RESTful API

Guide

For additional support, call (604) 454-3792 or email support@algosolutions.com

AL061-GU-GF000API-001-R0
Firmware Version 5.2
support@algosolutions.com
May 29, 2023

Algo Communication Products Ltd.
4500 Beedie Street, Burnaby
V5J 5L2, BC, Canada
1-604-454-3790
www.algosolutions.com

Information Notices

Note

Note indicates useful updates, information, and instructions that should be followed

Disclaimer

The information contained in this document is believed to be accurate in all respects but is not warranted by Algo. The information is subject to change without notice and should not be construed in any way as a commitment by Algo or any of its affiliates or subsidiaries. Algo and its affiliates and subsidiaries assume no responsibility for any errors or omissions in this document. Revisions of this document or new editions of it may be issued to incorporate such changes. Algo assumes no liability for damages or claims resulting from any use of this manual or such products, software, firmware, and/or hardware.

No part of this document can be reproduced or transmitted in any form or by any means – electronic or mechanical – for any purpose without written permission from Algo.

For additional information or technical assistance in North America, please contact Algo's support team:

Algo Technical Support
1-604-454-3792
support@algosolutions.com

Table of Contents

1 GENERAL

1.1 Introduction

This document describes how Algo RESTful API can be used to access, manipulate, and trigger actions on Algo IP Endpoints on your network through HTTP/HTTPS requests, as well as a couple of different authentication methods with varying levels of security.

Requesting systems can interact with Algo devices through a uniform and predefined set of stateless operations defined in this document. Requests are made to a resource's URI with a JSON payload and elicit a JSON response. HTTP/HTTPS GET, POST, and PUT requests are made to resource URI along with the JSON payload (see commands section for a list of payloads).

1.2 Authentication

There are three types of authentications:

- Standard (recommended)
- Basic
- None (not recommended)

The Standard authentication uses a Hash-based Message Authentication Code (HMAC) with an SHA-256 encoded digest. Basic authentication uses Base64 encoding and should only be used over HTTPS. No authentication should only be used with extreme care as it provides no authentication. See the Authentication Requirements section for more details.

2 SETUP AND CONFIGURATION

2.1 Prerequisites

- This document assumes the Algo endpoint is running firmware version 3.3 or higher.
- The time difference between the requestor and the Algo devices should be less than 30 seconds to use standard authentication.
- Ensure NTP (Network Time Protocol) is in use. The addresses of custom NTP servers may be configured in the *Advanced Settings* → *Time* tab.

Note

The pre-configured NTP servers are publicly hosted, therefore internet connection is required to reach it. If no internet connection is available, configure a local NTP server and enter its IP address.

- Ensure the Algo device system time is adjusted to the correct time zone. This can be done by navigating to the *Advanced Settings* → *Time* tab.

2.2 Enabling the RESTful API

1. Log into the web interface and navigate to the *Advanced Settings* → *Admin* tab.

2. Scroll down to the **API Support** section, enable the RESTful API and set the Password as desired (deafault password: algo)

Note

Standard authentication is enabled by default.

2.3 Enable Basic Authentication (Optional)

1. In the web interface, navigate to the *System* → *Maintenance* tab and download the configuration file.
2. Open the configuration file with any text editor and add the following line: **api.auth.basic = 1**
3. Save and upload the modified configuration file back to the device using the **Restore Configuration File** feature in the *System* → *Maintenance* tab.

2.4 No Authentication Method (Optional)

To enable the no authentication method, leave the **RESTful API Password** field empty. This method is not recommended and should only be used for testing purposes only as it provides no security.

2.5 Enabling Simple Control Interface (Optional)

1. On the web interface, navigate to the *System* → *Maintenance* tab and download the configuration file.
2. Open the configuration file using a text editor and add two lines. Change the <pw> to your desire password.

   Admin.web.sci = 1

   Sci.admin.pwd = <pw>
3. Save and upload the modified configuration file back to the device using the **Restore Configuration File** feature in the *System* → *Maintenance* tab.

3 AUTHENTICATION REQUIREMENTS

Please email support@algosolutions.com if you would like a standard or basic authentication sample code.

3.1 Standard Authentication Request with a JSON Payload

Required headers in HTTP/HTTPS request

> Content-Type: “application/json”

> Content-MD5: [content_md5]

Example

Content-MD5: 74362cc86588b2b3c5a4491baf80375b

> Authorization: hmac admin:[nonce]:[hmac_output]

The authorization headers consist of:

1. The string 'hmac admin' followed by a colon ':'.
2. Nonce – A random or non-repeating value, followed by a colon ':'.
3. Hmac_output – generated by the RESTful API Password (secret-key) configured on your device and the HMAC input, as per below:

   [request_method]:[request_uri]:[content_md5]:[content_type]:[timestamp]:[nonce]

HMAC input example: (using 'algo' as the secret key)

POST:/api/controls/tone/start:6e43c05d82f71e77c586e29edb93b129:application/json:1601312252:49936

Generate HMAC with password and HMAC input string as digest using SHA-256:

HMAC output example:

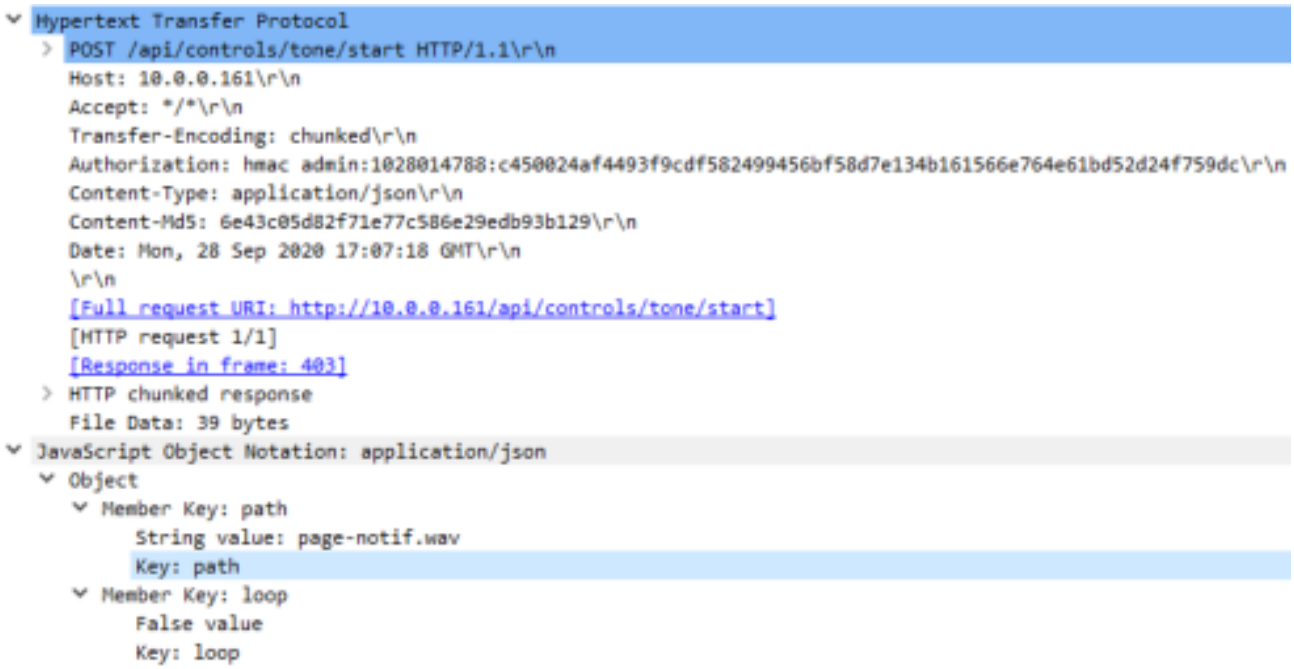
2e109d7aeed54a1cb04c6b72b1d854f442cf1ca15eb0af32f2512dd77ab6b330

> Date: day, date month, year hr:min:sec GMT

Example

Date: Thur, 22 Sept, 2022 02:33:07 GMT

Standard authentication with payload example:



3.2 Standard Authentication Request Without a JSON Payload

Identical to 3.1 with content related headers/hmac input omitted.

HMAC input: [request_method]:[request_uri]:[timestamp]:[nonce]

HMAC input example: (using 'algo' as the secret key)

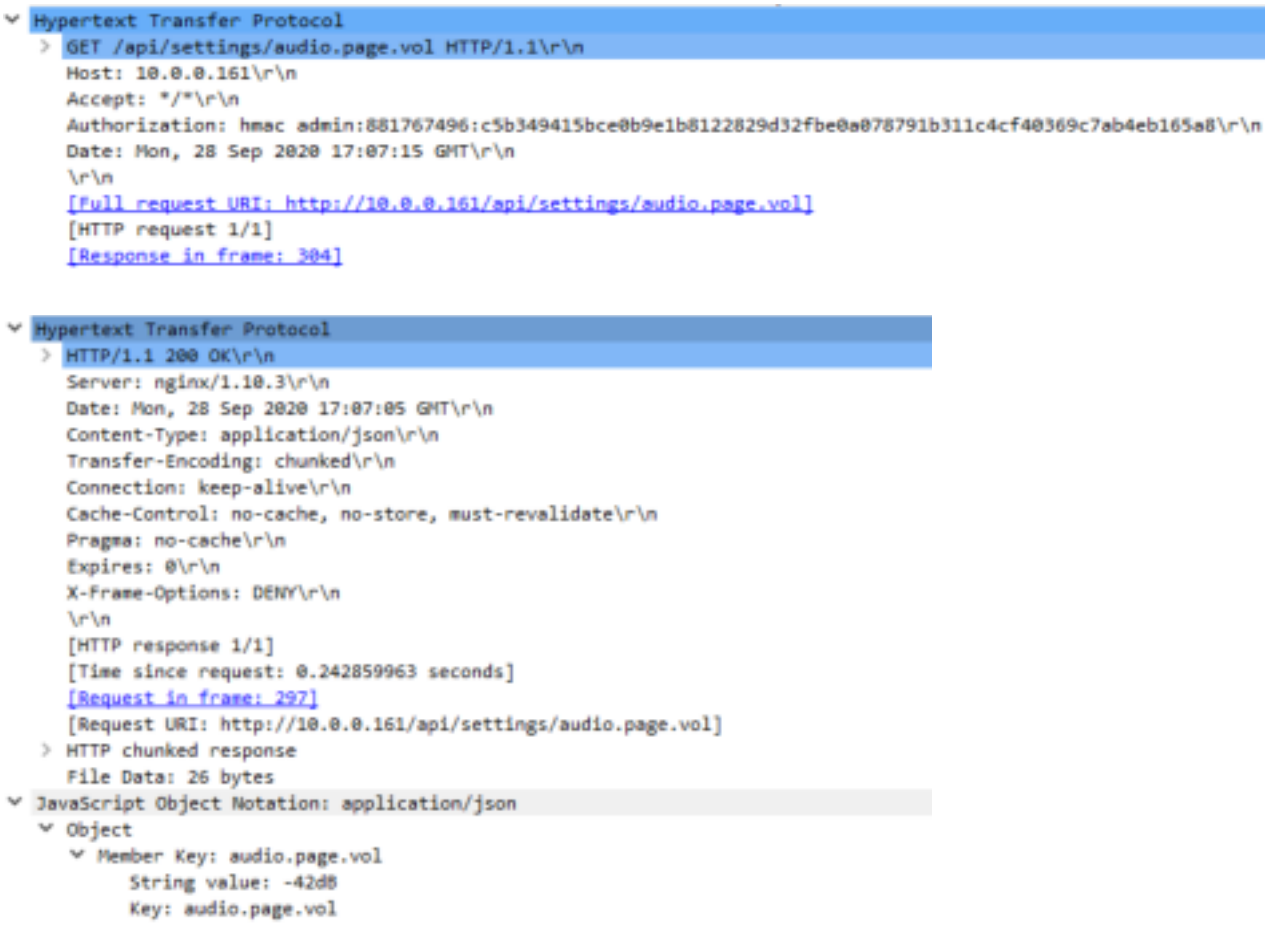
GET:/api/settings/audio.page.vol:1601312252:49936

Generate HMAC with password and HMAC input string using SHA-256:

HMAC output example:

c5b349415bce0b9e1b8122829d32fbe0a078791b311c4cf40369c7ab4eb165a8

Standard authentication without payload example:



3.3 Basic Authentication Request

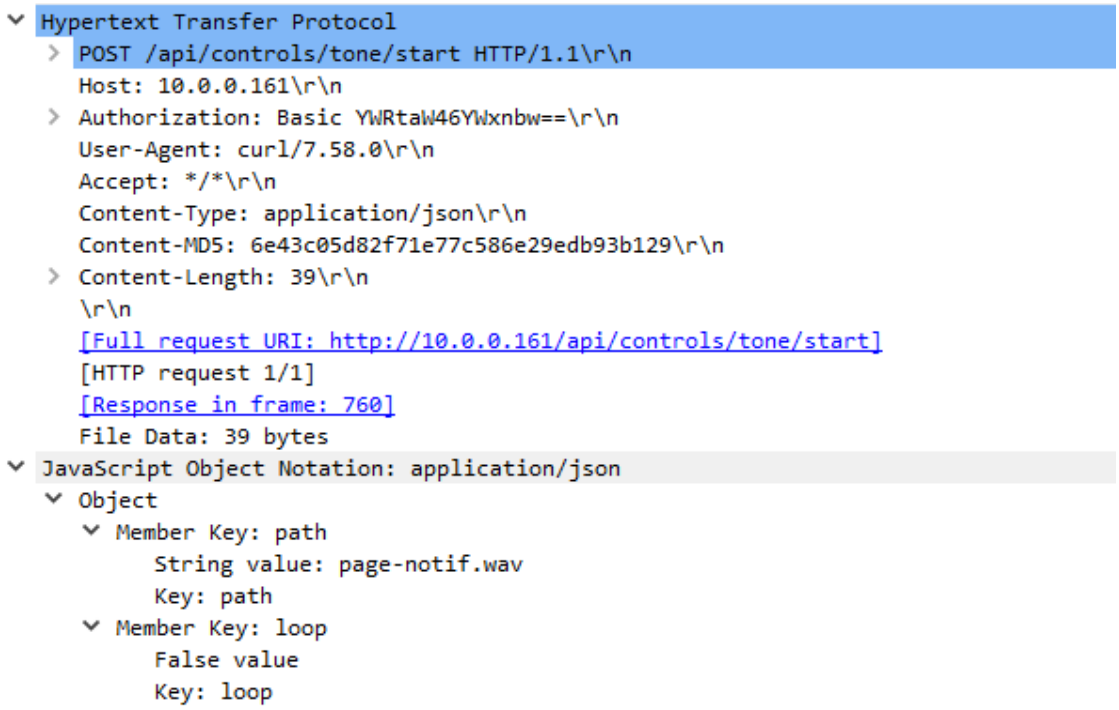
This method of authentication should be used with care as it is less secure than the standard method.

> Authorization: Basic [base64]

Example:

Authorization: Basic YWRtaW46YWxnbwo=

Basic authentication example:



4 COMMANDS

4.1 RESTful API Commands

Below is a list of all supported API commands.

Note

A PUT request changes or creates a permanent resource that survives a reboot, while a POST request only controls the device for the current session.

Description	Method	URI	Payload Parameters	Return Example	Product	FW
Retrieve the value of a specific parameter.	GET	/api/settings/[key-name] Ex./api/settings/audio.page.vol	N/A	{"audio.page.vol": "-18dB"}	All	> 3.3
Return the ambient noise level measured in decibels. Ambient Noise Compensation must be enabled in Basic Settings -> Features tab.	GET	/api/info/audio.noise.level	N/A	{"audio.noise.level": 72}	Speakers Display Speakers	> 3.3
Extract the status of the relay input terminal.	GET	/api/info/input.relay.status	N/A	{"input.relay.status": "idle"} or {"input.relay.status": "active"}	All products with a relay input, except the 8063. See below.	> 4.1
Extract the status of Input 1 or Input 2 terminals.	GET	/api/info/input.relay1.status or /api/info/input.relay2.status	N/A	{"input.relay1.status": "idle"} or {"input.relay1.status": "active"}	8063	> 4.1
Retrieve the list of tone files currently installed.	GET	/api/info/tonelist	N/A	{"tonelist":["bell-na.wav","bell-uk.wav","buzzer.wav",...]}	All	> 5.0
Retrieve the device information that is displayed on the Status page.	GET	/api/info/status	N/A	Full list of information from the Status tab.	All	> 5.4

Description	Method	URI	Payload Parameters	Return Example	Product	FW
Retrieve the product information that is displayed on the About page.	GET	/api/info/about	N/A	All information present on the About tab.	All	> 5.4
Activate the strobe with desired color and pattern parameters.	POST	/api/controls/strobe/start	pattern: {0 - 15} color1: {blue, red, amber, green} color2: {blue, red, amber, green} ledlvl: {1 - 255} holdover: {true, false}	N/A	8128(G2) 8138 8190S	> 3.3
Stop the strobe.	POST	/api/controls/strobe/stop	N/A	N/A	8128(G2) 8138 8190S	> 3.3
Play a tone once or loop it.	POST	/api/controls/tone/start	path: {tone} ie. chime.wav loop: {true, false} or {0, 1} e.g. {"path":"chime.wav", "loop":true}	N/A	Speakers 8301 8373 8028(G2) 8201 8039	> 3.3
Stop the tone.	POST	/api/controls/tone/stop	N/A	N/A	Speakers 8301 8373 8028(G2) 8201 8039	> 3.3
Call a phone extension with a pre-recorded message.	POST	/api/controls/call/start	{"extension":"2099", "tone":"gong.wav", "interval":"0", "maxdur":"10"}	N/A	Speakers 8301 8410 8420	> 3.3
End the call.	POST	/api/controls/call/stop	N/A	N/A	Speakers 8301 8410 8420	> 3.3

Description	Method	URI	Payload Parameters	Return Example	Product	FW
Initiate a one-way page call. The device will receive the audio stream from target extension.	POST	/api/controls/call/page	{"extension":"<ext>"}	N/A	Speakers 8410 8420	> 5.3.4
Reboot the target endpoint.	POST	/api/controls/reboot	N/A	N/A	All	> 3.3
Unlock the door. "local" controls the local relay "netdc1" controls the remote network door controller (8063)	POST	/api/controls/door/unlock	doorid: {local, netdc1} *Optional	N/A	8039 8028(G2) 8201 8063	> 3.3
Lock the door.	POST	/api/controls/door/lock	doorid: {local, netdc1} *Optional	N/A	8039 8028(G2) 8201 8063	> 3.3
Enable the 24v aux out relay.	POST	api/controls/24v/enable	N/A	N/A	8063	> 5.0
Disable the 24v aux out relay.	POST	api/controls/24v/disable	N/A	N/A	8063	> 5.0
Enable the output relay.	POST	/api/controls/relay/enable	N/A	N/A	8063	> 5.0
Disable the output relay.	POST	/api/controls/relay/disable	N/A	N/A	8063	> 5.0
Check Algo's firmware server for latest firmware version.	POST	/api/controls/upgrade/check	N/A	{"version": "updated"} or {"version": "<fw version>"}	All	> 4.1
Check Algo's firmware server for latest firmware version and upgrade to that version.	POST	/api/controls/upgrade/start	N/A	{"status": "updated"} or {"status": "upgrading <fw version>", "url": <download url>} or {"status": "<message>"}	All	> 4.1
Display an image or pattern on the screen.	POST	/api/controls/screen/start	See below	N/A	8410 8420	> 5.3.4
Stop the screen pattern and return to the default screen.	POST	/api/controls/screen/stop	N/A	N/A	8410 8420	> 5.3.4

Description	Method	URI	Payload Parameters	Return Example	Product	FW
Restart the main application.	POST	/api/controls/reload	N/A	N/A	All	> 5.3.4
Start listening to a direct audio stream. Configure the port number to which the stream is being sent.	POST	/api/controls/rx/start	{"port": <port>}	N/A	All	> 5.3.4
Stop listening to a direct audio stream.	POST	/api/controls/rx/stop	N/A	N/A	All	> 5.3.4
Set the multicast mode.	PUT	/api/state/mcast/update/	{"mode":"sender", "address":<address>, "port":<port>, "type":"rtp"} or {"mode":"sender", "address":<address>, "port":<port>, "type":"poly", "group":1} **Note**: If controls/tone/start is used before this command, the tone will play using current settings on the web UI.	N/A	8301	> 5.0
Insert a value to a specific parameter from JSON payload.	PUT	/api/settings	parameter: {value} e.g. {"audio.page.vol": "-3dB"}	N/A	8180(G2) 8186 8190 8190S 8301 8373	> 3.3

4.2 Simple Control Interface (SCI) Commands

All SCI commands are GET requests and have the common parameters “usi” and “admin” for authentication.

Example:

GET http://<IP>/sci/controls/door/unlock?usr=admin&pwd=algo&doorid=local

Description	URI	Additional Payload Parameters	Products	FW
Unlock the door. “local” controls the local relay “netdc1” controls the remote network door controller (8063)	/sci/controls/door/unlock	doorid: {local, netdc1} *Optional	8039 8028(G2) 8201 8063	> 3.3
Lock the door.	/sci/controls/door/lock	doorid: {local, netdc1} *Optional	8039 8028(G2) 8201 8063	> 3.3
Play a tone once or loop it.	/sci/controls/tone/start	path: {tone} ie. chime.wav loop: {true, false} or {0, 1}	All	> 3.3
Stop the tone.	/sci/controls/tone/stop	N/A	All	> 3.3
Activate the strobe with desired color and pattern parameters.	/sci/controls/strobe/start	pattern: {0 - 15} color1: {blue, red, amber, green} color2: {blue, red, amber, green} ledlvl: {1 - 255} holdover: {true, false}	8128(G2) 8138 8190S	> 3.3
Stop the strobe.	/sci/controls/strobe/stop	N/A	8128(G2) 8138 8190S	> 3.3