


SV600 API

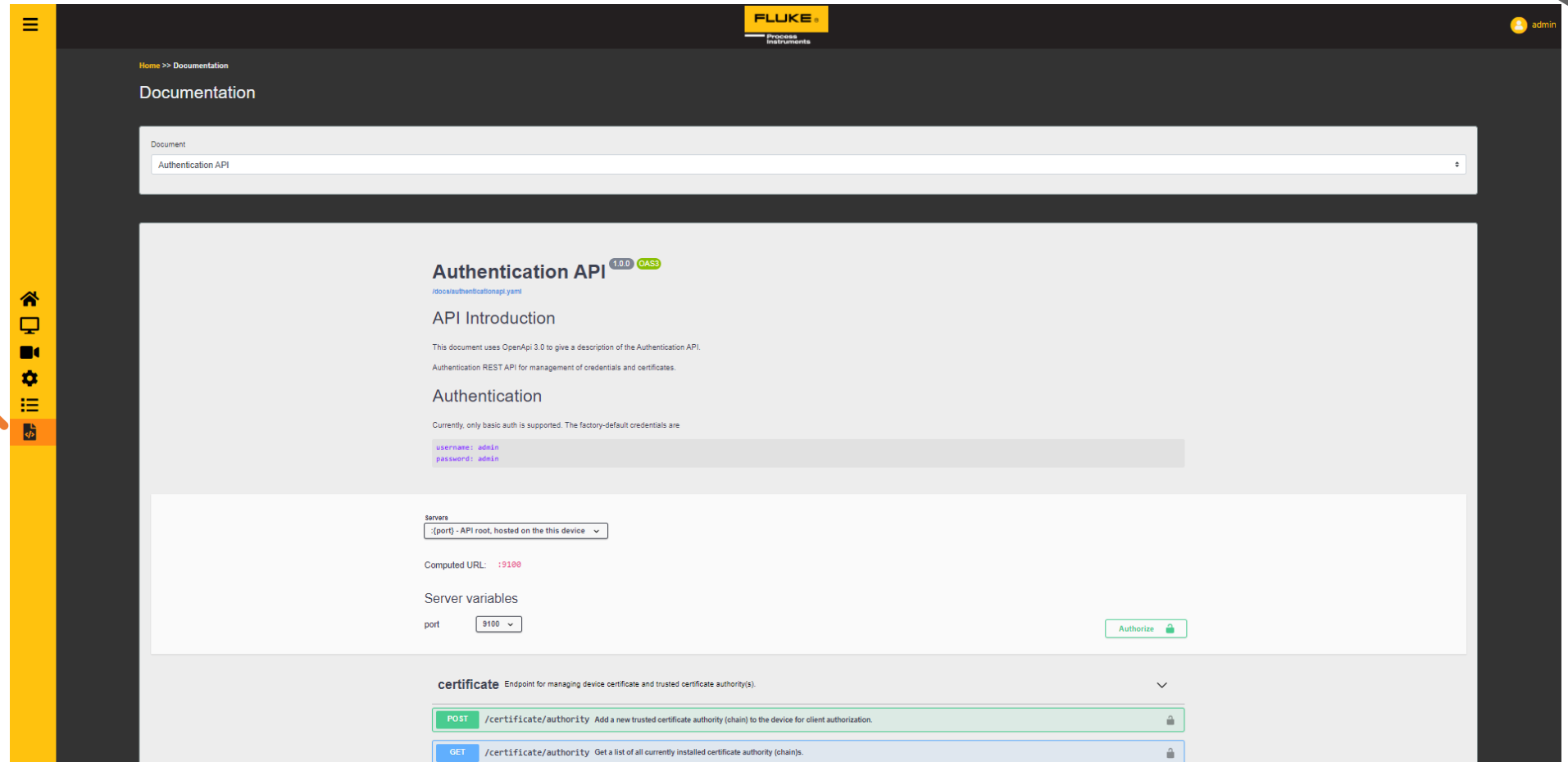
FLUKE®

Process
Instruments

SV600 API

To access the documentation about the API click the  Button In the menu:

- Documentation API Overview



Home >> Documentation

Documentation

Document

Authentication API

Authentication API ^{1.0.0} OAS3

[/doc/authenticationapi.yaml](#)

API Introduction

This document uses OpenApi 3.0 to give a description of the Authentication API.

Authentication REST API for management of credentials and certificates.

Authentication

Currently, only basic auth is supported. The factory-default credentials are

```
username: admin
password: admin
```

Servers

{port} - API root, hosted on the this device

Computed URL: :9100

Server variables

port 9100

Authorize

certificate Endpoint for managing device certificate and trusted certificate authority(s).

POST /certificate/authority Add a new trusted certificate authority (chain) to the device for client authorization.

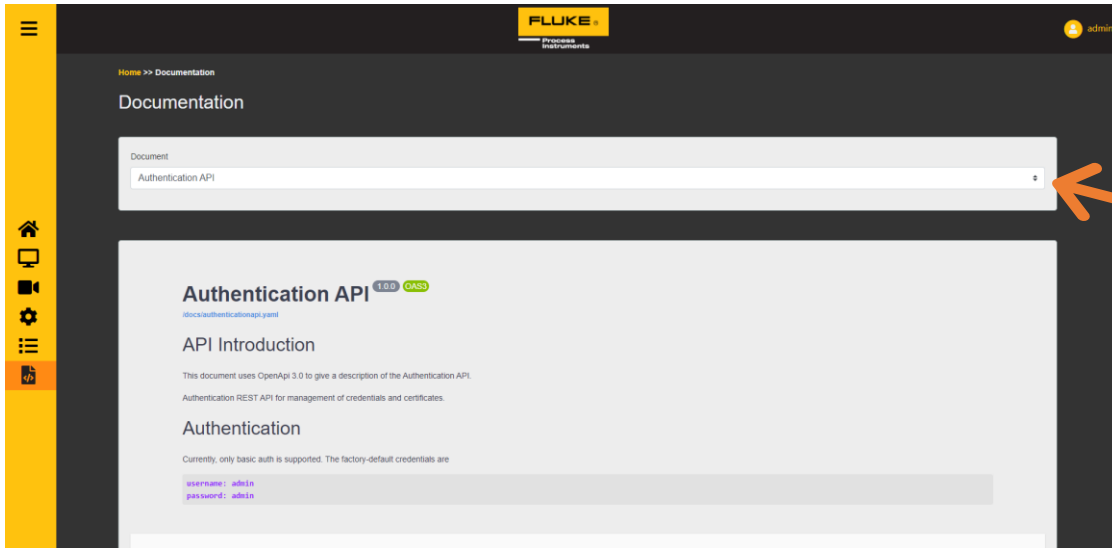
GET /certificate/authority Get a list of all currently installed certificate authority (chains).

SV600 API

(Application Programming Interface)

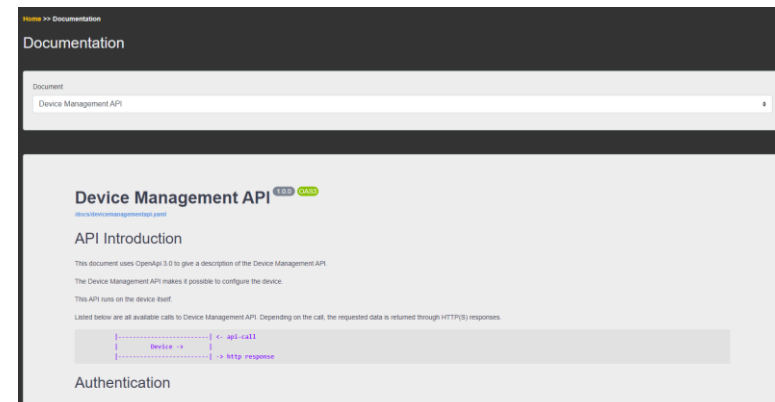
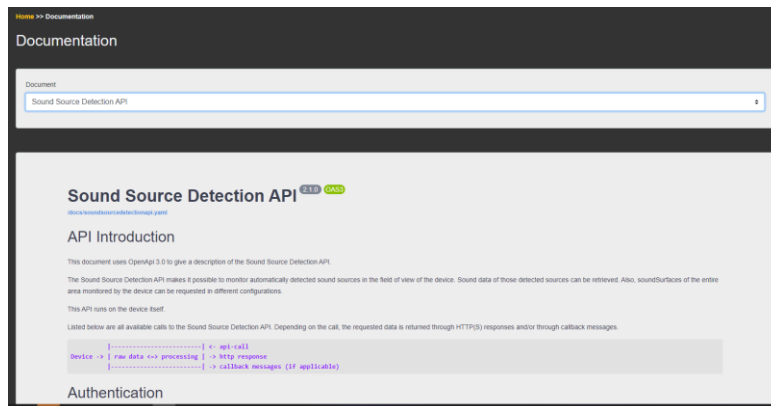
- The API is based on a HTTP REST API. All calls are communicated via HTTP or WebSocket.
- The API protocol is split in three levels. The device hosts the following documents:
 - 1. Sound Source Detection API
 - 2. Authentication API
 - 3. Device manager
- Everything what is visible in the frontend (Wen Interface) can also be taken from the API

SV600 API



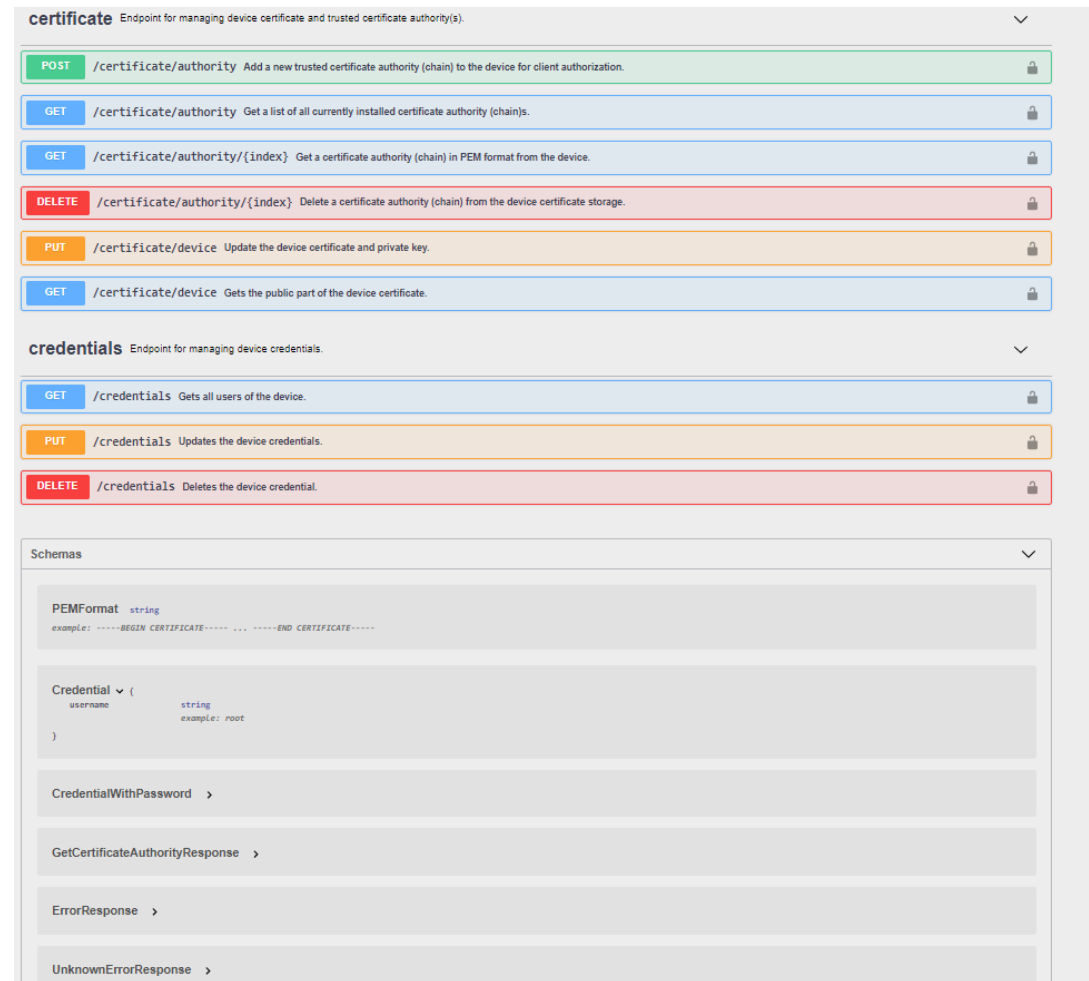
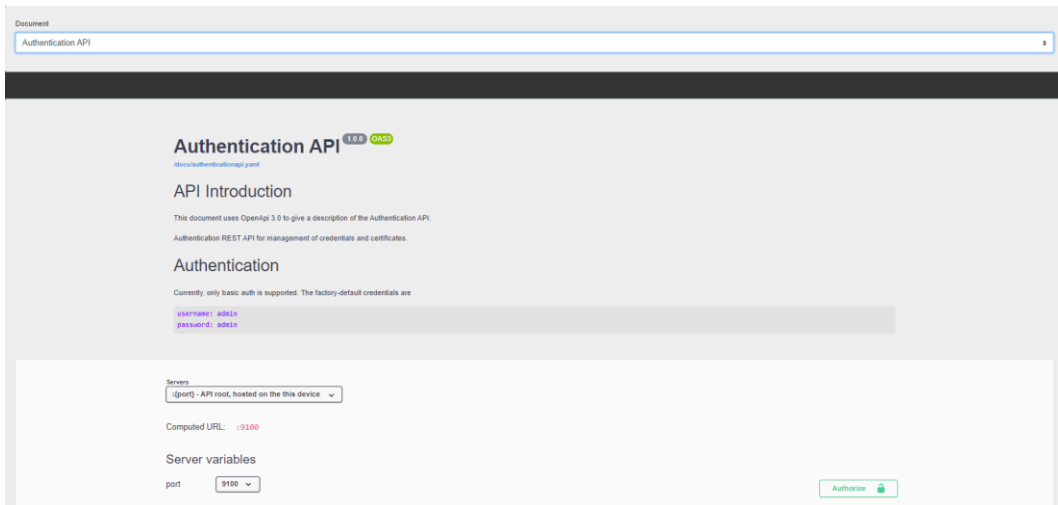
Click on the small arrows to get the selection menu for the 3 API levels:

- Authentication API
- Device manager
- Sound Source Detection API



API Overview

- Authentication service



API Overview

- Sound Service Detection Service

Document

Sound Source Detection API

Sound Source Detection API 2.1.0 OAS3

[/docs/sound sourcedetectionapi.yaml](#)

API Introduction

This document uses OpenApi 3.0 to give a description of the Sound Source Detection API.

The Sound Source Detection API makes it possible to monitor automatically detected sound sources in the field of view of the device. Sound data of those detected sources can be retrieved. Also, soundSurfaces of the entire area monitored by the device can be requested in different configurations.

This API runs on the device itself.

Listed below are all available calls to the Sound Source Detection API. Depending on the call, the requested data is returned through HTTP(S) responses and/or through callback messages.

```

}-----} <- api-call
Device -> | raw data <-> processing | -> http response
}-----} -> callback messages (if applicable)
    
```

Authentication

Currently, only basic auth is supported. The factory-default credentials are

```

username: admin
password: admin
    
```

Credentials can be managed with the [Authentication API](#). The only exception for this is the /video call. For this a nonce authentication string is used that can be obtained via the call /video/nonce

Servers

[port] - API root, hosted on the this device

Computed URL: :9011

Server variables

port 9011

Authorize

actions Calls related to actions used for "if then that" functionality

- GET /actions Get a list of the current actions present in the system
- DELETE /actions/{id} Delete action with this id
- PUT /actions/{id} Update settings of an action

callbackchannels Calls related to callbackchannels used for data transfer

sensors Calls related to the different sensors present in the device

- GET /sensors Get a list with information of all sensors of the device

soundSurfaces Calls related to soundSurfaces generated by the device

- GET /data/soundsurface/{id} Get results from the specified time and duration. If no time and no duration are given, it gives back the most recent result. If a time is given, but no duration, it gives all data from the timestamp until 'now'. If a duration is given, but no timestamp, it gives all data from the past 'duration'
- POST /data/soundsurface/{id}/subscription Subscribe to receive SoundSurfaceDataMessages through the specified callback channel with the specified configuration
- DELETE /data/soundsurface/{id}/subscription/{subscriptionId} Unsubscribe from receiving the data stream of the subscription belonging to the specified subscriptionId
- GET /soundsurfaces Get a list of all soundSurfaces currently being generated by the device
- POST /soundsurfaces Add a new soundSurface configuration with which to create soundSurfaces to the device
- PUT /soundsurfaces/{id} Update settings of a soundSurface
- DELETE /soundsurfaces/{id} Delete a soundSurface configuration that was added using the POST /soundsurface call

subscriptions Calls related to subscriptions running on the device

measurements Calls related to measurements performed by the device

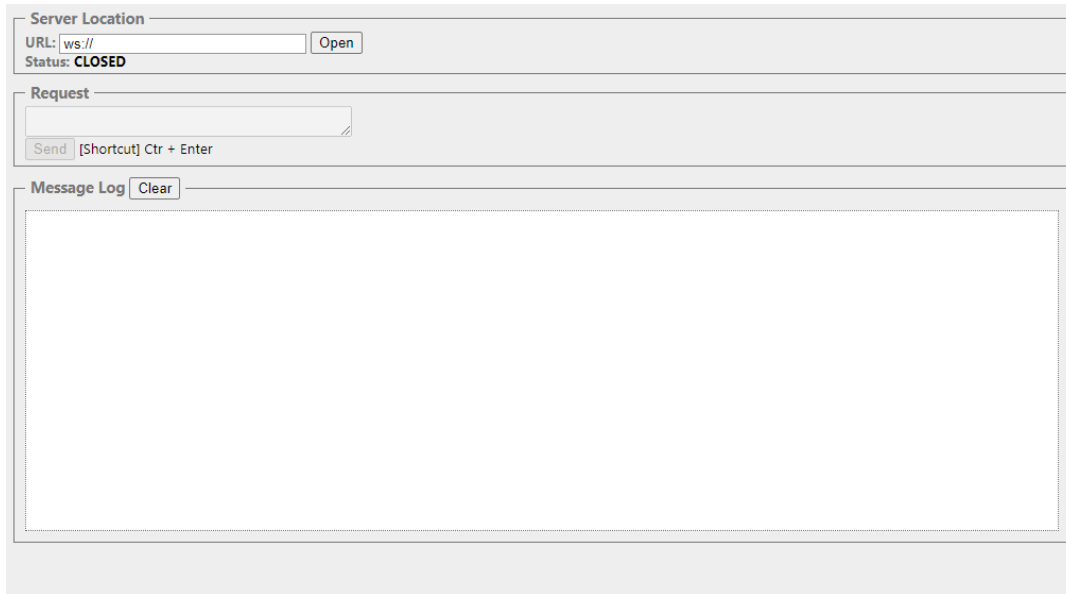
events Calls related to event triggers performed by the device

video Calls related to video streaming

Using Websocket with SV600

- Download Wescocket client in your browser
 - e.g Simple WebSocket Client in the Chrome webstore

Using Websocket with SV600



Server Location
URL:
Status: **CLOSED**

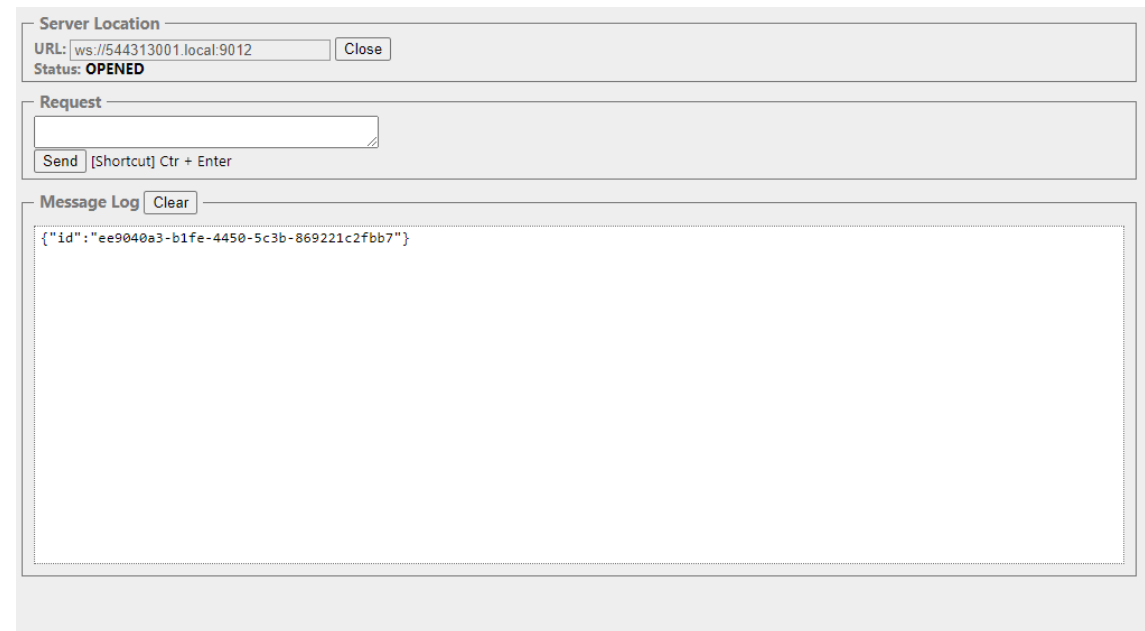
Request

 [Shortcut] Ctr + Enter

Message Log

The screenshot shows a web interface with three main sections. The top section, 'Server Location', has a text input field containing 'ws://' and an 'Open' button. Below it, the status is 'CLOSED'. The middle section, 'Request', has a text input field and a 'Send' button with the shortcut '[Shortcut] Ctr + Enter'. The bottom section, 'Message Log', has a 'Clear' button and a large empty text area.

Open connection to your SV600:
ws://SERIALNUMBER.local:9012
Or
ws://IP-ADRESS:9012



Server Location
URL:
Status: **OPENED**

Request

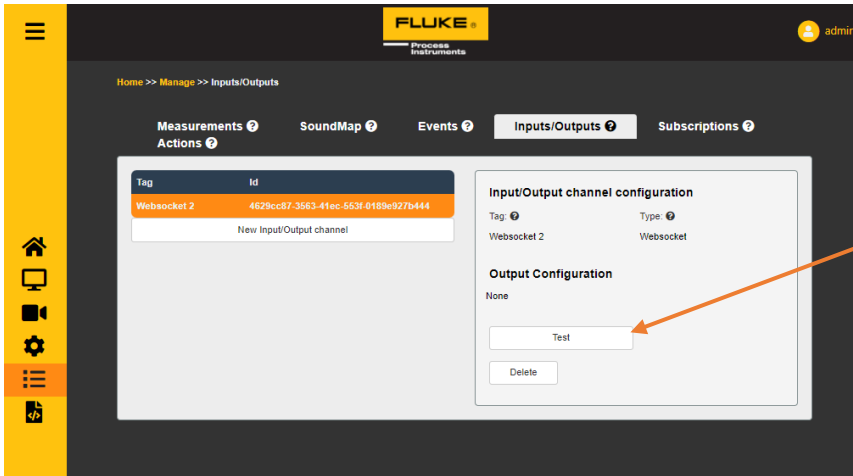
 [Shortcut] Ctr + Enter

Message Log

```
{"id": "ee9040a3-b1fe-4450-5c3b-869221c2fbb7"}
```

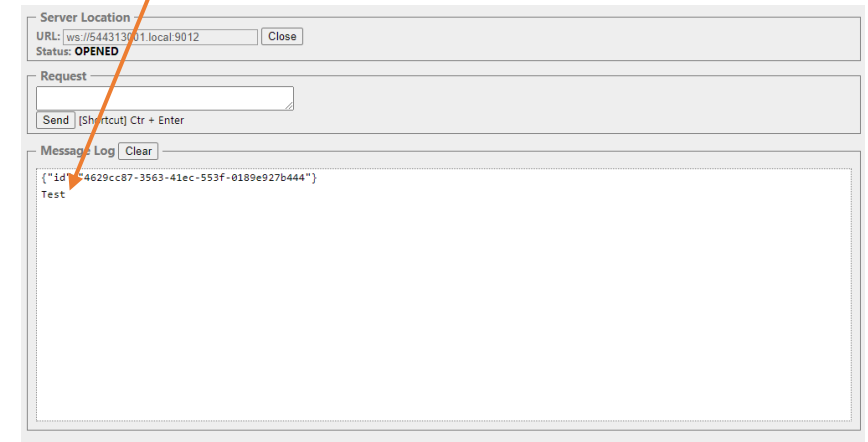
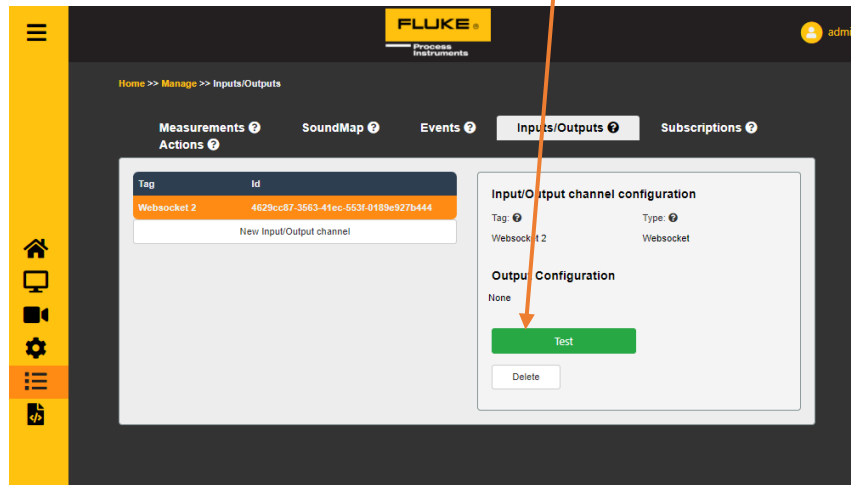
The screenshot shows the same web interface as the first one, but now the status is 'OPENED'. The 'Server Location' section shows the URL 'ws://544313001.local:9012' and a 'Close' button. The 'Request' section is empty. The 'Message Log' section now contains a JSON message: {"id": "ee9040a3-b1fe-4450-5c3b-869221c2fbb7"}.

Using Websocket with SV600



Go to Inputs/Outputs and check Websocket after connected to SV600

- Click on Test
- Test button will turn green
- Click Test button → Sent text „Test“ can be seen in WebSocket window



Using Websocket with SV600

Set up Event in Measurement here:
 Tag: Event Smartphone 22kHz
 Frequency Range between 20KHz to 23kHz

Link Measurement to Events:
 Here to measurement: „Event Smartphone 22kHz
 Set Threshold above 50 dB

