audio stream automatically upon receipt of a valid UDP Broadcast message, because a Video Door Station is a security device and not powered by an 8 GHz Octa-Core CPU with 32 GB RAM and thus is not capable to handle many concurrent streams.

A Video Door Station (traditional and nowadays) is designed to have an active video and audio call with only one mobile device simultaneously (1:1, not 1:n) ! The connection to the first mobile device is automatically interrupted if another (additional) mobile devices picks the call. Please do not try to lower the end-user experience by realizing simultaneous video and audio calls with several mobile devices. A Video Door Station does not support this.

When the End-user makes user-interaction (by e.g. pressing a button on the mobile device) then the mobile device tries to fetch a video and audio stream from the Video Door Station. If it fails because the Video Door Station is busy because another mobile is already fetching the video and audio stream then the End-user should be notified with e.g. "Line is busy, someone is already speaking" by the mobile device.

**URL SCHEME**

The URL-Scheme offers simple methods to interact with the DoorBird App on an iOS
or Android smartphone in order to perform tasks like opening a specific DoorBird device.

General information about how to use these URL Schemes are available within the programming guides of iOS and Android:

- [https://developer.android.com/training/basics/intents/sending.html](https://developer.android.com/training/basics/intents/sending.html)

**Calling URL**

The DoorBird Apps register the URL-Scheme doorbird:// which can be called from within a browser or an App.

To test the available services you can enter the URLs on iOS in Safari or on Android with the App "URL Scheme Sender"

**Available services**

1. Opens the DoorBird App
   
   `doorbird://`

2. Opens the DoorBird App, and shows the specific DoorBird device based on it's device ID in the live view. If the passed DoorBird device doesn't exists, only the App starts.
   
   `doorbird://live/%id`

3. Opens the DoorBird App and prevents fullscreen mode for landscape. (available since DoorBird app version 4.40)
   
   `doorbird://nofullscreen`

4. Opens the DoorBird App, prevents fullscreen mode for landscape and shows the specific DoorBird device based on it's device ID in the live view. If the passed DoorBird device doesn't exists, only the App starts. (available since DoorBird app version 4.40)
   
   `doorbird://nofullscreen/live/%id`

The device ID is six characters long and is contained in the DoorBird's usernames. The first six characters of a username represent the device ID.

**Example**

© 2023 by Bird Home Automation GmbH
HTTP INTERFACE DESCRIPTION

AUTHENTICATION

Please use Basic or Digest authentication as defined in RFC 2617 for each HTTP request. Use the same credentials as you are using to add a device to the DoorBird App.

Alternatively to authentication as defined in RFC 2617 you can use the plain-text HTTP parameters "http-user" and "http-password" to authenticate (more insecure because of plain-text, but some third-party home automation platforms support only HTTP parameters), e.g. "http://<device-ip>/bha-api/video.cgi?http-user=xxxxxx0001&http-password=xxxxxxx".

ENCRIPTION

Any communication between the device and the DoorBird Cloud servers is fully encrypted with latest encryption technology.

The HTTP interface of the device for third-party integrations is available unencrypted on TCP port 80 (HTTP protocol) and encrypted on TCP port 443 (HTTPS protocol) in the local area network (LAN). Certificate Authorities (CA) do not issue certificates for IP addresses, so the device has a self-signed certificate pre-installed for the HTTPS protocol in the LAN for third-party integrations.

Any request to the device can be realized either with http://<device-ip>/(..) or https://<device-ip>/(..) in the local area network for third-party integrations.

Exception: Video- and audio-streaming requests are currently not available with https in the local area network for third-party integrations. For video- and audio-streaming requests, you must obtain a temporary (!) Session ID first and then append the Session ID as parameter to any of your video- or audio-streaming request, in order to not transmit credentials in plaintext (video-streaming: http://<device-ip>/bha-api/video.cgi?sessionid=<session-id>, audio-streaming: http://<device-ip>/bha-api/audio-receive.cgi?sessionid=<session-id>).

Please make sure your client accepts the pre-installed self-signed certificate of the device (e.g. when using wget, append the self-signed certificate in the wget command by using command line parameter "--ca-certificate=..." and "-CAfile" or accept any certificate (insecure) by using the wget command line parameter "--no-check-certificate").

To create a temporary Session ID (valid for 10 minutes), use the following method.
Session ID is only valid for 10 minutes.

**Method:** GET

**Required permission:** valid user

**Syntax:**
```
http://<device-ip>/bha-api/getsession.cgi
```

**Example Request:**
```
http://<device-ip>/bha-api/getsession.cgi
```

**Return:**
```
HTTP/1.1 200 OK
Content-Type: application/json

{ "BHA": {
  "RETURNCODE": "1",
  "SESSIONID": "ISXA9dzpUfPUSlrNFufdOGDWRy9WadbtXtB45v9YFc3jMLf4yR50a37gak9f",
  "NOTIFICATION_ENCRYPTION_KEY": "7zuPtzNJZc72Fc2CI13cwz3ROqw2e0EtFLzv465JB2AyE2m0qrHhQqrsXh13T1"
}
```

To invalidate / destroy a Session ID, use the following method.

**Method:** GET

**Required permission:** valid user

**Syntax:**
```
http://<device-ip>/bha-api/getsession.cgi?<parameter>=<value>
```

<table>
<thead>
<tr>
<th>parameter</th>
<th>Values</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>invalidate</td>
<td>&lt;old_session_id&gt;</td>
<td>Hand over the session-id you want to invalidate through the request.</td>
</tr>
</tbody>
</table>

**Example Request:**
```
http://<device-ip>/bha-api/getsession.cgi?invalidate=
ISXA9dzpUfPUSlrNFufdOGDWRy9WadbtXtB45v9YFc3jMLf4yR50a37gak9f
```

**Return:**
```
HTTP/1.1 200 OK
Content-Type: application/json

{ "BHA": {
  "RETURNCODE": "1",
  "SESSIONID": ""
}
```
AVAILABLE PERMISSIONS

Users can have several permissions, which can be managed in the Administration area of the DoorBird App.

- **Watch always**: User can see live view and control the door/s at any time, even if there is no ring event
- **History**: User can access the history (e.g. Cloud-Recording)
- **Motion**: User can obtain motion notifications and access the motion history (e.g. Cloud-Recording)
- **API-Operator**: User can change settings through the Open API (www.doorbird.com/api) e.g. schedules and notifications and e.g. initiate an active SIP call (“makecall”) sing API command. You should only enable the “API-Operator” permission for users which are used on central Home Automation Servers, you should NOT enable “API-Operator” permission for users which are configured in Home Automation panels or Apps of end-users because then this user could change global settings for other users.

DEMONSTRATION

You may browse to http://<device-ip>/bha-api/view.html with a web browser to see a demonstration of the API in a standard webpage.
LIVE VIDEO REQUEST

Returns a multipart JPEG live video stream with the default resolution and compression as defined in the system configuration. When MJPG video is requested, the server returns a continuous flow of JPEG files. The content type is “multipart/x-mixed-replace” and each image ends with a boundary string <boundary>. The returned image and HTTP data is equal to the request for a live image request. An average of up to 8 fps can be provided using this third-party API, depending on the network speed and load factor of the device.

When the request is correct, but the requesting user has no permission to view the live stream at the moment, the request is answered with a return code 204. This usually happens for users without “watch-always” permission when there was no ring event in the past 5 minutes.

Please note, that the video connection can get interrupted at any time, when the official DoorBird App requests the stream. It has precedence over users of the LAN-API.

Method: GET
Required permission: valid user, “watch always” or ring event in the past 5 minutes for the requesting user

Syntax:

http://<device-ip>/bha-api/video.cgi

Example Request:

http://<device-ip>/bha-api/video.cgi

Return:

HTTP/1.0 200 OK\r\nContent-Type: multipart/x-mixed-replace;boundary=<boundary>\r\n\r
<br>\r\n<br>\r\n<br>\r\n<br>:\n
where the proposed <boundary> is “my-boundary” and the returned <image section> field is

Content-Type: image/jpeg\r\nContent-Length: <image size>\r\n\r
<JPEG image data>

Example: Requested Multipart JPEG image.

Copyright © 2023 by Bird Home Automation GmbH
LIVE IMAGE REQUEST

Returns a JPEG file with the default resolution and compression as defined in the system configuration. The content type is “image/jpeg”.

When the request is correct, but the requesting user has no permission to view the live image at the moment, the request is answered with return code 204. This usually happens for users without “watch-always” permission when there was no ring event in the past 5 minutes.

Method: GET

Required permission: valid user, “watch always” or ring event in the past 1 minute for the requesting user

Syntax:

http://<device-ip>/bha-api/image.cgi

Example Request:

http://<device-ip>/bha-api/image.cgi

Return:

HTTP/1.0 200 OK\r\nContent-Type: image/jpeg\r\nContent-Length: <image size>\r\n\r\n<JPEG image data>\r\n
OPEN DOOR

Energize the door opener/ alarm output relay of the device. Returns JSON.

We assume, that the API user watches the live image in order to open the door or trigger relays. So, when the request is correct, but the requesting user has no permission to view the live image at the moment, the request is answered with return code 204. This usually happens for users without “watch-always” permission when there was no ring event in the past 5 minutes.

**Method:** GET

**Required permission:** valid user, “watch always” or ring event in the past 5 minutes for the requesting user

**Syntax:**

```plaintext
http://<device-ip>/bha-api/open-door.cgi?<parameter>=<value>
```

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Values</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>r=string</td>
<td>1</td>
<td>2</td>
</tr>
</tbody>
</table>

**Example Requests:**

- `http://<device-ip>/bha-api/open-door.cgi`
- `http://<device-ip>/bha-api/open-door.cgi?r=1`
- `http://<device-ip>/bha-api/open-door.cgi?r=gggaaa@1`
LIGHT ON
Energize the light relay of the device. Returns JSON.

We assume, that the API user watches the live image in order to activate the light. So, when the request is correct, but the requesting user has no permission to view the live image at the moment, the request is answered with return code 204. This usually happens for users without “watch-always” permission when there was no ring event in the past 5 minutes.

Method: GET
Required permission: valid user, “watch always” or ring event in the past 5 minutes for the requesting user

Syntax:
http://<device-ip>/bha-api/light-on.cgi

Example Request:
http://<deviceip>/bha-api/light-on.cgi
HISTORY IMAGE REQUEST

Returns a JPEG history image with the default resolution and compression as defined in the system configuration. The history images are stored in the cloud.

If the authentication of the requesting user is ok, but he has no permission for this history, the request is answered with response code 204. This can be either the general “history” permission or the more specific “motion” permission.

Method: GET

Required permission: valid user, history permission, motion permission to access images from motion events

Syntax:

http://<device-ip>/bha-api/history.cgi?<parameter>=<value>

<table>
<thead>
<tr>
<th>&lt;parameter&gt;=&lt;value&gt;</th>
<th>Values</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>index=&lt;int&gt;</td>
<td>1..50</td>
<td>Index of the history images, where 1 is the latest history image</td>
</tr>
<tr>
<td>event=&lt;string&gt;</td>
<td>doorbell</td>
<td>motionsensor</td>
</tr>
</tbody>
</table>

Example Request:

http://<device-ip>/bha-api/history.cgi?index=1
http://<device-ip>/bha-api/history.cgi?index=22
http://<device-ip>/bha-api/history.cgi?
event=motionsensor&index=5

Return:

HTTP/1.0 200 OK\r\nContent-Type: image/jpeg\r\nContent-Length: <image size>\r\n\r
<JPEG image data>\r\n
MONITOR REQUEST

Returns the state of motionsensor and doorbell as a continuous multipart stream. Trigger information about rfid and keypad events coming soon. There are up to 8 concurrent Streams allowed. When all streams are busy returns HTTP code 509.

Method: GET
Required permission: valid user

Syntax:
http://<device-ip>/bha-api/monitor.cgi?ring=doorbell[,motionsensor]

<table>
<thead>
<tr>
<th>&lt;parameter&gt;=&lt;value&gt;</th>
<th>Values</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ring=&lt;string&gt;</td>
<td>doorbell</td>
<td>motionsensor</td>
</tr>
</tbody>
</table>

Note: rfid and keypad events coming soon

Example Request:
http://<device-ip>/bha-api/monitor.cgi?ring=doorbell,motionsensor

Example Return:
HTTP/1.1 200 OK\r\nContent-Type: multipart/x-mixed-replace; boundary=--ioboundary\r\n\r\n--ioboundary\r\nContent-Type: text/plain\r\n\r\ndoorbell:H\r\n\r\n--ioboundary\r\nContent-Type: text/plain\r\n\r\nmotionsensor:L\r\n\r
. . . \r\nContent-Type: text/plain\r\n\r\ndoorbell:L\r\n\r\n--ioboundary\r\nContent-Type: text/plain\r\n\r\nmotionsensor:L\r\n\r
HTTP status codes:
200 – OK
400 – Parameter missing or invalid
401 – Authentication required
LIVE AUDIO RECEIVE AND TRANSMIT – General information

- Audio can be received and transmitted via our HTTP interface or SIP interface.
- AEC/ANR: The transmitting user device (e.g. Home Automation tablet) MUST do the echo and noise reduction on its own (AEC, ANR). The DoorBird Video Door Station has a high-end hardware- and software-based echo canceller built-in, but due to natural physical circumstances you have to do echo cancellation on both sides, user device and door station. Our native Apps use high-end self-learning AEC and ANR algorithms. Our AEC and ANR algorithms in our native Apps are one of our core technologies and thus not available for any third party. You must develop an AEC / ANR on your own or use the native AEC / ANR of the operating system, see e.g. [https://developer.apple.com/library/content/documentation/MusicAudio/Conceptual/AudioUnitHostingGuide_iOS/UsingSpecificAudioUnits/UsingSpecificAudioUnits.html](https://developer.apple.com/library/content/documentation/MusicAudio/Conceptual/AudioUnitHostingGuide_iOS/UsingSpecificAudioUnits/UsingSpecificAudioUnits.html) and [https://developer.android.com/reference/android/media/audiofx/AcousticEchoCanceler.html](https://developer.android.com/reference/android/media/audiofx/AcousticEchoCanceler.html) for information about acoustic echo cancellation.
- Codec: When using this API audio MUST be G.711 μ-law (sampling rate 8000 Hz).
- Wireshark: When using Wireshark for packet inspection during development, audio transmission might not be shown in the “http” Filter, please chose a different filter.

LIVE AUDIO RECEIVE

Use this method to obtain real-time audio (G.711 μ-law) from the device.

When the request is correct, but the requesting user has no permission to view the live stream at the moment, the request is answered with a return code 204. This usually happens for users without “watch-always” permission when there was no ring event in the past 5 minutes.

Please note, that the audio connection can get interrupted at any time, when the official DoorBird App requests the stream. It has precedence over users of the LAN-API.

**Method:** GET

**Required permission:** valid user, “watch always” or ring event in the past 5 minutes for the requesting user

**Syntax:**

```
http://<device-ip>/bha-api/audio-receive.cgi
```

**Example Request:**

```
http://<device-ip>/bha-api/audio-receive.cgi
```

**Return:**
LIVE AUDIO TRANSMIT

Transmit audio (G.711 μ-law) from your mobile device (e.g. Home Automation tablet) to the device. Only one consumer can transmit audio (talk) at the same time. The second consumer will be rejected.

When the request is correct, but the requesting user has no permission to view the live stream at the moment, the request is answered with a return code 204. This usually happens for users without “watch-always” permission when there was no ring event in the past 5 minutes.

Please note, that the audio connection can get interrupted at any time, when the official DoorBird App requests the stream. It has precedence over users of the LAN-API.

Method: POST

Required permission: valid user, “watch always” or ring event in the past 5 minutes for the requesting user

Syntax:

http://<device-ip>/bha-api/audio-transmit.cgi

Example 1: Singlepart audio data transmit with G.711 μ-law (authorization omitted).

```plaintext
POST /bha-api/audio-transmit.cgi HTTP/1.0\r\nContent-Type: audio/basic\r\nContent-Length: 9999999\r\nConnection: Keep-Alive\r\nCache-Control: no-cache\r\n\r\n<AUDIO DATA>\n<AUDIO DATA>\n<AUDIO DATA> ...
```

Example 2: Usage with gstreamer.

```plaintext
gst-launch-1.0 alsasrc ! queue ! audioconvert ! audioresample ! "audio/x-raw,format=S16LE,rate=8000,channels=1" ! mulawenc ! "audio/x-mulaw,rate=8000,channels=1" ! curlhttpsink location=http://<device-ip>/bha-api/audio-transmit.cgi content-type="audio/basic" use-content-length=true user=xxxxxx0001 passwd=xxxxxxxxx
```
INFO REQUEST
Get some version information from the device in JSON format. Starting with firmware 000108, the relays configuration is included in the JSON output. It includes both, physical relays and paired DoorBird IP I/O Door Controllers.

Method: GET
Required permission: valid user

Syntax:
http://<device-ip>/bha-api/info.cgi

Example Request:
http://<device-ip>/bha-api/info.cgi

Example Return 1: DoorBird D101 with outdated firmware
{
  "BHA": {
    "RETURNCODE": "1",
    "VERSION": [{
      "FIRMWARE": "000096",
      "BUILD_NUMBER": "41865"
    }]
  }
}

Example Return 2: DoorBird D21x with paired DoorBird IP I/O DoorController
{
  "BHA": {
    "RETURNCODE": "1",
    "VERSION": [{
      "FIRMWARE": "000109",
      "BUILD_NUMBER": "15120529",
      "PRIMARY_MAC_ADDR": "1CCAE3700000",
      "RELAYS": ["1", "2", "gggaaa@1", "gggaaa@2"],
      "DEVICE-TYPE": "DoorBird D101"
    }]
  }
}
In order to react on events and execute actions on the DoorBird device, this API features favorites and schedules. Favorites are basic configuration units that can be used in schedules, e.g. an HTTP(S)-URL for notifications or SIP numbers. Favorites can be seen as address book entries. If you want to use an HTTP favorite for different events (e.g. smart home server), it is advised to save it's twice and handle the event type in the URL.

The user needs the “API operator” permission in order to access favorites.cgi.

Firmware 000110 or higher is required on your DoorBird/BirdGuard in order to access favorites and schedules.

Please note: being part of the LAN-2-LAN API, the firmware does not check the validity of certificates from HTTPS connections. It is not possible to get such certificates for IP addresses. The connection to your HTTPS favorites will still be encrypted.

### LIST FAVORITES

List all currently configured favorites as JSON.

**Method:** GET

**Required permission:** valid user, API operator permission

**Syntax:**

```
http://<device-ip>/bha-api/favorites.cgi
```

**Example Request:**

```
http://<device-ip>/bha-api/favorites.cgi
```

**Return:**

```
HTTP/1.0 200 OK

{  
    "sip":{
        "0":{
            "title":"Concierge",
            "value":"1234@sip.example.com"
        }
    },
    "http":{
        "1":{
            "title":"MyServer",
            "value":"http://10.0.0.1/foo/notify"
        },
        "5":{
            "title":"ServerX",
```
ADD OR CHANGE FAVORITE

Add a new favorite or change an existing favorite. If you add a new favorite, its id is available as response header value for key “favoriteid”.

Method: GET
Required permission: valid user, API operator permission

Syntax:

http://<device-ip>/bha-api/favorites.cgi?
action=save&amp;<parameter>=<value>

<table>
<thead>
<tr>
<th>parameter</th>
<th>values</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>action</td>
<td>save</td>
<td>Fixed parameter for saving favorites</td>
</tr>
<tr>
<td>type</td>
<td>sip</td>
<td>http</td>
</tr>
<tr>
<td>title</td>
<td>name / title</td>
<td>Name or short description of the favorite</td>
</tr>
<tr>
<td>value</td>
<td>URL / address</td>
<td>URL of the favorite, including protocol and user credentials, if necessary (see example above). This can be an HTTP(S) URL or an SIP target.</td>
</tr>
<tr>
<td>id</td>
<td>optional: id of the favorite</td>
<td>Specify the ID of the favorite to change; omit, if saving a new favorite</td>
</tr>
</tbody>
</table>

Example Requests: add HTTP favorite, change SIP favorite

http://<device-ip>/bha-api/favorites.cgi?
action=save&amp;type=http&amp;title=RingServ&amp;value=https://172.17.1.5/notify/ring

http://<device-ip>/bha-api/favorites.cgi?
action=save&amp;type=sip&amp;title=Johns %20Phone&amp;value=101@sip.domain.local&amp;id=2

Return:

HTTP/1.0 200 OK\r\n\r\n
HTTP status codes:
DELETE FAVORITE

Remove an favorite from the DoorBird device. If the favorite is actively used in a schedule configuration, the schedule entry will also be removed.

**Method:** GET

**Required permission:** valid user, API operator permission

**Syntax:**

```
http://<device-ip>/bha-api/favorites.cgi?
action=remove&<parameter>=<value>
```

<table>
<thead>
<tr>
<th>&lt;parameter&gt;=&lt;value&gt;</th>
<th>Values</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>action=&lt;string&gt;</td>
<td>remove</td>
<td>Fixed parameter for deleting favorites</td>
</tr>
<tr>
<td>type=&lt;string&gt;</td>
<td>sip</td>
<td>http</td>
</tr>
<tr>
<td>id=&lt;int&gt;</td>
<td>ID of the favorite</td>
<td>The ID of the favorite to delete</td>
</tr>
</tbody>
</table>

**Example Request:**

```
http://<device-ip>/bha-api/favorites.cgi?
action=remove&type=sip&id=2
```

**Return:**

```
HTTP/1.0 200 OK\r\n\r\n```

**HTTP status codes:**

- 200 – OK
- 400 – Parameter missing or invalid
- 401 – Authentication required
- 500 – Internal error
SCHEDULE MANAGEMENT

With schedule entries one can configure the actions, that a DoorBird device executes at certain events. It is possible to setup the input event (e.g. ring, motion), the output event (e.g. HTTP notification, SIP call) and the time window, where this rule is active.

The API handles 3 different schedule types: “once” (one time event), “from-to” and “weekdays” (recurring time ranges on a weekly base).

“Once” is rather self explanatory, the schedule gets invalid after a single use. You can enable it again, so you don't have to configure it again.

“From-to” can be used for all distinct – not recurring - time ranges. Time unit are seconds since 1.1.1970 and the timezone is UTC.

Seconds are used as unit for time information in “weekdays” setup. Starting point is Sunday 0:00 o'clock. Maximum time value for weekly events is 604799 seconds (7 days * 24 hours * 60 minutes * 60 seconds – 1 second). The 24 hours of a day are divided into 48 time slices of 30 minutes (1800 seconds) length each. It is important, that all starting times are multiples of 1800 seconds. Start and end second of time intervals are included, thats why one must subtract a second to calculate the ending time of an interval. Use “from” 0 “to” 604799 to configure a schedule entry which is valid for the whole week (always).

There can be only one schedule entry for each output type, time slot and event slot. Example: just one HTTP trigger for ring events at 8 o'clock. But an SIP call at the same time is possible. If more events are needed, one needs to implement event-multiplexing. Exceptions are relays, where more can be triggered for the same event.

The user needs the “API operator” permission in order to access schedules.cgi.

Firmware 000110 or higher is required on your DoorBird/BirdGuard in order to access favorites and schedules.

Hint: entries from the old “notification.cgi” configuration get migrated to schedule entries.

LIST SCHEDULES

List all currently configured schedules settings as JSON.

Method: GET

Required permission: valid user, API operator permission

Syntax:

http://<device-ip>/bha-api/schedule.cgi

Attributes of the schedule JSON object:

<table>
<thead>
<tr>
<th>JSON attributes</th>
<th>Values</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>input</td>
<td>doorbell</td>
<td>motion</td>
</tr>
<tr>
<td>param</td>
<td>Parameter value for the input, e.g. doorbell number, RFID transponder id, fingerprint id</td>
<td></td>
</tr>
<tr>
<td>--------------</td>
<td>--------------------------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>output</td>
<td>JSON array</td>
<td></td>
</tr>
<tr>
<td></td>
<td>JSON array of output action configurations</td>
<td></td>
</tr>
</tbody>
</table>

Attributes of the JSON object for an “output” configuration:

<table>
<thead>
<tr>
<th>JSON attributes</th>
<th>Values</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>event</td>
<td>notify</td>
<td>sip</td>
</tr>
<tr>
<td></td>
<td>Type “notify” is used to trigger push notifications and cloud recordings</td>
<td></td>
</tr>
<tr>
<td>param</td>
<td>&lt;&gt;</td>
<td>&lt;sip-favorite-id&gt;</td>
</tr>
<tr>
<td>schedule</td>
<td>once</td>
<td>from-to</td>
</tr>
</tbody>
</table>

Example Request:

```
http://<device-ip>/bha-api/schedule.cgi
```

Return:

```
HTTP/1.0 200 OK

{
    "input":"doorbell",
    "param":"1",
    "output":[
        {
            "event":"http",
            "param":"1",
            "schedule":{
                "weekdays":[
                    {
                        "from":"122400",
                        "to":"151199"
                    }
                ]
            }
        }
    ],

    "input":"motion",
    "param":",
    "output":[
        {
            "event":"relay",
            "param":"2",
            "schedule":{
                "from-to":[
                    {
                        "from":"122400",
                        "to":"151199"
                    }
                ]
            }
        }
    ]
}
```
ADD OR UPDATE SCHEDULE ENTRY

Add or update an schedule setting by sending the configuration as JSON object. One request for each input type (e.g. "motion" or "doorbell #6789") is required.

Method: POST

Required permission: valid user, API operator permission

Syntax:

http://<device-ip>/bha-api/schedule.cgi

Example Request:

http://<device-ip>/bha-api/schedule.cgi

Example JSON content for doorbell events:

```json
{
  "input": "doorbell",
  "param": "1",
  "output": [{
    "event": "notify",  <!-- send notifications (push, history) -->
    "param": "",
    "enabled": "1",
    "schedule": {
      "weekdays": [{
        "to": "82799",
        "from": "82800"
      }]
    }
  }, {
    "event": "http",
    "param": "3",  <!-- trigger http favorite #3 -->
    "enabled": "1",
    "schedule": {
      "weekdays": [{
        "to": "82799",
        "from": "82800"
      }]
    }
  }]
}
```
Example JSON content for motion events:

```
{
    "input": "motion",
    "param": "",
    "output": [{
        "event": "notify",
        "param": "",
        "enabled": "1",
        "schedule": {
            "weekdays": [{  <!--always trigger -->
                "to": "82799",
                "From": "82800"
            }
        }
    }, {
        "event": "relay",
        "param": "1",
        "enabled": "1",
        "schedule": {
            "weekdays": [{  <!--always trigger -->
                "to": "82799",
                "From": "82800"
            }
        }
    }
    ]
}
```

Return:

```
HTTP/1.0 200 OK

HTTP status codes:
200 – OK
400 – Any of: invalid JSON format; Content-Length header missing; Content-Length header does not match real content size; content size too large
401 – Authentication required
500 – internal error
507 – Size limit exceeded (too many or to large schedules

DELETE SCHEDULE ENTRY

Delete an schedule entry.

Method: GET

Required permission: valid user, API operator permission

Syntax:
```
http://<device-ip>/bha-api/schedule.cgi?action=remove&<parameter>=<value>
```
<table>
<thead>
<tr>
<th><code>&lt;parameter&gt;</code>=&lt;<code>value&gt;</code></th>
<th>Values</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>action</code>=&lt;string&gt;</td>
<td>remove</td>
<td>Fixed parameter for deleting a schedule entry</td>
</tr>
<tr>
<td><code>input</code>=&lt;string&gt;doorbell</td>
<td>motion</td>
<td>rfid</td>
</tr>
<tr>
<td><code>param</code>=&lt;string&gt;</td>
<td><code>&lt;doorbell-number&gt;</code></td>
<td>&lt;&gt;</td>
</tr>
</tbody>
</table>

**Example Request:**

```
http://<device-ip>/bha-api/schedule.cgi?
action=remove&input=motion&param=xxx
```

**Return:**

```
HTTP/1.0 200 OK
\r\n\r\n```

**HTTP status codes:**
- 200 – OK
- 401 – Authentication required
- 500 – internal error
RESTART

Restarts the device. There will be no diagnostic sound (e.g. “successfully connected to internet”) after this restart.

**Method:** GET  
**Required permission:** valid user, API operator permission

**Syntax:**
http://<device-ip>/bha-api/restart.cgi

**Example Request:**
http://<device-ip>/bha-api/restart.cgi

**Return:**
HTTP/1.0 200 OK

HTTP status codes:
- 200 – OK
- 401 – Authentication required
- 503 – device is busy (e.g. currently installing an firmware update)

EVENT MONITORING (UDP BROADCASTS)

Since November 2023 there is a new “v.2” handling for encrypting/decrypting the events. The version 1 has been deprecated and will be removed in the future. It can also be disabled by the user in the administration area of the app. Integrations which are using the old version should update it as soon as possible. The new version simplifies the decryption by no longer using the password stretching algorithm “Argon2i” but instead using a longer independent password.

**Event Monitoring v2**

After an event occurred, the DoorBird sends multiple identical UDP-Broadcasts on the ports 6524 and 35344 for every user and every connected device. You can split each package in two sections. The first one contains information about the package, the second part about encryption with payload. Please notice, we are also sending keep alive broadcasts every 7 seconds on this two ports, these packets are not relevant for the decryption of event broadcasts, you can skip them.

To decode these UDP packets in version 2, the algorithm ChaCha20 must be supported. It is for example included in the well-known Sodium crypto library (libsodium).

**First Part:**

<table>
<thead>
<tr>
<th>Fieldname</th>
<th>Length in Bytes</th>
<th>Datatype</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>IDENT</td>
<td>3</td>
<td>Byte</td>
<td>To identify this kind of package the first three</td>
</tr>
</tbody>
</table>
bytes contains a identifier.
IDENT[0] = 0xDE
IDENT[1] = 0xAD
IDENT[2] = 0xBE

The VERSION Flag allows us to distinguish between different encryptions and package types. Right now we support the following types:
0x01 – deprecated - ChaCha20-Poly1305 with Argon2i
0x02 – ChaCha20-Poly1305

Second Part for a package in VERSION 0x02:

<table>
<thead>
<tr>
<th>Fieldname</th>
<th>Length in Bytes</th>
<th>Datatype</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>NONCE</td>
<td>8</td>
<td>Byte</td>
<td>Used for encryption with ChaCha20-Poly1305</td>
</tr>
<tr>
<td>CIPHERTEXT</td>
<td>34*</td>
<td>Byte</td>
<td>With ChaCha20-Poly1305 encrypted text which contains informations about the Event.</td>
</tr>
</tbody>
</table>

*The encrypted CIPHERTEXT contains 16 byte with random values, these bytes are no longer present after decryption.

The CIPHERTEXT after decryption:

<table>
<thead>
<tr>
<th>Fieldname</th>
<th>Length in Bytes</th>
<th>Datatype</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>INTERCOM_ID</td>
<td>6</td>
<td>String</td>
<td>The first 6 digits of the user name. You can ignore all packets, where this doesn't match your DoorBird user).</td>
</tr>
<tr>
<td>EVENT</td>
<td>8</td>
<td>String</td>
<td>Contains the doorbell or „motion“ to detect which event was triggered. Padded with spaces.</td>
</tr>
<tr>
<td>TIMESTAMP</td>
<td>4</td>
<td>Long</td>
<td>A Unix timestamp, long.</td>
</tr>
</tbody>
</table>

Used Algorithms:

<table>
<thead>
<tr>
<th>Version</th>
<th>Name</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x02</td>
<td>ChaCha20-Poly1305</td>
<td>Authenticated Encryption</td>
</tr>
</tbody>
</table>
Step by step example:

**Step 1:** only needed one time (or after the password of the user changed):

Request the key which is used for decrypting the notifications. It can be obtained by calling the `getsession.cgi` request and using the DoorBirdUser and password for authentication.

**Syntax:**

```
http://<device-ip>/bha-api/getsession.cgi
```

**Return:**

```
HTTP/1.1 200 OK
Content-Type: application/json

{ "BHA": {
    "RETURNCODE": "1",
    "SESSIONID": "ISX9dzpUPUSlRFufd0gGDWry9WadbtXTbB45v9YFc3jMLf4yR50a37gak9F",
    "NOTIFICATION_ENCRYPTION_KEY": "BHYGHyRKtGzBjku2tj2XUk1dXYQ3VqmfbKoCtxXJ604lgSzpgIwZ6onrSh"
} }
```

The value for the key "NOTIFICATION_ENCRYPTION_KEY" needs to be stored and can be used from there on to decrypt the UDP notification packets. The key is valid until the password of the user changes, then it would need to be requested again via the same call. **So this request needs to be done only once and should not be done for each received packet.**

**Notice:** The length of "NOTIFICATION_ENCRYPTION_KEY" is 32-64 bytes. This length was selected for compatibility for future changes. For ChaCha20 only the first 32 Bytes of that key will be used. All additional bytes will be ignored by ChaCha20 algorithm itself.

**Step 2:** You capture the following packet via UDP:

```
0xDE 0xAD 0xBE 0x02 0x96 0x13 0x80 0xD4 0x62 0x2E 0xBE 0xE7 0x2A 0x9F 0xC3 0xFF 0x0B 0xEF 0x62 0x64 0xF2 0xAE 0x91 0x94 0x92 0x14 0x8B 0xBD 0x30 0xEB 0x05 0xBD 0xCE 0x36 0x7C 0x33 0xD4 0x29 0x3F 0xAF 0xE0 0x60 0x45 0x9E 0x65 0x10
```

**Step 3:** Split it up:

<table>
<thead>
<tr>
<th>Field</th>
<th>Content</th>
</tr>
</thead>
<tbody>
<tr>
<td>IDENT</td>
<td>0xDE 0xAD 0xBE</td>
</tr>
<tr>
<td>VERSION</td>
<td>0x02</td>
</tr>
<tr>
<td>NONCE</td>
<td>0x96 0x13 0x80 0xD4 0x62 0x2E 0xBE 0xE7</td>
</tr>
<tr>
<td>CIPHERTEXT</td>
<td>0x2A 0x9F 0xC3 0xFF 0x9B 0xEF 0x62 0x64 0xF2 0xAE 0x91 0x94 0x92 0x14 0x8B 0xBD 0x30 0xEB 0x05 0xBD 0xCE 0x36 0x7C 0x33 0xD4 0x29 0x3F 0xAF 0xE0 0x60 0x45 0x9E 0x65 0x10</td>
</tr>
</tbody>
</table>

**Step 4:** Decrypt CIPHERTEXT with ChaCha20-Poly1305, use the password you
obtained one time via getsession.cgi, output should be this:
0x67 0x68 0x69 0x6B 0x7A 0x69 0x31 0x20 0x20 0x20 0x20 0x20 0x20 0x20 0x65 0x4D 0x13 0x51

**Step 5: Split the output up:**

<table>
<thead>
<tr>
<th>Field</th>
<th>Byte Value</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>INTERCOM_ID</td>
<td>0x67 0x68 0x69 0x6B 0x7A 0x69</td>
<td>ghikzi</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Starting 6 chars from the user name. Skip the packet, if this doesn't match your DoorBird user).</td>
</tr>
<tr>
<td>EVENT</td>
<td>0x31 0x20 0x20 0x20 0x20 0x20</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(doorbell number from an D1101 in this example, padded with spaces)</td>
</tr>
<tr>
<td>TIMESTAMP</td>
<td>0x65 0x4D 0x13 0x51</td>
<td>1699550033 or readable for humans: Thursday, 09. November 2023 17:13:53 UTC</td>
</tr>
</tbody>
</table>

**EXAMPLE SOURCE CODE**

The following c-code shows the decoding part using libsodium method calls. It uses a few internal structs, methods and macros, but these are self explanatory.

**Decryption:**

```c
NotifyBroadcastCiphertext decryptBroadcastNotification(const NotifyBroadcast* notification, const Password* password) {
    NotifyBroadcastCiphertext decrypted = {{0},{0},0};
    if(crypto_aead_chacha20poly1305_decrypt((unsigned char*)&decrypted, NULL, NULL, notification->ciphertext, sizeof(notification->ciphertext), NULL, 0, notification->nonce, password->key)!=0){
        LOGGING("crypto_aead_chacha20poly1305_decrypt() failed");
    }
    return decrypted;
}
```
RTSP INTERFACE DESCRIPTION

LIVE VIDEO REQUEST

Returns a MPEG4 H.264 live video stream with the default resolution and compression as defined in the system configuration. Uses RTSP on 554 and the RTSP-over-HTTP protocol on port 8557 of DoorBird and BirdGuard devices. An average of up to 12 fps can be provided using this third-party API, depending on the network speed and load factor of the device. Requires standard RTSP authentication (no parameter authentication supported as we support for HTTP).

When the request is correct, but the requesting user has no permission to view the live stream at the moment, the request is answered with a return code 204. This usually happens for users without “watch-always” permission when there was no ring event in the past 5 minutes.

Please note, that the RTSP connection can get interrupted at any time, when the official DoorBird App requests the stream. It has precedence over users of the LAN-API.

Method: GET

Required permission: valid user, “watch always” or ring event in the past 5 minutes for the requesting user

Syntax:

- rtsp://<device-ip>:<device-rtsp-port>/mpeg/media.amp
- rtsp://<device-ip>:<device-rtsp-port>/mpeg/720p/media.amp *(*)
- rtsp://<device-ip>:<device-rtsp-port>/mpeg/1080p/media.amp (**)

Example Request:

- rtsp://<device-ip>:8557/mpeg/media.amp
- rtsp://<device-ip>/mpeg/media.amp
- rtsp://<device-ip>/mpeg/1080p/media.amp

(*) supported by DoorBird Video Door Station D10x/D21x from Firmware-Version 129
(**) supported by DoorBird Video Door Station D11x only
Session Initiated Protocol (SIP)

SIP
This method is to configure and use the SIP service which is built into the device. The SIP service is in early stage of development, please note the following:

- The SIP registration will be stored permanently in the device. Every hardware restart will force the SIP service to restart / register again.
- You must use a username/password combination with the “API operator” permission (DoorBird App → Administration → User → Edit → Permissions). This makes sure that no other user (a non SIP relevant user) can modify the SIP settings.
- Each SIP call terminates 180 seconds after it was initiated, for security reason (auto-hangup).
- The SIP service will not initiate the call automatically when someone pushes the doorbell button, you have to listen to the notifications of the device (see document: LAN API) and initiate the call with the “makecall” action of “sip.cgi”
- The device supports only one simultaneous SIP call
- If DTMF Support is enabled the telephone (recipient of the SIP call) can trigger the door open relay by pressing the pincode for relay.
- Making / establishing the call takes sometimes a few seconds, the SIP handshake needs some synchronization, this is comparable to a standard SIP softphone or hardphone.
- We do a lot of debugging. If something is not working, please wait 5-7 seconds. If this doesn't solve the issue, please restart the device and let us know what environment you are using (network, SIP Proxy, calling device etc.)
- Don’t penetrate the device with many concurrent SIP requests, please wait min 3 seconds between each SIP request
- Please note that you need an Acoustic Echo Canceller (AEC) on the client-side as well as an Acoustic Noise Canceller (ANC). This ensures high quality audio calls.

Permissions

<table>
<thead>
<tr>
<th>PERMISSIONS</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Watch always</td>
<td></td>
</tr>
<tr>
<td>History</td>
<td></td>
</tr>
<tr>
<td>Motion</td>
<td></td>
</tr>
<tr>
<td>API-Operator</td>
<td></td>
</tr>
</tbody>
</table>

Copyright © 2023 by Bird Home Automation GmbH
• The device will close any ongoing SIP connection, if there is an listen/talk request from the official DoorBird App.
• From device version 000099 on, peer-2-peer (P2P) SIP calls are supported. After enabling the SIP functionality with “enable=1”, the device is ready to receive SIP calls on port 5060. To issue outgoing calls from the device, call “action=makecall” or configure automatic calls on ring events using parameter “autocall_doorbell_url”.

**SIP Registration**

Register to a SIP Proxy. This is not necessary, if you are using peer-2-peer calls.

**Method:** GET  
**Required permission:** valid user, “API operator” permission

**Syntax:**

```plaintext
http://<device-ip>/bha-api/sip.cgi?action=registration&user=<user>&password=<password>&url=<url>
```

<table>
<thead>
<tr>
<th>&lt;parameter&gt;</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>user=&lt;String&gt;</td>
<td>Authentication user for the SIP Proxy</td>
</tr>
<tr>
<td>password=&lt;String&gt;</td>
<td>Authentication password for the SIP Proxy</td>
</tr>
<tr>
<td>url=&lt;String&gt;</td>
<td>IP/Hostname of the SIP Proxy</td>
</tr>
</tbody>
</table>

**Example “registration” Request:**

```plaintext
http://<deviceip>/bha-api/sip.cgi?action=registration&user=foo&password=bar&url=192.168.123.22
```

**Returns:**

- 200 if everything is okay.
- 401 on authentication failure (user name/password from Basic Authentication wrong or no “API operator” permission)

**SIP Make Call**

Manually initiate the SIP call. This can either be an peer-2-peer call or a “normal” SIP call using the configured PBX / SIP proxy.

**Method:** GET  
**Required permission:** valid user, “API operator” permission

**Syntax:**

```plaintext
http://<device-ip>/bha-api/sip.cgi?action=makecall&url=<url>
```

<table>
<thead>
<tr>
<th>&lt;parameter&gt;=&lt;value&gt;</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>url=&lt;String&gt;</td>
<td>SIP URL to call</td>
</tr>
</tbody>
</table>

**Example “makecall” Request:**

```plaintext
http://<deviceip>/bha-api/sip.cgi?action=makecall&url=sip:108@192.168.123.22
```
SIP Hangup

Hangup the SIP call. If there is currently no ongoing call, this CGI returns “200 OK” too.

Method: GET

Syntax:

```
http://<device-ip>/bha-api/sip.cgi?action=hangup
```

Returns:

- 200 if everything is ok
- 400 if there is something wrong, e.g. parameter missing
- 401 on authentication failure (user name/password from Basic Authentication wrong or no “API operator” permission)
- 503 if the call does not work, e.g. line busy

SIP Settings (CGI)

Configure several SIP related settings.

Important note on the autocall_doorbell_url setting: this feature is deprecated here and will be removed in the future. It got replaced by schedule.cgi (together with favorites.cgi), because of the possibility to use schedules (e.g. call a different number at night). Currently all changes to autocall_doorbell_url get migrated into appropriate favorite and schedule entries in order to stay compatible with existing 3rd party drivers.

Method: GET

Required permission: valid user, “API operator” permission

Syntax:

```
http://<device-ip>/bha-api/sip.cgi?action=settings&amp;<parameter>=&lt;value&gt;
```

<table>
<thead>
<tr>
<th>&lt;parameter&gt;</th>
<th>Values</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>enable</td>
<td>0..1</td>
<td>Enable or disable SIP registration after reboot of the device, default: 0</td>
</tr>
<tr>
<td>mic_volume</td>
<td>1..100</td>
<td>Set the microphone volume, default: 33</td>
</tr>
<tr>
<td>spk_volume</td>
<td>1..100</td>
<td>Set the speaker volume, default: 70</td>
</tr>
<tr>
<td>dtmf</td>
<td>0..1</td>
<td>Enable or disable DTMF support, default: 0</td>
</tr>
<tr>
<td>autocall_doorbell_url</td>
<td>URL or &quot;none&quot;</td>
<td>DEPRECATED: use schedule.cgi SIP URL to automatically call upon doorbell event. By the second doorbell event hangs the previous</td>
</tr>
<tr>
<td>Parameter</td>
<td>Value Range</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------</td>
<td>-------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>relay1_passcode</td>
<td>0..99999999</td>
<td>Pincode for triggering the door open relay. Set to “none” to disable this automatic call. Default: “none”</td>
</tr>
<tr>
<td>incoming_call_enable</td>
<td>0..1</td>
<td>Enable or disable incoming calls. Default: 0</td>
</tr>
<tr>
<td>incoming_call_user</td>
<td></td>
<td>Allowed SIP user which will be authenticated for DoorBird. E.g. “sip:10.0.0.1:5060” or “sip:user@10.0.0.2:5060”.</td>
</tr>
<tr>
<td>anc</td>
<td>0..1</td>
<td>Enable or disable acoustic noise cancellation. Default: 1</td>
</tr>
<tr>
<td>ring_time_limit</td>
<td>10..300</td>
<td>Set the maximum ringing time in seconds. Default: 300</td>
</tr>
<tr>
<td>call_time_limit</td>
<td>30..300</td>
<td>Set maximum call duration in seconds. Default: 300</td>
</tr>
</tbody>
</table>

**Example “settings” Request:**

```
http://<device-ip>/bha-api/sip.cgi?action=settings&autocall_doorbell_url=sip:108@192.168.123.22
```

**Returns:**
- 200 if everything is okay
- 401 on authentication failure (user name/password from Basic Authentication wrong or no “API operator” permission)

**SIP Status**

You can query the current SIP status by calling the following URL.

**Method:** GET  
**Required permission:** valid user, “API operator” permission

**Syntax:**

```
http://<device-ip>/bha-api/sip.cgi?action=status
```

**Returns:**
- JSON, where “LASTERRORCODE”: “200” means that the SIP client is successfully registered. The attribute LASTERRORCODE contains the most recent SIP status code and “LASTERRORTEXT” the most recent SIP error text.
- 200 if everything is okay
- 401 on authentication failure (user name/password from Basic Authentication wrong or no “API operator” permission)

**SIP Settings Reset**

Resets all SIP related settings except the license, e.g. SIP proxy settings (action=registration) and SIP settings (action=settings). Hangs up any ongoing call.

**Method:** GET  
**Required permission:** valid user, “API operator” permission

**Syntax:**

```
http://<device-ip>/bha-api/sip.cgi?action=reset
```

**Returns:**
• JSON, where ‘LASTERRORCODE’: “200” means that the SIP client is successfully registered. The LASTERRORCODE returns the most recent SIP status code and “LASTERRORTEXT” the most recent SIP error text.
• 200 if everything is okay
• 401 on authentication failure (user name/password from Basic Authentication wrong or no “API operator” permission)

SIP Settings (DoorBird App)
The DoorBird App supports setting SIP properties directly at the Administration section. Log into your DoorBird with the Administration credentials (e.g. QR code), scroll down to “Expert Settings” where you can find the “SIP Settings”.

Note: if your SIP proxy is configured to use a different port than the default port for SIP communication (5060), add the port number to the address of your proxy in “SIP Proxy”, separated with a colon, e.g. 10.11.12.13:9999.

Example SIP settings inside the DoorBird App:

<table>
<thead>
<tr>
<th>Back</th>
<th>SIP Settings</th>
<th>Save</th>
</tr>
</thead>
<tbody>
<tr>
<td>SIP SETTINGS</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SIP activated</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SIP Proxy</td>
<td>10.11.12.13:9999</td>
<td></td>
</tr>
<tr>
<td>SIP User</td>
<td>12345</td>
<td></td>
</tr>
<tr>
<td>SIP Password</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DTMF</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Light PIN</td>
<td>321</td>
<td></td>
</tr>
<tr>
<td>Relay 1 PIN</td>
<td>562</td>
<td></td>
</tr>
<tr>
<td>Relay 2 PIN</td>
<td>996</td>
<td></td>
</tr>
<tr>
<td>Allow incoming calls</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Allowed SIP Users</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>Button action during call</td>
<td>No action</td>
<td></td>
</tr>
</tbody>
</table>