How To.

AXIS Audio Manager Pro API

Best practices and examples

July 11, 2024
Rev. 1.1

Contents

Please note that Axis doesn't take any responsibility for how this configuration may affect your system. If the modification fails or if you get other unexpected results, you may have to restore the settings to default.

Introduction

This document describes best practices, examples, and integration notes on how to integrate AXIS Audio Manager Pro through an API (Application Programming Interface). For the full API specification, go to Axis VAPIX library.

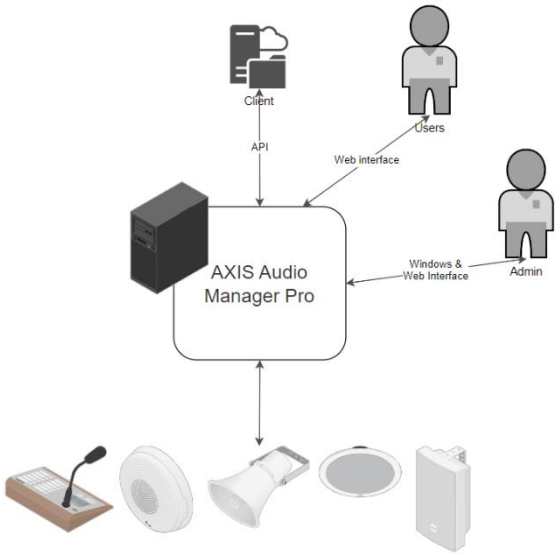
AXIS Audio Manager Pro is an audio management solution to configure and control a site of up to several thousands of speakers. It enables calibration of volumes, creation of scheduled playback, paging (live callouts), to name a few features.
AXIS Audio Manager Pro API offers the possibility to interact programmatically with an existing audio site. With the API you can for example ask the system to play a clip at a certain area.
Users of the API would typically be:

- Mobile applications
- Action engines e.g., Node-RED
- Custom scripts/programs to automate tasks
- Systems such as alarm systems or video management systems (VMS)

Note! The API is an operational API intended to give the client a way to interact with or control an already configured and running audio site. The scope for the API is not to offer all the functionality that is available in AXIS Audio Manager Pro.

Prerequisites

AXIS Audio Manager Pro 4.3 or later for API version 1.0
AXIS Audio Manager Pro 4.6 or later for API version 1.1



Versions and versioning

Version 1.0 of the API was released with AXIS Audio Manager Pro 4.3 in Q2 2023. Version 1.0 includes endpoints to achieve two main use cases:

- Play an audio file (clip) in the whole audio site or a certain area (zone).
  Example: "Play the closing message (closing.mp3) to the store area"
- Enable/disable parts of the audio site.
  Example: "Disable the physical zone called building B to stop all planned content and pages to that area temporarily"

Version 1.1 of the API was released with AXIS Audio Manager Pro 4.6 in Q2 2024. Version 1.1 includes endpoints to achieve these main use cases:

- Change the volume for all or specific content types in the whole audio site or a certain area (zone).
- Check the health status (online, offline, playing, error) of the devices in the whole audio site or a certain area.
- Send live audio (page) from another system into AXIS Audio Manager Pro using SIP or RTP.

The API will be expanded over time to offer more use cases.
Any major or breaking changes will result in a new major version with a new end point URL (e.g., /api/**v2.0**/audioSessions/). This way the old versions will still work for a period before being deprecated.

For versions 4.6 and later, both v1.0 and v1.1 are available.

How to access the API

To access the API, it first needs to be enabled in AXIS Audio Manager Pro.
Go to System Settings > API Access. Enable the API and set a username and password.

Note! The API is only accessible through https (not http). Port 443 is default for communication. If you use another port, use https://<server ip>:<port>/api... in the URLs.

Note that both URLs and json data is case sensitive!

Examples on how to implement the API

Below follows a set of examples on how to use the APIs for AXIS Audio Manager Pro.

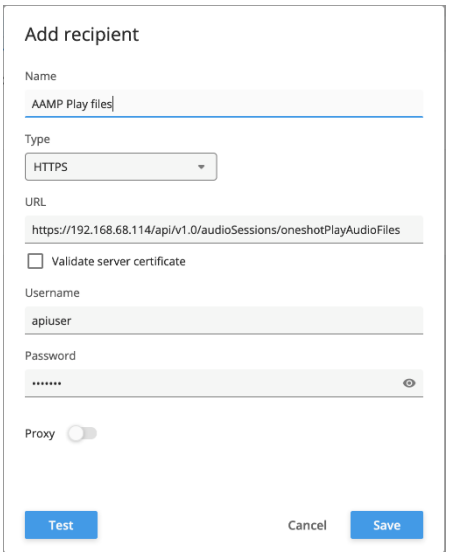
Example A: Play an audio clip with action engine

- In this example we will go through how to play an audio clip once the sound level reaches a certain level.
- Play disturb.mp3 in Building A if the sound level reaches above a set level during nighttime.
- The action engine that is available in most Axis devices is used in this example.

1. Prepare the audio clip (disturb.mp3) and upload it as an announcement in AXIS Audio Manager Pro.
2. Prepare the command to play the audio clip.
    a. Make a request to get the id of the audio clip by entering this URL in the web browser: https://<server ip>/api/v1.0/audioFiles. Enter the API username and password. A list of all the audio files on the server will be returned in the browser window. Find the file and note its id.

    b. Make a similar request to get the id of all the targets available. Enter a URL https://<server ip>/api/v1.0/targets. Extract the target id(s) you want to use and note them down.

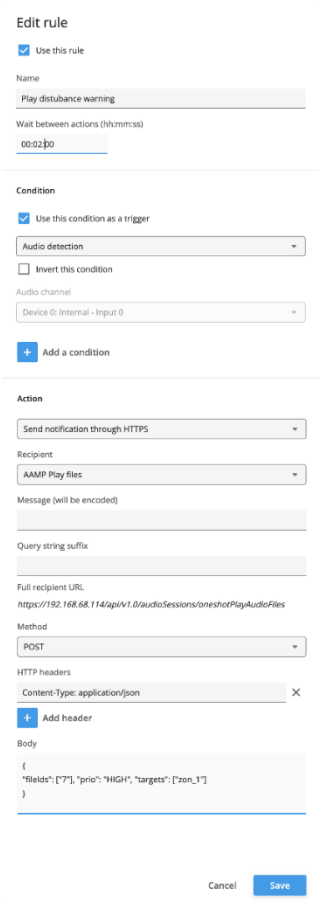
3. On the device that should measure the sound level and send the command to play the audio clip:
    a. Create a recipient in System>Events>Recipients.
    Create a recipient for every API endpoint you plan to use, by entering the URL: https://<server ip>/api/v1.0/audioSessions/oneshotPlayAudioFiles.
    Uncheck “Validate server certificate” unless you have set up a trusted chain of certificate. Note that the test button might not create a successful result here as the endpoint requires a POST message with additional data.



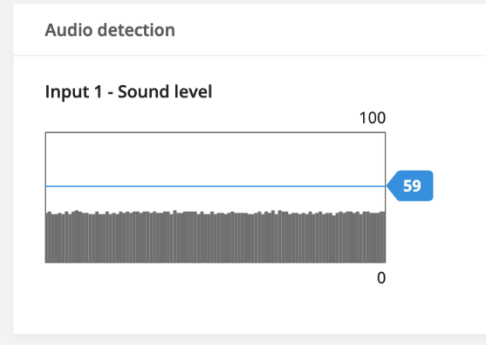
    b. Create a rule in System>Events>Rules
    Choose the action **Audio detection.** Remember to keep the mic input enabled both physically on the device and under Audio>Device settings.
    Choose the action **Send notification through HTTPS** and use the recipient from above. Choose the POST method and set at least one header:
    Content-Type: application/json
    Craft the body as a json according to the API specification. Example:

```
{
        "fileIds": ["1", "2"],
         "prio": "HIGH",
        "targets": ["zon_1", "sit_1"]
}
```

Note! Don't forget to enable **Use this rule** & **Use this condition as a trigger.**
Set the **Wait between actions** to the minimal time wanted between triggering this action.



4. Set the sound level for when the system should react in System>Detectors>Audio detection.



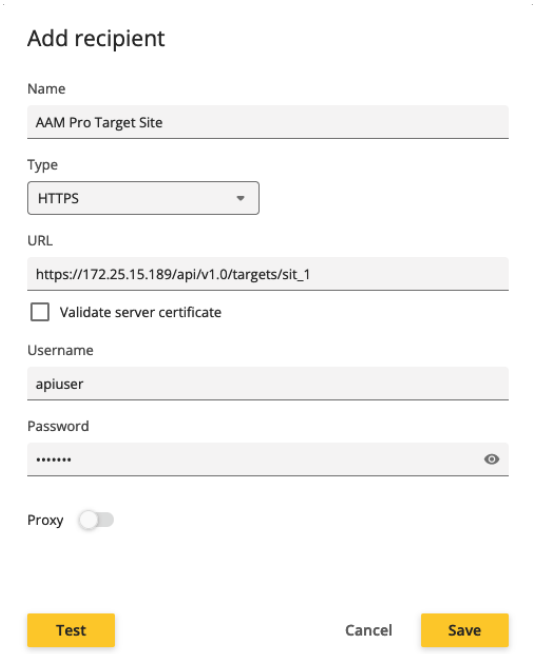
5. Done! When the sound level reaches above the level (59 in this case), the action will be triggered.

Example B: Disable (mute) a site based on digital input

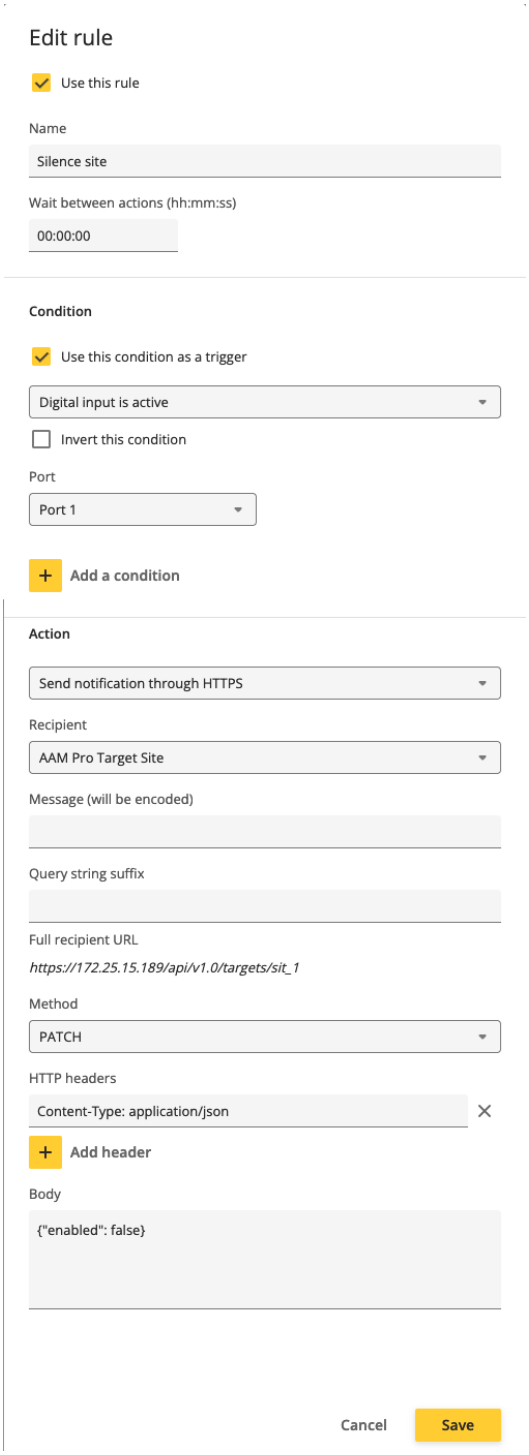
- In this example we will go through how to mute a site when a digital input is active.
- This example can be used to mute a site if a fire alarm is active.
- The action engine that is available in most Axis devices is used in this example.

1. Find the id of the target you want to silence; it can be the whole site or a specific zone. Make a request to get all the available targets by entering this URL in a browser: https://<server ip>/api/v1.0/targets. Extract the target id(s) you want to use and note them down.

2. Create a recipient in System>Events>Recipients on the speaker that should be muted. Add the id(s) of the targeted target in the URL e.g., https://<server ip>/api/v1.0/targets/sit_1. Don't forget to uncheck "Validate server certificate" if you have not setup a valid certificate chain. Note that the test button will produce an error, as the API endpoint require a PATCH request with additional data supplied.

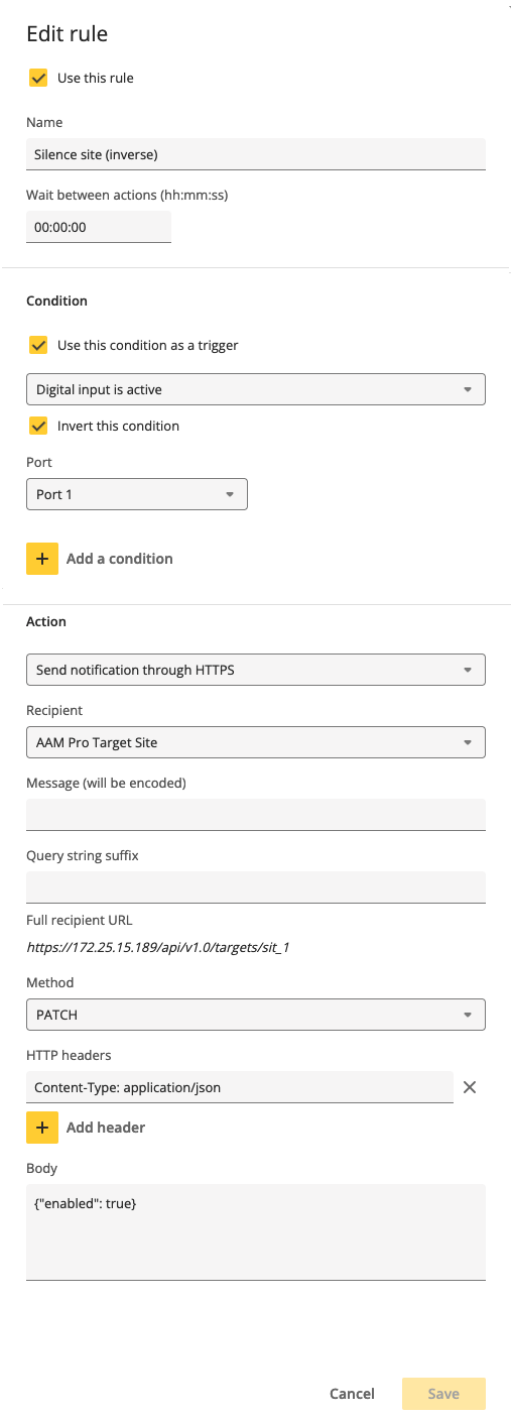


3. Create a rule in System>Events>Rules.
   Craft the rule to silence the site, based on if the digital input is activated. Make a PATCH request setting the "enable" parameter to false for the site and use the content header: Content-Type: application/json, like in the image below. A disabled site or zone will not play anything in the speakers. When enabled again, the playback will start again.



Note! Don't forget to enable **Use this rule** & **Use this condition as a trigger.** Set the **Wait between actions** to the minimal time wanted between triggering this action.

4. If you would like to enable the site again when the digital input goes back to a non-active state, setup another rule like below. Here we invert the condition by checking the checkbox, and we enable the site by sending true instead of false as the value of "enabled" in the json.

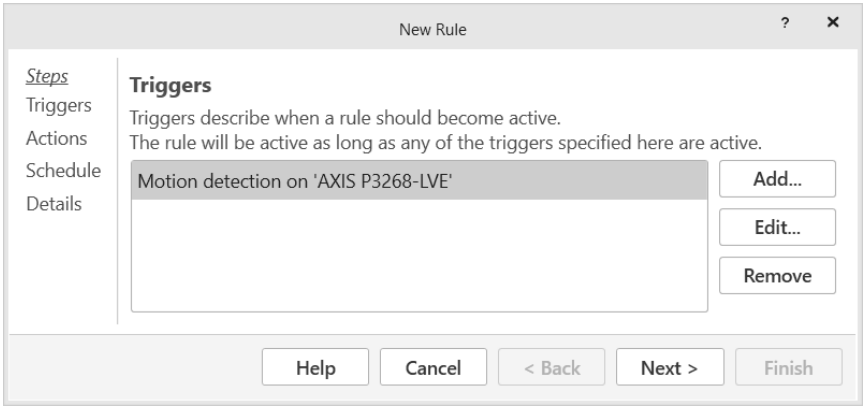


5. Done! Test your setup.

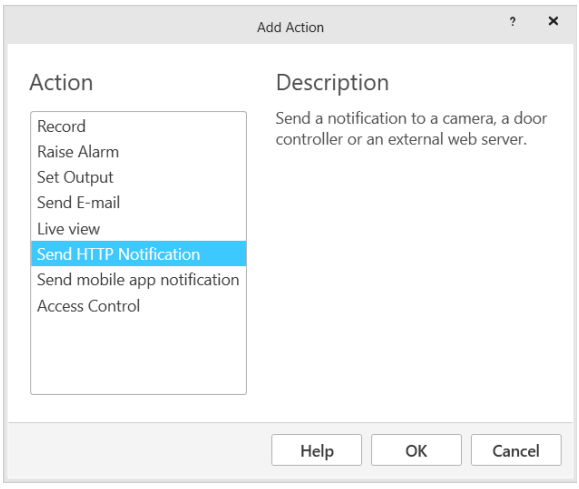
Example C: Initiate an audio file playback from AXIS Camera Station Pro

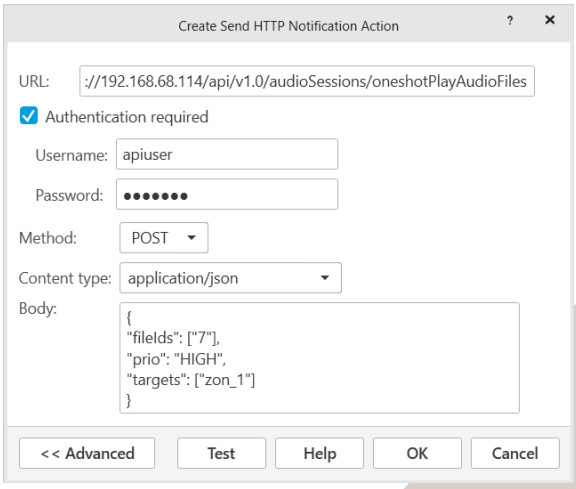
- In this example we will configure AXIS Camera Station Pro (ACS Pro) to call AXIS Audio Manager Pro API when a camera detects a motion.
- Note that ACS Pro only accepts requests to servers with a trusted certificate. If the certificate check fails it will give an error message saying “Failed to send notification”.

1. In ACS Pro, go to Recording and events > Action rules.
2. Create a new action rule. Select the trigger you want.



3. Press **Next** and add the action that should be triggered. Configure the action by adding the API call details. Press **Test** to execute the action and verify the result.





4. Press **OK** and save the rule. Done!

Example D: Initiate a repeating message when a button is pressed

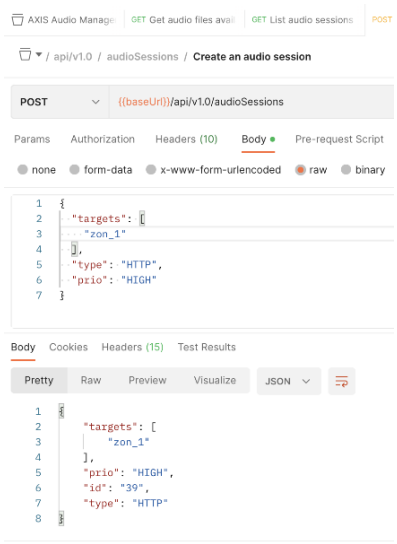
- Many Axis products have integrated digital input and output ports that enable connection to external devices such as physical buttons, detectors, lights, switches, and alarm relays. In this example we use a 2-state-button to initiate a continuous playback of two audio clips until the button turns it off.

1. Prepare the audio clips and upload them as an announcement in AXIS Audio Manager Pro.

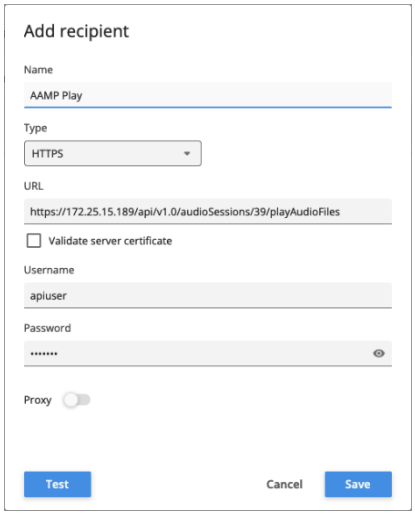
2. Prepare the command to play the audio clips.
    a. Find the id of the clips by making a request to the API through the browser: Enter the URL https://<server ip>/api/v1.0/audioFiles, enter the API username and password. A list of all the audio files on the server will be returned in the browser window. Find the file and note down its id.

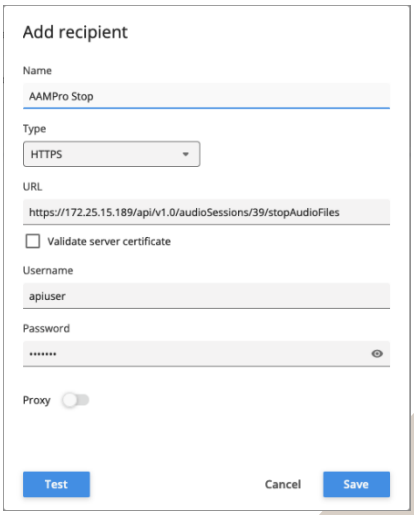
    b. Find the id of the target: Make a similar request to get all the targets available. Enter the URL https://<server ip>/api/v1.0/targets and extract the target id(s) you want to use and note them down.

3. Create an audio session that will be used repeatedly for the playback. To do this you need to make a POST request to the endpoint https://<serverip>/api/v1.0/audioSessions with a json payload. To accomplish this, Postman could be used. The target zones, together with prio and type should be sent in the json payload. Note that you must set your API username and password in Authorization→Digest Auth and probably disable SSL certficate verification in Settings. When the audio session is created you will get a Status 200 response and the json response like the picture below. Note the id of the returned session.

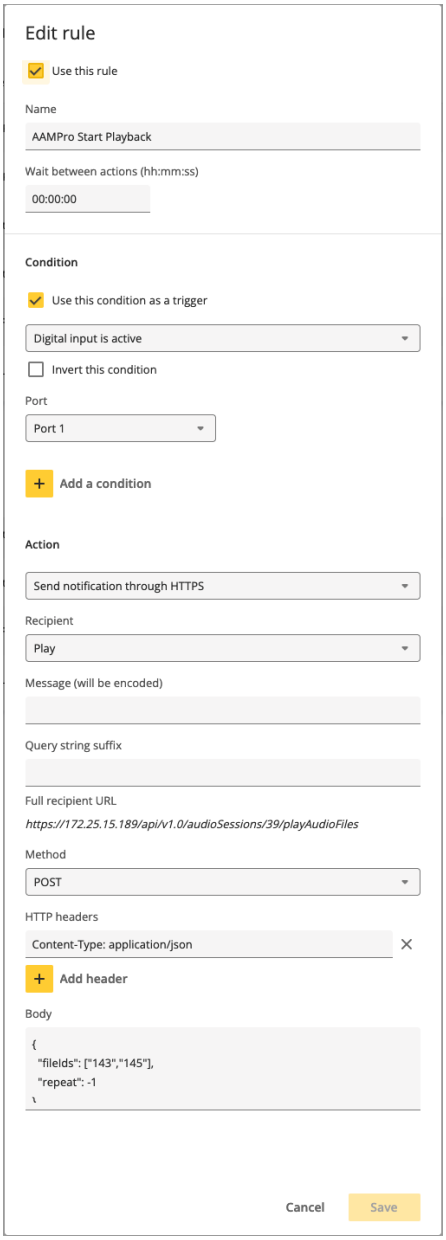


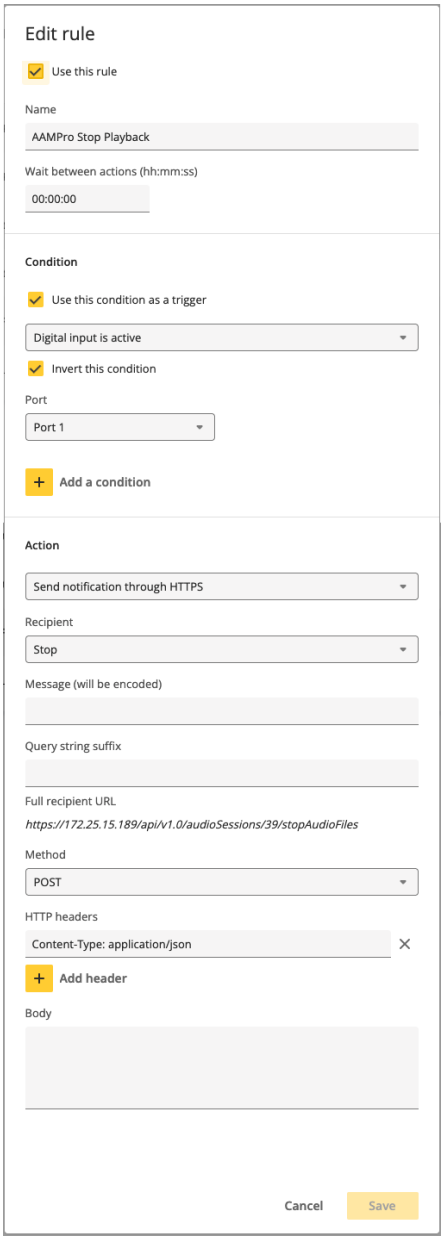
4. On the device that has the button connected to its IO port, add two recipients, one for starting and one for stopping the playback. Go to System>Events>Recipients and use the id of the audio session in the URL. Make sure to uncheck "Validate server certificate" if you do not have a trusted certificate chain set up.





5. On the device that has the button connected to its I/O port, configure two rules, one for playing and one for stopping the playback. Go to System>Events>Rules and craft the json that will be used in the play rule. Add the header: Content-Type: application/json. And remember to invert one of the conditions if you use "is active".





6. Done! You are now able to press the button to start playback and stop it by resetting the button to its original state.

Example E: Play an audio clip through a Python script

- In this example we will go through how to use Python to find the id of an audio clip and a zone, and then play the clip in that zone.
- This script finds the ids of the zone and audio files based on the nice names. It is possible to simplify the script if hard coding the ids.

```
import requests

# API endpoint URLs
BASE_API_URL = "https://172.25.15.189/api/v1.1/"
AUDIO_SESSIONS_API_URL = BASE_API_URL + "audioSessions"
AUDIO_FILES_API_URL = BASE_API_URL + "audioFiles"
TARGETS_API_URL = BASE_API_URL + "targets"
API_USER = 'apiuser'
API_PW = 'apipass'

# Get the ID of the audio file
audio_file_name = "sample-3s.mp3"
audio_file_id = None
try:
    response = requests.get(AUDIO_FILES_API_URL, auth=requests.auth.HTTPDigestAuth(API_USER, API_PW),
verify=False)

    response.raise_for_status()
    audio_files = response.json()
    audio_file_id = next((af["id"] for af in audio_files if af.get("name") == audio_file_name), None)
except requests.exceptions.HTTPError as e:
    print(f"Error getting audio files: {e}")
    exit(1)

if not audio_file_id:
    print(f"Error: Could not find audio file '{audio_file_name}'")
    exit(1)

# Get the ID of the target
target_name = "Aam PRO Output1"
target_id = None
try:
    response = requests.get(TARGETS_API_URL, auth=requests.auth.HTTPDigestAuth(API_USER, API_PW),
verify=False)
    response.raise_for_status()
    targets = response.json()
    target_id = next((t["id"] for t in targets if t.get("niceName") == target_name), None)
except requests.exceptions.HTTPError as e:
    print(f"Error getting targets: {e}")
```

```
    exit(1)

if not target_id:
    print(f"Error: Could not find target '{target_name}'")
    exit(1)

# Create and play the audio session
audio_session_data = {
    "prio": "LOW",
    "targets": [target_id],
    "fileIds": [audio_file_id]
}

try:
    response = requests.post(AUDIO_SESSIONS_API_URL + "/oneshotPlayAudioFiles", json=audio_session_data,
auth=requests.auth.HTTPDigestAuth(API_USER, API_PW), verify=False)
    response.raise_for_status()
    print(f"Success: {response}")
except requests.exceptions.HTTPError as e:
    print(f"Error creating audio session: {e}")
    exit(1)
```

Example F: Change the volume with action engine

- The Volume Controllers API can be used to increasing and decreasing the output volume in a whole or parts of a site.
- In this example we will go through how to change the volume during a certain time interval.
- The action engine that is available in most Axis devices is used in this example.

1. In AXIS Audio Manager Pro UI, create a volume controller for the zone or individual devices that should be affected by the action. It is also possible to set what content types (music, announcement, paging) the volume controller should control.

2. Make a request in the web browser to get the id of all the available volume controllers: https://<servername>/api/v1.1/volumeControllers. Extract the id(s) you want to use.

3. On the device that should send the command to increase the volume:
    a. Create a schedule in System>Events>Schedules of when the volume should be increased.

    b. Create a recipient in System>Events>Recipients.
    Create a recipient of every API endpoint you plan to use, by entering the URL: https://<servername>/api/v1.1/volumeControllers/{volumeControllerId}
    Uncheck “Validate server certificate” unless you have set up a trusted chain of certificate.

c. Create a rule in System>Events>Rules.
Choose the condition **Schedule**. Choose the action **Send notification through HTTPS** and use the recipient from above. Choose the PATCH method and set the header:
Content-Type: application/json
Craft the body as a json according to the API specification.

This example will set the offset of the volume controller to 40 units (louder than the calibrated value 0) on a scale from -100 to +100:

```
{
    "muted": false,
    "volumeOffset": "40"
}
```

d. Create another rule that decreases the volume again when the rush hour is over.

e. Done!

Example G: Device health

- This is a Python script that checks the status (online/playing/offline) for all devices of the audio site and outputs the results. Note that IPs and API credentials needs to be updated according to your installation. A script like this can be run regularly and be updated to set off a warning email or alarm when devices go offline. The example can also be expanded to use the API websockets to get continuous updates when then status of devices changes, removing the need to poll the status repeatedly.

```
import requests

# API endpoint and credentials
TARGETS_API_URL = "https://172.25.15.189/api/v1.1/targets"
API_USER = 'apiuser'
API_PW = 'apipass'

def fetch_targets(url, user, password):
    """Fetch targets from the API with HTTP Digest Authentication."""
    try:
        response = requests.get(url, auth=HTTPDigestAuth(user, password), verify=False)
        response.raise_for_status()
        return response.json()
    except requests.RequestException as err:
        print(f"Error getting targets: {err}")
        exit(1)

def count_device_statuses(devices):
    """Count the statuses of devices."""
    status_counts = {
        'online': 0,
        'playing': 0,
        'offline/error': 0
    }
    for device in devices:
        status = device.get('status', 'offline/error').lower()
        if status == 'online':
            status_counts['online'] += 1
        elif status == 'playing':
            status_counts['playing'] += 1
        else:
            status_counts['offline/error'] += 1
    return status_counts

def main():
    """Main function to fetch targets and count device statuses."""
    targets = fetch_targets(TARGETS_API_URL, API_USER, API_PW)
    devices = [target for target in targets if target['type'] == 'device']
```

```
    status_counts = count_device_statuses(devices)
    print(f"Device status counts: {status_counts}")

if __name__ == '__main__':
    main()
```

Troubleshooting of your API implementation

This section contains a few tips on how to solve some problems you may experience. Please contact Axis support if problems persist, or for any other questions regarding Axis devices or AXIS Audio Manager Pro APIs.

1. Make sure the API is enabled under System Settings > API Access

2. Note that most parameters are case sensitive. Check the case one more time.

3. It is easy to forget a, or " or {. Validate your json to make sure there are no mistakes. A good webpage for this is https://jsonlint.com/.

4. Make sure that you are using HTTPS and not HTTP to send all the requests.

5. For device action engine troubles
    a. Make sure you unchecked the "Validate server certificate" for the recipient.

    b. You can check the device log in System>Logs>View the system log. Search for lines like this one, and check if there is any information about the result.

    [NOTICE] actionengined-user[867]: Action rule "<<x>>" is starting action "Send notification through HTTPS"

    If there is no such line in the log, the action may not have been triggered at all.

6. Make sure you have set the header Content-Type: application/json for all POST and PATCH requests.

7. Test if your request works in a tool like Postman. Postman is a great tool for trying out API requests and we recommend using it to craft your request before moving it to an action engine.

8. Upgrade the firmware of the devices and use the factory default option. Also update the version of AXIS Audio Manager Pro and clean the database.