Sound location device API guide

Version 0.6

VERSION	DATE	AUTHOR(S)	COMMENTS
0.1 - Draft	07/15/2022	DT	Draft release
0.2 – Edits	08/12/2022	JD	Edited for writing style/grammar
0.3 – Edits	09/01/2022	JLR/JD/OB	Edited for writing style/grammar/code updates
0.4 – Edits	09/01/2022	JLR/JD/OB	Edited for code updates and terminology
0.5 – Edits	10/20/2022	JD	Added HDL410 systems
0.6 – Edits	05/18/2023	MK/OB/DD/JD	Added HDL310 systems and room layout endpoint; copy edited

Table of Contents

Introduction

The sound location device API runs within Nureva® Console software. As a local service API, it monitors and reports sound locations as registered by Nureva audio conferencing systems to any connected client. The server provides a simple set of APIs that allows clients to connect, query device and room information and receive streaming sound location events.

Prerequisites

The following prerequisites must be met in your application before you can use the sound location device API:

1. A WebSocket client to communicate with the Nureva sound location service. A variety of open-source WebSocket libraries is available online for most popular programming languages.
2. Nureva Console gives you the ability to configure access and ports used by the service. Your application must be able to connect on the port configured, using the protocols as defined in the section below.

Integration specification

Requests and responses will be in JSON format and included in the WebSocket message sent to the server. They will be in plain text with no authorization or encryption applied.

Authorization and access control will be provided via an allowlist of clients that can be configured through Nureva Console. Clients that have not been granted access to initiate requests to the server will be denied and will receive an unauthorized error response.

Request format

All requests made to the server (initiated by the client) will follow the format as detailed below:

```
{
   "request": string,
   "requestId": string,
   "clientId": string,
   "body": any | undefined
}
```

request: The path of the endpoint requested.

requestId: A value that a client should provide to associate a response back to the originating request. The value is echoed back in the response for a request where applicable. GUIDs work well for this purpose. The value is *required*, can't consist only of whitespace and is limited to a *maximum of 50 characters*. This should be unique per request, so the correlation between request and response can be maintained.

clientId: An identifier for an integrating client. The value is echoed back in the response to a request where applicable. The value is *required*, can't consist only of whitespace and is limited to *a maximum of 50 characters*. This should be unique per integration, but the same value can and should be used in each request made.

body: Any arguments necessary for the request will be included here. *Currently not required for audio streaming.*

Response format

All responses or messages originating from the server (and sent to clients) will follow the format as detailed below:

```
{
   "request": string,
   "requestId": string,
   "clientId": string,
   "body": any
}
```

request/requestId/clientId: The value is echoed back from the original request made to the service unless it was invalid.

body: The requested information is returned here.

Error response format

Error responses sent to client requests will follow the format below. A full table of status codes is included further below.

```
{
   "request": string,
   "requestId": string,
   "clientId": string,
   "errors": {
         "statusCode": number,
         "message": string
   }[]
}
```

request/requestId/clientId: The value is echoed back from the original request made to the service unless it was invalid.

statusCode: A numeric code indicating the specific error encountered with the request.

message: Contextual information about the error and what prompted it.

Connecting to the sound location device API service

Clients need to make an initial WebSocket connection to the configured IP and port of the server as defined in the configuration in Nureva Console. After a connection is established, clients can make requests to the server by using the request format described above.

Note that the WebSocket message is not compressed, so clients need to connect to the WebSocket server with the “perMessageDeflate:false” option.

A **ping** endpoint is supported, per the WebSocket specification. **Ping** messages sent in accordance with the WebSocket specification will receive a **pong** response back.

There is no automatic closing of this connection; it will be kept active as long as neither party has attempted to close it. The only scenario where it would be closed is if the Nureva Console desktop client is restarted; otherwise, we expect the connection to remain open and closure would be initiated by the integrating client.

Connection limits

The server itself has an internal limit on the number of open connections that can be made at any one time. Currently, this limit is 5 open connections at a time – additional connection attempts will be rejected until an open connection becomes available.

Authorization

Client authentication mechanisms are not required for device APIs over the local network. Requests will be in plain text and no password or credentials will be expected or supplied. The IP address or hostname of requesting clients will be checked against an allowlist. Requests will be rejected if the IP address or hostname is not on the list. This list is configurable within Nureva Console.

API specification

Sound location data stream

Once the initial request is made, sound location data registered by the connected audio device will start streaming at a regular interval. The data will be streamed provided the WebSocket connection is open and the data can be retrieved.

If an error occurs, the WebSocket will respond with an appropriate error code. If a Nureva device firmware update is in progress, an error code will be returned and sound location data streaming will be paused. Once the firmware update is complete, sound location data streaming will resume automatically.

```
Request
{
   "request": "v1/devices/audioLocation",
   "requestId": string,
   "clientId": string
}

Response
{
   "request": "v1/devices/audioLocation",
   "requestId": string,
   "clientId": string,
   "body": {
      "azimuth": number,
      "powerLevel": number,
      "coordinates": { "x": number, "y": number  } undefined,
      "time": string
   }
}
```

azimuth: Audio direction angle in degrees. Range -70 to 70, e.g., 65.

powerLevel: Sound power level in decibels, e.g., 30.

time: ISO 8601 time string for the time the location information was generated, e.g., "2022-06-07T09:01:23.486Z".

coordinates: (Optional.) The x, y coordinates of the location in millimeters. Available with HDL410 systems only. Bar 0's (i.e., Port 1) location is taken as the origin (0, 0) of the coordinate system, where positive y values are positioned in front of the bar, while negative and positive x values are to the left and right of the bar, respectively, when facing in the same direction as the bar.

Errors

- Status Code 10000 – Bad request
    - Request does not conform to the specification. The message field will include detailed information about which part of the request is incorrect.
- Status Code 11000 – Unsupported device
    - Connected hardware device does not support audio location streaming
- Status Code 11001 – Device not connected
    - No audio device is connected
- Status Code 11002 – Interrupted by firmware update
    - If a firmware update is being applied to a connected Nureva device, audio location information will be unavailable until that update is complete. An error code will be supplied.
- Status Code 11003 – Audio location unavailable
    - Audio location information is not available; a general use error in the event we cannot retrieve and send audio location
- Status Code 11004 – Microphone is muted
- Status Code 11005 – Speaker bar is disconnected

Sample requests

1. Valid request/response

Request


```
{
   "request": "v1/devices/audioLocation",
   "requestId": "3abe203s-42b7-4b0b-9awaf-5c381793a192",
   "clientId": "test"
}
```


Response


```
{
   "request": "v1/devices/audioLocation",
   "requestId": "3abe203s-42b7-4b0b-9awaf-5c381793a192",
   "clientId": "test",
   "body": {
      "azimuth": -35,
      "powerLevel": 0,
      "coordinates": {
         "x": -2890,
         "y": 2936,
      },
      "time": "2022-07-15T15:07:23.375Z"
   }
}
```


2. Invalid request – no requestId or clientId

Request


```
{
   "request": "v1/devices/audioLocation",
   "requestId": "",
   "clientId": ""
}
```


Response


```
{
   "request": "v1/devices/audioLocation",
   "requestId": "",
   "clientId": "",
   "errors": [
      {
         "statusCode": 10000,
```

```
      "message": "clientId: String can't be empty or whitespace"
    },
    {
      "statusCode": 10000,
      "message": "requestId: String can't be empty or whitespace"
    }
  ]
}
```


3. Unsupported device – note this is a valid request but returns an error due to an unsupported audio device.


```
Request
{
  "request": "v1/devices/audioLocation",
  "requestId": "3abe203s-42b7-4b0b-9awaf-5c381793a192",
  "clientId": "test"
}

Response
{
  "request": "v1/devices/audioLocation",
  "requestId": "3abe203s-42b7-4b0b-9awaf-5c381793a192",
  "clientId": "test"
  "errors": [
    {
      "statusCode": 11000,
      "message": "Unsupported device"
    }
  ]
}
```


Device information

Returns device information for the connected Nureva audio conferencing system, including the model and firmware version.


```
Request
{
  "request": "v1/devices/info",
  "requestId": string,
  "clientId": string
}

Response
{
  "request": "v1/devices/info",
  "requestId": string,
  "clientId": string,
  "body": {
    "model": string,
    "firmwareVersion": string
  }
}
```


model: The model of the default device. Possible values:

- hdl300
- dual-hdl
- hdl310
- hdl410

firmwareVersion: The firmware version of the default device. Format: x.y.z

Errors

- Status Code 10000 – Bad request
    - Request does not conform to the specification. The message field will include detailed information about which part of the request is incorrect.
- Status Code 11000 – Unsupported device
    - Connected hardware device does not support audio location streaming
- Status Code 11001 – Device not connected
    - No audio device is connected

Sample requests

1. Valid request/response

```
Request
{
   "request": "v1/devices/info",
   "requestId": "1234",
   "clientId": "1111"
}

Response
{
   "request": "v1/devices/info",
   "requestId": "1234",
   "clientId": "1111",
   "body": {
      "firmwareVersion": "3.1.9",
      "model": "hdl300"
   }
}
```

2. Invalid request – no requestId or clientId

```
Request
{
   "request": "v1/devices/info",
   "requestId": "",
   "clientId": ""
}

Response
{
   "request": "v1/devices/info",
   "requestId": "",
   "clientId": "",
   "errors": [
      {
         "statusCode": 10000,
         "message": "clientId: String can't be empty or whitespace"
      },
      {
         "statusCode": 10000,
         "message": "requestId: String can't be empty or whitespace"
      }
   ]
}
```

3. clientId parameter too long

```
Request
{
   "request": "v1/devices/info",
```

```
  "clientId": "This_id_is_longer_than_fifty_characters_and_is_too_long!"
}
```


Response


```
{
  "request": "v1/devices/info",
  "requestId": "",
  "clientId": "",
  "errors": [
    {
      "statusCode": 10000,
      "message": "clientId: String must contain at most 50 character(s)"
    },
    {
      "statusCode": 10000,
      "message": "requestId: Required"
    }
  ]
}
```


4. No device connected – note this is a valid request format, but no audio device is connected, so it returns an error

Request


```
{
  "request": "v1/devices/info",
  "requestId": "1234",
  "clientId": "1111"
}
```


Response


```
{
  "request": "v1/devices/info",
  "requestId": "1234",
  "clientId": "1111",
  "errors": [
    {
      "statusCode": 11001,
      "message": "Device not connected"
    }
  ]
}
```


5. Unsupported device – note this is a valid request format, but an unsupported audio device is connected, so it returns an error

Request


```
{
  "request": "v1/devices/info",
  "requestId": "1234",
  "clientId": "1111"
}
```


Response


```
{
  "request": "v1/devices/info",
  "requestId": "1234",
  "clientId": "1111",
  "errors": [
    {
      "statusCode": 11000,
      "message": "Unsupported device"
```

```
        }
      ]
    }
```

Room layout information

The connected Nureva audio conferencing system returns layout information of the room. Currently, only HDL410 systems support this feature. The information includes dimensions of the room and locations of the individual bars making up the system.

Rooms are assumed to be rectangular, and each wall is identified by a cardinal direction. North, south, west and east for the top, bottom, left and right wall, respectively. When interpreting the bar location coordinates, the southwest corner of the room is taken as the origin (0, 0).

Be aware that the room layout's origin may differ from the one in the sound location data stream, which uses Bar 0's location as the origin. Bar 0 will always be situated on the south wall of the room layout. Therefore, to convert a sound location coordinate (x, y) to a room coordinate, use (x + x', y), where x' is the x value of Bar 0's location in the response's bars array. Each index in the bars array corresponds to that bar's order in the system, so Bar 0 is the first element and Bar 1 is the second element.

Note that room dimensions are configurable by end-users through the Nureva Console software, whereas bar locations are detected through the calibration process during hardware installation. The bar locations are not guaranteed to align precisely with a wall as a result.

```
Request
{
   "request": "v1/room/layout",
   "requestId": string,
   "clientId": string
}

Response
{
   "request": "v1/room/layout",
   "requestId": string,
   "clientId": string,
   "body": {
      "roomDimensions": {
         "x": number,
         "y": number
      },
      "bars": {
         "wall": "North" | "South" | "West" | "East",
         "location": {
            "x": number,
            "y": number
         }
      }[]
   }
}
```

roomDimensions: The dimensions of the room in millimeters. The x value represents the west-east dimension and the y value represents the north-south dimension.

bars: An ordered array where each element is the location for a single bar in the system. Each index corresponds to that bar's order in the system (e.g., first element is Bar 0, second element is Bar 1).

- **wall:** The wall the bar is located on. Not to be confused with the direction the bar is facing. For example, if it is located on the south wall, then it is facing north; if it is on the west wall, then it is facing east.
- **location:** The x, y coordinates of the bar's location in millimeters. The southwest corner of the room is taken as the origin (0, 0) of the coordinate system, where positive y values are in the north direction and positive x values are in the east direction.

Errors

- Status Code 10000 – Bad request
  - Request does not conform to the specification. The message field will include detailed information about which part of the request is incorrect.
- Status Code 11000 – Unsupported device
  - Connected hardware device does not support audio location streaming or does not support room layout capabilities
- Status Code 11001 – Device not connected
  - No audio device is connected
- Status Code 11006 – Room info unavailable
- Status Code 11007 – Bar locations unavailable

Sample requests

1. Valid request/response

Request

```
{
   "request": "v1/room/layout",
   "requestId": "1234",
   "clientId": "1111"
}
```

Response

```
{
   "request": "v1/room/layout",
   "requestId": "1234",
   "clientId": "1111",
   "body": {
      "roomDimensions": {
         "x": 16764,
         "y": 10668
      },
      "bars": [
         {
            "wall": "South",
            "location": {
               "x": 8382,
               "y": 0
            }
         },
         {
            "wall": "North",
            "location": {
               "x": 11382,
               "y": 10002
            }
         }
      ]
   }
```

```
}
```

2. Unsupported device – note this is a valid request format, but the device doesn't support room layout capabilities, so it returns an error

```
Request
{
   "request": "v1/room/layout",
   "requestId": "1234",
   "clientId": "1111"
}

Response
{
   "request": "v1/room/layout",
   "requestId": "1234",
   "clientId": "1111",
   "errors": [
      {
         "statusCode": 11000,
         "message": "Unsupported device"
      }
   ]
}
```

Status codes

Code	Description
3000	Socket closed: Unauthorized access
4050	Socket closed: Maximum number of connected clients reached
4051	Socket closed: Requesting IP is not in valid range
10000	Bad request (e.g., invalid request format, invalid parameters, etc.)
10001	Unexpected error preventing request fulfillment
11000	Unsupported device
11001	Device not connected
11002	Interrupted by firmware update
11003	Audio location unavailable
11004	Microphone is muted
11005	Speaker bar is disconnected
11006	Room info unavailable
11007	Bar locations unavailable