

# HDANYWHERE

Seamless **Entertainment** Discreet **Technology**

# API Changelog

Version	Notes
1.00	First Release.
1.01 (05/04/18)	Addition of Device Type to response /api/data/201/[x]
1.02 (13/04/18)	Amendment of API glossary with revised descriptions of video input, video output, audio input & audio output. Addition of identify API.
1.03 (18/04/18)	Addition of first boot status to /api/data/100
1.04 (23/04/18)	Addition of Zone switching, volume and mute API's
1.05 (30/04/18)	Addition of example responses added, addition data added to responses for /data/100 & /data/101
1.06 (24/07/18)	IR port ID and addressing table for MHUB systems pg. 40
1.07 (01/11/18)	Updated Supported systems table with hostname entry and official name
1.08 (11/12/18)	Added IR port assignment in /data/100 & 101 and added JSON to /irpass/ as body requirement
1.09 (30/01/19)	Changed supported IR data formats, added data/203
1.10 (30/05/19)	ARC, CEC APIs added
1.11 (18/07/19)	Updated responses, added boolean type.
1.12 (24/02/20)	Updated data/100, data/200, data/200/x/, data/203, data/203/x/. Added data/204, control/fixaudio/ and command/rs232pass
1.13 (17/11/20)	Added I/O reference for MHUB PRO 2.0 systems
1.14 (13/05/21)	Added execution time to sequence data
1.15 (04/08/21)	Added data/204, data/205 and data/205/[x]
1.16 (01/11/21)	Added api/control/output, updated data/200, data/200/[z], data/203 and data/203/[z], expanded port mapping to include CEC and RS232 information.
1.17(18/01/22)	Added api/control/function, updated api/control/sequence and api/data/202 to include function information
1.18 (05/07/22)	added api/control/volume/step api
1.19 (05/10/23)	Added support for 44100A, exanded CEC api's to support input control

# API Prologue

---

Protocol(s)	REST/HTTP/OAuth 2.0
Output	XML/JSON
API Language	English (UK)
Current API Version	2.1 (Contact HDA for older versions)
OS Version	8.20+(MOS), 10.01+(uOS)
OS FW Version	2.0+
Scope	HDA Cloud (Internet) & MHUB/ZP (LAN)

## Overview

The HDANYWHERE (HDA) API has been developed to encourage developers to write their own applications for HDA systems. The API seeks to address the following objectives:

1. To reduce the amount of time required to write control and monitoring applications
2. To create a universal API spanning the entire HDA product range (see supported device list)
3. To increase interoperability between HDA systems
4. To grant third-party developers access to advanced IO feature-set

<b>API Changelog</b>	<b>2</b>
<b>API Prologue</b>	<b>3</b>
Overview	3
<b>API Basics</b>	<b>6</b>
Built using REST	6
Common characteristics	6
HTTP verbs	6
Naming convention	7
Finding MHUB in your network	8
Supported Systems	9
Working with multiple MHUB systems	10
<b>How MHUB Works</b>	<b>11</b>
The basic principles of a matrix	11
Inputs and outputs	11
Infrared (IR) ports	12
Stacking MHUB	13
Inputs and outputs on hybrid systems	14
Zones	15
Control	16
Grouping	16
<b>API Resources</b>	<b>17</b>
API breakdown	17
API response format / request structure	18
Quick steps for Standalone connection to any MHUB system.	19
Critical	20
/api/reboot/1/ Full reboot / Power Cycle	20
/api/reboot/2/ Reboot MHUB-QS	20
/api/power/0/ Standby ON	21
/api/power/1/ Standby OFF (Turn On)	21
/api/identify/ Identify	22
Data	23
/api/data/0/ MHUB Power State	23
/api/data/100/ System Information: Standalone	24
/api/data/101/ System Information: Stacked Mode	26
/api/data/102/ MHUB Zones	27
/api/data/103/ MHUB Groups	28
/api/data/200/ Status - Single system	29
/api/data/200/[zid] MHUB Zone Status - Single MHUB	30
/api/data/201/ MHUB uControl Pack summary - Single	31
/api/data/201/[x] - Single uControl	32
/api/data/202/ Sequences and Functions	33
/api/data/203/ MHUB Status - Stacked MHUB	34
/api/data/203/[zid] MHUB Zone Status - Stacked MHUB	35
/api/data/204/ uControl	36
/api/data/205/ MHUB uControl Pack summary - Stack	37

/api/data/205/[x] - Stack	38
uControl	39
Operation	39
/api/control/switch/	
Switching	39
/api/control/switch/zone/	
Zone Switching (MHUB AUDIO only)	40
/api/control/fixaudio/	
Source audio extraction	41
/api/control/volume/	
Set Output Volume	42
/api/control/volume/zone/	
Set Zone Volume (MHUB AUDIO only)	43
/api/control/arc/	
Audio Return Channel (ARC)	44
/api/control/mute/	
Mute	45
/api/control/mute/zone/	
Mute Zone (MHUB AUDIO only)	46
/api/control/output/	
Shutdown Amplifier (MHUB AUDIO only)	47
/api/control/group/create/	
Creating a Group	48
/api/control/group/delete/	
Deleting a Group	49
/api/control/group/[add/delete]/	
Adding or removing a Zone from Group	50
/api/control/group/volume/set/	
Changing the volume in a Group	51
/api/control/mutegroup/	
Muting the audio in a Group	52
/api/control/sequence/	
Execute Sequence	53
/api/control/function/	
Execute Sequence	54
IO	55
/api/command/ir/	
Execute uControl Command (IR)	55
/api/command/irpass/	
IR Passthrough	56
/api/command/cec/	
Execute uControl Command (CEC)	57
/api/command/cecpass/	
CEC Passthrough	58
/api/command/rs232config/	
RS232 port configuration	59
/api/command/rs232pass/	
RS232 Passthrough	60
<b>UDP State Output</b>	<b>61</b>
<b>HDA Cloud References</b>	<b>62</b>
<b>Reference Material</b>	<b>63</b>
DNS-SD responses	64
Port mapping	65
API glossary	66
uControl command IDs	69

# API Basics

---

## Built using REST

The HDA API is built using REST principles and defines a set of functions which developers can use to control MHUB (or any HDA device), access its IO, or report back data. This interaction is performed via the HTTP protocol. Using HTTP and REST means that applications can be made for using any programming language.

## Common characteristics

- You access functions or resources by sending a HTTP request to the MHUB API server. The server replies with a response that contains either the data you requested, or the status indicator, or even both.
- All functions or resources are located from the base URL at <http://devicehost/api/>
- Any call to the API server will return standard [HTTP response codes](#).

## HTTP verbs

HTTP verbs are used to manage the state of resources, these are: GET and POST. PUT, and DELETE are not supported.

## Naming convention

All interaction between your application and a HDA device is conducted in `lowercase`. Ensure that you do not confuse physical port labelling on the device which appears in uppercase (A, B, C etc) with their programmatic addressable equivalents. For example, to send an API command to MHUB PRO (8x8) output port **labelled "F"** you would instruct your application to replace the cap "F" with a lowercase "f".

Type	Standard (if applicable)	Example
ID	lowerCamelCase	inputAudio1
Labels/Array names	lowercase_seperated_with_underscore	start_id
Boolean	n/a	<b>boolean</b>
MHUB inputs	n/a	1,2,3,4...
MHUB outputs	n/a	a,b,c,d...
Arrays	n/a	<b>bold</b>

## Finding MHUB in your network

MHUB systems use multicast DNS (mDNS) to identify themselves on a network to a client. You can find the full mDNS protocol here [RFC 6762](#).

HDANYWHERE products are announced under 2 services. Depending on the year of release it is either “\_http.\_tcp” or “\_hda.\_tcp”. This is described in detail in the section [Supported Systems](#).

To find MHUB on the network you will need to use DNS Service Discovery (DNS-SD) as supported by your operating system. The client will need to make a DNS-SD query to (244.0.0.251) and await a response. HDA recommends that you use the appropriate API on the client device to achieve this. For example, if you are using iOS then you can use the Bonjour API library. If you are using Windows or Linux then the Avahi API is applicable.

Please refer to the Reference Material section for a full list of DNS-SD responses.

### Example DNS-SD query / response

If you are searching for MHUB PRO (4x4) 70 (2016) and your MHUB is on the latest version of MHUB-OS and Firmware then your response will match the following:

```
Hostname = [MHUB4K44PRO.local]
Address = [IP Address]
Port = [80]
Txt = ["MHUB PRO (4x4) 70"]
```



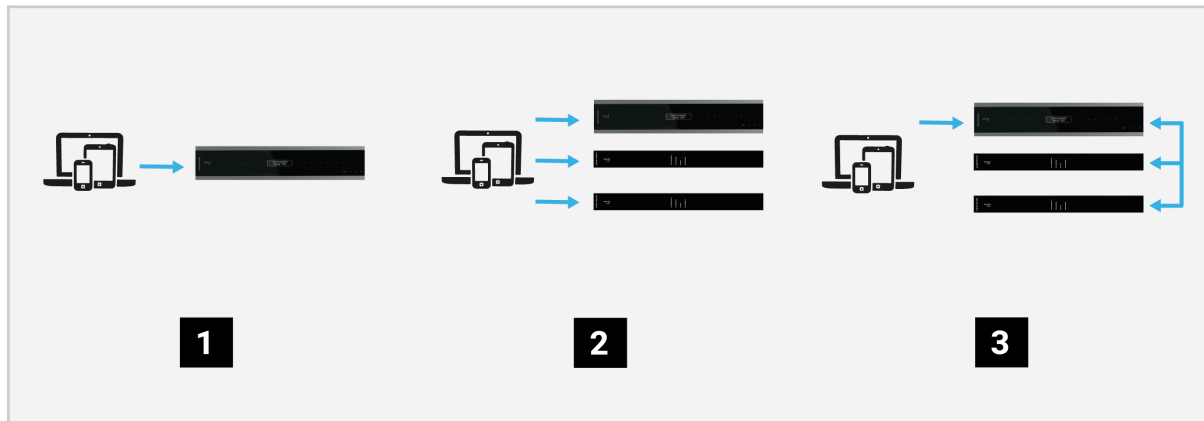
## Supported Systems

Official Name	Service	API Version	Device Type	Device ID	Release	SKU	Hostname
<b>MHUB U</b>							
MHUB (4x3+1)	_http._tcp	2.0	Video Matrix	V1	2018	1.05.904.012.1	MHUB431U
MHUB (8x6+2)	_http._tcp	2.0	Video Matrix	V1	2018	1.05.904.013.1	MHUB862U
MHUB U (4x1+1)	_hda._tcp	2.1	Hybrid Matrix	H1	2020	1.05.904.023.1	MHUBU41140
MHUB U (4x3+1)	_hda._tcp	2.1	Video Matrix	V1	2019	1.05.904.012.2	MHUBU43140
MHUB U (8x6+2)	_hda._tcp	2.1	Video Matrix	V1	2019	1.05.904.013.2	MHUBU86240
<b>MHUB PRO</b>							
MHUB PRO (4x4) 40	_http._tcp	2.0	Video Matrix	V1	2017	1.05.904.014.1	MHUBPRO4440
MHUB PRO (4x4) 70	_http._tcp	2.0	Video Matrix	V1	2016	1.05.904.008.2	MHUB4K44PRO
MHUB PRO (8x8) 40	_http._tcp	2.0	Video Matrix	V1	2017	1.05.904.015.1	MHUBPRO8840
MHUB PRO (8x8) 70	_http._tcp	2.0	Video Matrix	V1	2016	1.05.904.009.2	MHUB4K88PRO
<b>MHUB PRO 2.0</b>							
MHUB PRO 2.0 (4x4) 40	_hda._tcp	2.1	Hybrid Matrix	H1	2019	1.05.904.021.1	MHUBPRO24440
MHUB PRO 2.0 (8x8) 100	_hda._tcp	2.1	Hybrid Matrix	H1	2019	1.05.904.022.1	MHUBPRO288100
<b>MHUB S</b>							
MHUB S (8+8x8) 100	_hda._tcp	2.1	Video Matrix	V1	2021	1.05.904.025.1	MHUBS888100
MHUB S (8+8x8) 100 A	_hda._tcp	2.1	Video Matrix	HV1	2024	1.05.904.025.1	MHUBS888100A
<b>MHUB MAX</b>							
MHUB MAX (4x4)	_http._tcp	2.0	Video Matrix	V1	2017	1.05.904.011.1	MHUBMAX44
<b>MHUB AUDIO</b>							
MHUB AUDIO (6x4)	_http._tcp	2.0	Audio Matrix	A1	2018	1.05.912.001.1	MHUBAUDIO64
MULTI ZONE AMP (6x4) 55	_hda._tcp	2.1	Audio Matrix	A1	2023	1.05.912.002.1	MZA6455
<b>ZONE PROCESSOR</b>							
UCONTROL ZONE PROCESSOR (5)	_hda._tcp	2.1	Control Processor	P1	2020	1.80.904.002.1	ZP5
UCONTROL ZONE PROCESSOR (1)	_hda._tcp	2.1	Control Processor	P1	2020	1.80.904.001.1	ZP1
<b>MHUB</b>							
MHUB (4x4) 100 A	_hda._tcp	2.1	Hybrid Matrix	HV1	2023	1.5.904.026.1	MHUB44100A

Table: Supported Systems

## Working with multiple MHUB systems

MHUB systems can be stacked up to a maximum of 4 times. Stacks can be created using only Audio Matrix devices (See Table: *Supported MHUB Systems*) or a combination of Audio Matrices and a single Video Matrix. Stacking of multiple Video Matrix systems is not supported using method #3 below however it can be achieved if you opt to communicate with MHUB using method #2.



### 1. Standalone

Your control application communicates directly with a single MHUB system.

### 2. Manual

Using the same principles as method #1 each MHUB is treated as a separate device. Some MHUB features such as Sequences and uControl (which takes advantage of both Video and Audio operations) will not work as inter-communication between MHUB devices is not supported. With this method, your control application will need to manage routing logic to ensure that video or audio or a combination of both is kept in sync throughout the system.

### 3. MHUB-OS/uOS Stack

This method requires prior configuration using HDA's uControl app before it can be enabled. Stack method will take I/O data from all MHUBs in the stacked arrangement and report back to your application as a single large Audio Matrix or Audio/Video Matrix system. In Stack each MHUB will be assigned with a rank (Master or Slave) and your application will communicate only with the Master device. MHUB-OS/uOS will function as if it was one large system meaning that uControl, Sequences, Voice control, and HDA Cloud operations are supported throughout the entire system scope.

# How MHUB Works

## The basic principles of a matrix

Video and Audio Matrices are capable of routing various types of digital or analogue AV inputs (Apple TV, XBOX One, Sky, TiVo, SONOS etc) to any compatible output (like a TV, projector or speaker) or to multiple outputs simultaneously. HDA is a manufacturer of these devices which it calls MHUB.

## Inputs and outputs

All MHUB systems share a common characteristic of having a collection of input ports (referenced using numbers: 1, 2, 3, 4 etc) and a collection of output ports or end-points (referenced using letters: A, B, C, D etc). To illustrate this; the MHUB PRO (4x4) has 4 inputs and 4 outputs and the MHUB (8x6+2) has 8 inputs and 8 outputs in total "6+2". Signals from an Input can be routed to any Output or combination of Outputs and this switch and split ability is what is defined as a matrix.

### Standalone I/O ID assignment

MHUB PRO (4X4)



Input ID: (1,2,3,4)

Output ID: (A,B,C,D)

## Infrared (IR) ports

MHUB systems with IR ports can be used to transmit IR commands directly by sending Pronto IR data to the addressable port ID on MHUB.

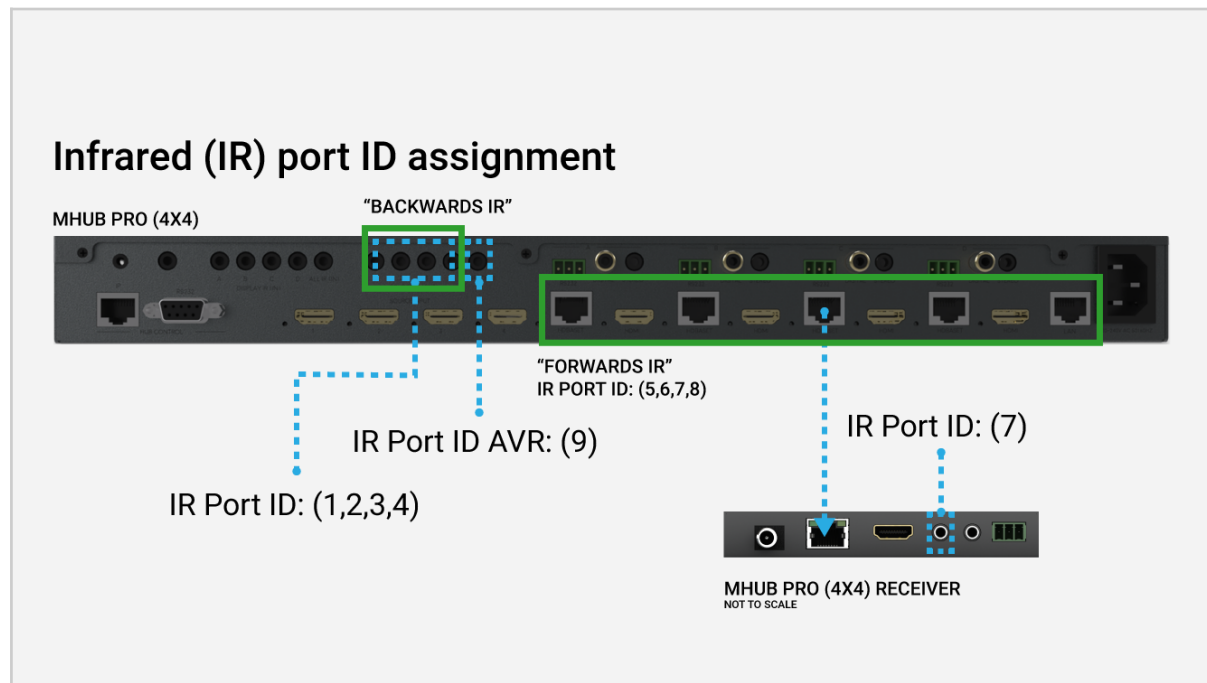
### Important:

The ID format for IR ports do not follow the same assignment rules for video or audio input/output ports. Instead, IR ports are all referenced using numbers which start from number 1 and increment depending on the number of IR ports declared by MHUB.

MHUB will distinguish within the API whether a collection of IR ports are located on the main chassis of MHUB (Backwards) or if they are found on one of its display receivers (Forwards). If you are using a HDA Zone Processor then this will be identified as a (Remote) IR port.

For example, the MHUB PRO (4x4) in the diagram below has 4 input ports (referenced using numbers: 1, 2, 3, 4 etc) and 4 output ports (referenced using letters: A, B, C, D etc). If you wanted to send an IR command to the output port labelled "D" then you would send it to IR Port ID = 8.

Additionally, AVR ports are declared as separate entities but follow the same numbering rule as above. The AVR port ID will always be ++1 after the final forwards ID port. In the diagram below you will see that the AVR port is marked ID = 9.



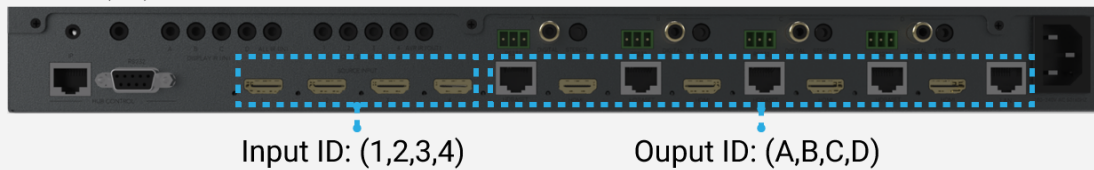
## Stacking MHUB

It is possible to stack more than one MHUB system together. If you enable MHUB's stacked mode then MHUB-OS will calculate the total number of inputs and outputs and assign addressable references to each of them. For example, if you stacked an MHUB PRO (4x4) with an MHUB AUDIO (6x4) then you would be able to reference each port as one large MHUB system as follows:

- **Inputs = 10**  
MHUB PRO (4x4) - Referenced: (1,2,3,4)  
MHUB AUDIO (6x4) - Referenced: (5,6,7,8,9,10)
- **Outputs = 8**  
MHUB PRO (4x4) - Referenced: a,b,c,d  
MHUB AUDIO (6x4) - Referenced: e,f,g,h

### Stacked I/O ID assignment

MHUB PRO (4X4)



MHUB AUDIO (6X4)

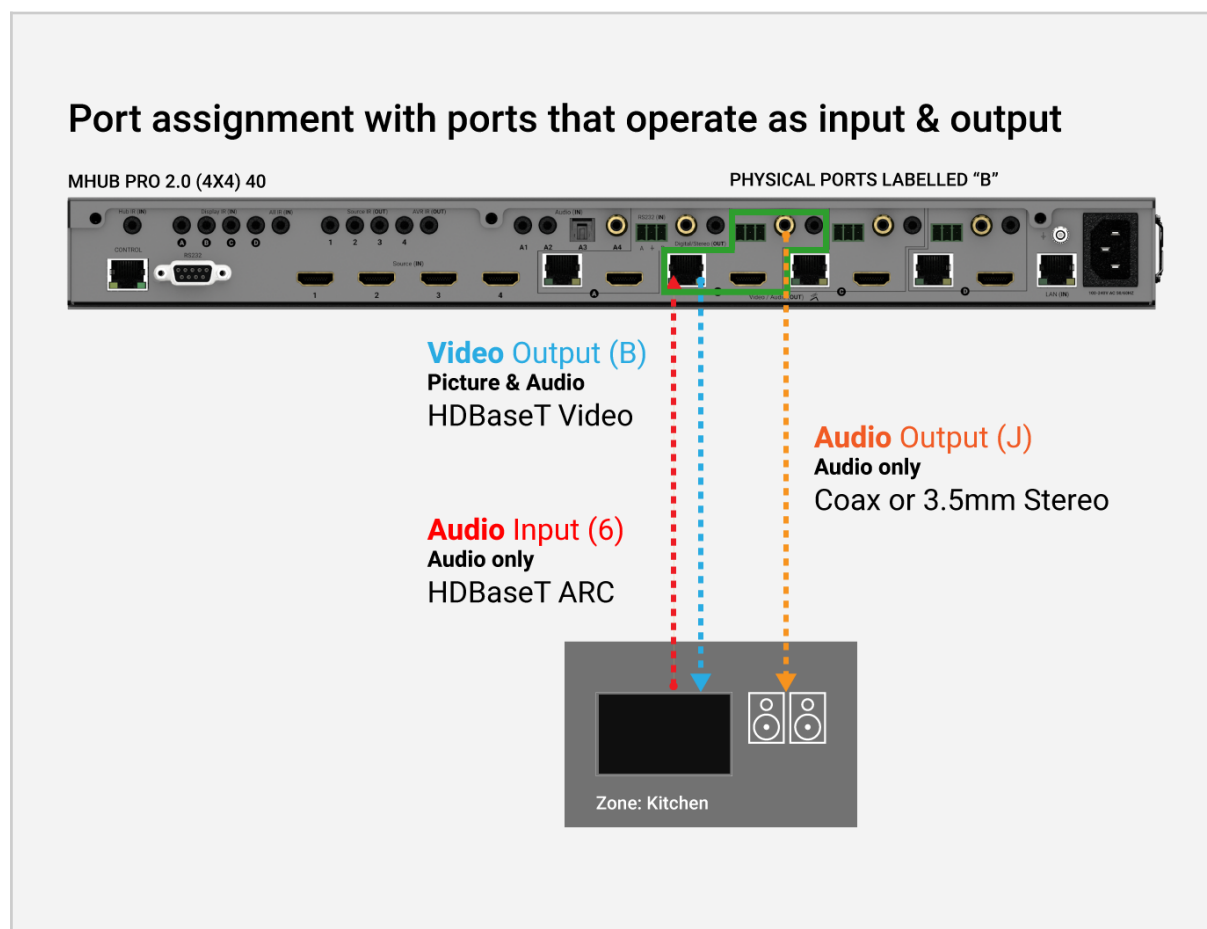


## Inputs and outputs on hybrid systems

Hybrid MHUB systems like MHUB PRO 2.0 contain features like Audio Return Channel (ARC) which allows a video output (video: picture + audio) to act as an audio input also. Referring to physical port labelling on the device itself will not assist you in helping identify the port ID.

Instead, port assignment and function follows the same nomenclature and method outlined in section "Stacking MHUB". Consequently, there will be more IDs reported in API feedback than there are physical ports on the device itself.

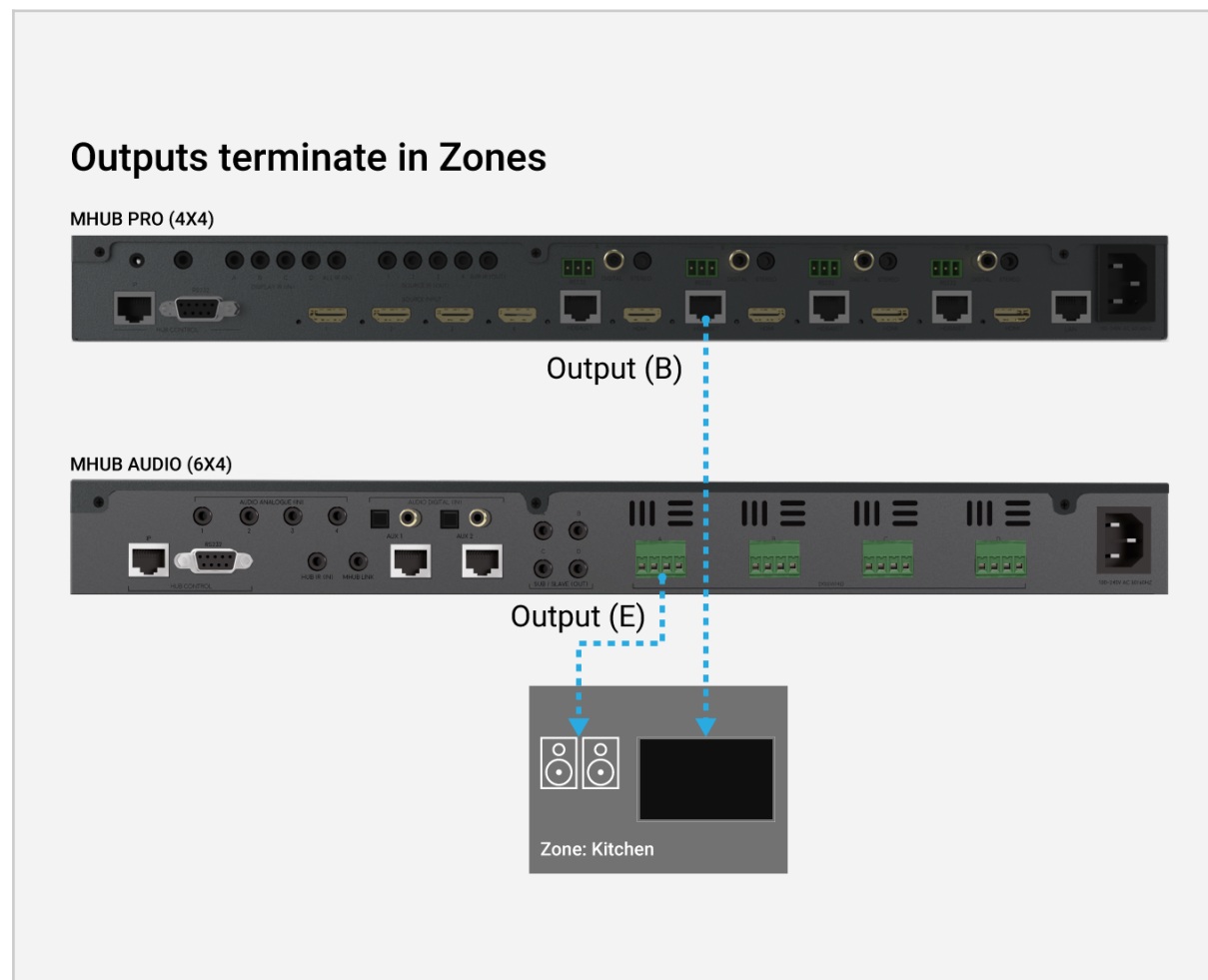
To illustrate this, in the diagram below, there is a MHUB PRO 2.0 (4x4) 40 system setup with a zone called "Kitchen". Inside that zone, there are outputs or end-points which remain referenced with letters (B). Assigning output B to Kitchen will gain command over the display and content routed to it. However, if that display is also ARC enabled, then MHUB can pull the audio from the display. This is treated as an input, so is indexed by numbers and is specifically referenced by ID (6). In this scenario it is possible to route, for example, a blu-ray (input 1) to output B - this will show the blu-ray on the kitchen display. If that display is ARC enabled then a further instruction can be sent to MHUB, this time to pull the audio as an input into MHUB and routed to the kitchen speakers, output J. From a user point of view all these instructions are happening in the kitchen (output B) but programmatically, each input or output needs to be uniquely identifiable.



## Zones

Zones are virtual constructs which define a physical space for MHUB end-points to terminate.

A Zone can include a maximum of 1 video matrix output and a maximum of 4 audio matrix outputs. For example, you can create a zone called “Kitchen” and assign 1 video and 1 audio output to it. A Zone must contain at least 1 output from an MHUB for it to be addressable for control. Note: outputs can not be shared or split to more than 1 Zone.



Zone quantity is defined by calculating the sum-total of all referenceable outputs on an MHUB system, standalone or stacked. Using the MHUB PRO (4x4) and MHUB AUDIO (6x4) example would indicate that that system can support a maximum of 8 zones.

## Control

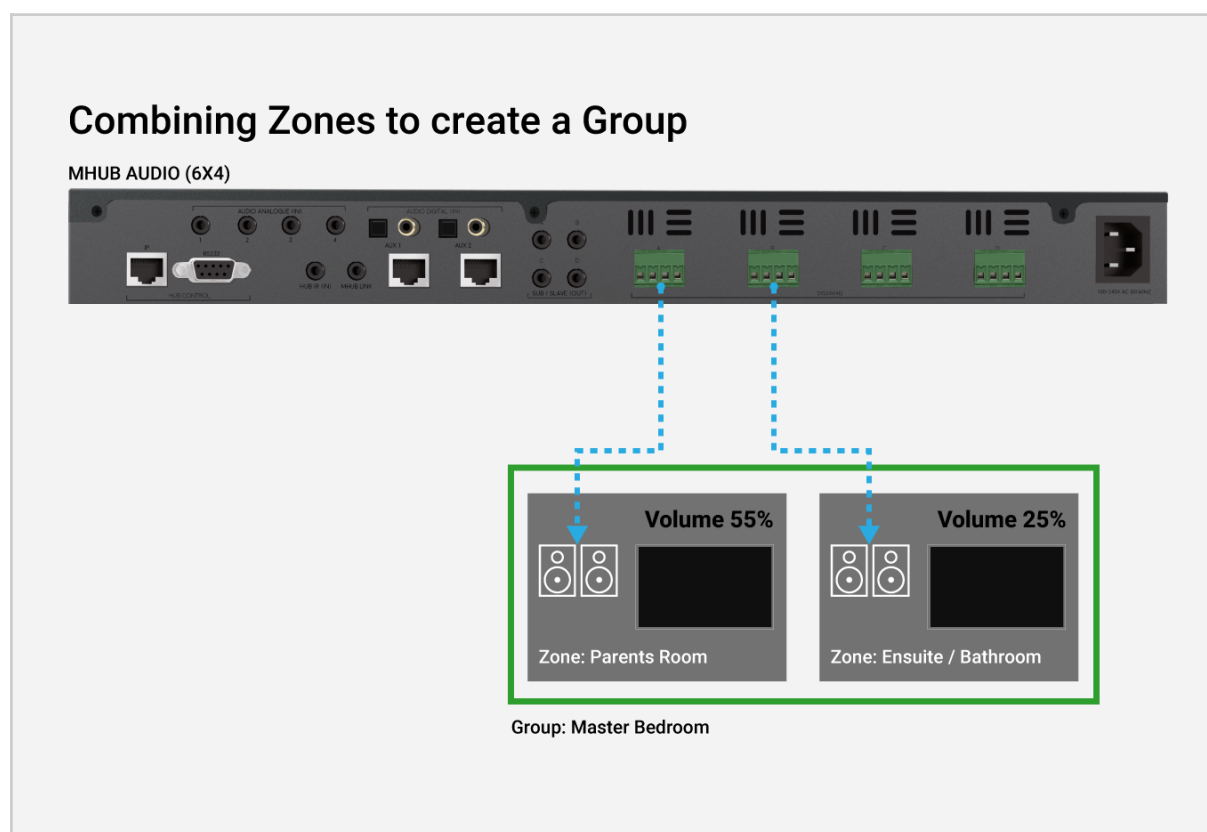
MHUB system control is achieved by giving your application a *Zone focus*. For most control API commands you will need the end user to identify which Zone they want to control before MHUB can respond correctly.

## Grouping

Zones which contain outputs from MHUB AUDIO systems can be grouped together allowing for audio adjustment over one or more Zones. Operations such as volume adjustment and muting can be applied by sending a single group API rather than targeting individual Zones.

Groups can support a maximum of 4 Zones. Grouped volume is reported by taking the average volume value in each Zone and adjusting this value will change the volume state uniformly in each Zone. Using the example in *Combining Zones to create a Group* you can see that Zones "Parents Room" and "Ensuite / Bathroom" are currently reporting volume states of 55% and 25% respectively. MHUB will average this value and report to your application that the Group volume is 40%. Adjusting the Group volume by incrementing the value +10 will set the Group volume at 50% and the Zones "Parents Room" and "Ensuite / Bathroom" become 65% and 35%.

If a Zone belongs to a Group it is still possible to change the individual Zone audio state. To do this you would instruct your application to target the Zone directly via the appropriate API.





# API Resources

---

## API breakdown

Control and reporting functions can be arranged into the following categories:

- **Critical**  
Universally supported control functions for HDA systems to perform basic critical operations such as turning devices on/off or rebooting video/audio/OS processors.
- **Data**  
A collection of resources designed to feedback the operational state of devices or to obtain configurational data so that you may control it.
- **Operation**  
General control functions and resources for everyday use and operation. These include operations like switching inputs, adjusting volume or executing Sequences.
- **IO**  
Use these resources to create advanced applications using device IO to control 3rd party hardware are connected to a HDA system.

## API response format / request structure

All functions or resources are located from the base URL at: <http://devicehost/api/>. Every API call will respond with a standardised header which details the current API version and whether any error was logged during execution.

### API response format

All APIs will always start with a header:

```
"header": {  
  "version": "2.1"  
}
```

Followed by:

```
"data": {  
  [API RESPONSE]  
}
```

### Example - to query if MHUB is on or off:

Some attributes within this document are highlighted in **bold**. Their definitions can be found in the Reference Material section at the end of this document. Data arrays are indicated in **bold**. Boolean responses are in **Green**.

```
{  
  "header": {  
    "version": "2.1"  
  },  
  "data": {  
    "power": true  
  }  
}
```

### Example - Error response

```
{  
  "header": {  
    "version": "2.1"  
  },  
  "error": {  
    "code": "1"  
  }  
}
```

## Quick steps for Standalone connection to any MHUB system.

The following example shows which API's need to be called to configure a standalone device. We are going to assume the first boot process has been completed. For this example we will use a MHUB PRO 2.0 (4x4) 40.

- A DNS scan is run on the network. The following MHUB hardware is discovered:

- Hostname = [MHUBPRO24440.local]
- Address = [192.168.1.1]
- Port = [80]
- Txt = ["HDANYWHERE MHUB PRO 2.0 (4x4) 40"]

- Then request */api/data/100*.

When called this API will return basic system information regarding the MHUB and will detail what input and output ports are available on the system along with their respective port IDs and user defined labels. This response will also inform you if the MHUB has been setup in a stack, or has an accessory device attached to it or is operating in standalone mode.

- You now need to ascertain what port IDs appear where. To do this you need to request */api/data/102*

This API will respond with all outputs or inputs that terminate or start in a zone along with a user defined zone label.

- Finally, to request the switch or operational state of the MHUB you would request */api/data/200*

This will provide state information for all zones so that your interface is populated with up-to-date data.

Your application will now hold data for all input and output quantities, their addressable IDs, where they terminate or start, their user defined labels and an up-to-date state value so that your interface can be drawn accurately to represent what is going on inside MHUB.

## Critical

/api/reboot/1/

Full reboot / Power Cycle

### **Performs a full reboot / power cycle of all hardware components within MHUB**

This request has a fixed execution time of 30,000ms (30 seconds) to ensure that communication with MHUB is available after reboot is complete. The reboot command is processed as soon as it is received and 30 seconds later the system is automatically brought back online.

**GET** request:

<http://devicehost/api/reboot/1/>

Response

```
"data": {  
  "Description": "Performing full reboot."  
}
```

/api/reboot/2/

Reboot OS

### **Performs a software reboot of OS but maintaining video/audio function active**

**GET** request:

<http://devicehost/api/reboot/2/>

Response

```
"data": {  
  "Description": "Rebooting MOS."  
}
```

## /api/power/0/ Standby ON (Turn Off)

### Put MHUB into a low power (standby) state - turn off

This command is not universally supported on every system. Please refer to the API Appendix for system support.

**GET** request:

<http://devicehost/api/power/0/>

Response

```
"data": {  
  "power": "Powering off MHUB."  
}
```

## /api/power/1/ Standby OFF (Turn On)

### Take MHUB out of its low power (standby) state - turn on

**GET** request:

<http://devicehost/api/power/1/>

Response

```
"data": {  
  "power": "Powering on MHUB."  
}
```

## /api/identify/ Identify

**Sets Power/front panel LEDs flashing to devices can be identified.**

MHUB AUDIO - HDANYWHERE and AUDIO logo flashing so MHUB can be identified

MZMA - HDANYWHERE logo Flashes

ZP5/ZP1/411/MHUB S - Power LED flashes

**GET** request:

<http://devicehost/api/identify/>

Response

```
"data": {  
    "identify": true  
}
```

## Data

/api/data/0/

MHUB Power State

**Find out if MHUB is on or off**

**GET** request:

<http://devicehost/api/data/0/>

true = MHUB is on

false = MHUB is off

Response

```
"data": {  
  "Power": boolean  
}
```

/api/data/100/

## System Information: Standalone

### Request input, output IDs and labels for all I/O ports in a standalone system

Returns basic system information including identifiable data, serial number, IO connectivity, software versions and stack status.

**GET** request:

<http://devicehost/api/data/100/>

Response:

```
{
  "data": {
    "io_data": {
      "input_video": "video input",
      "output_video": "video output",
      "output_video_mirror": "video output mirror",
      "input_audio": "audio input",
      "output_audio": "audio output",
      "output_audio_mirror": "audio output mirror"},
    "ir": {
      "backwards": {
        "draw": "boolean",
        "start_id": "start id",
        "ports": "ports"},
      "forwards": {
        "draw": "boolean",
        "start_id": "start id",
        "Ports": "ports"},
      "avr": {
        "draw": "boolean",
        "start_id": "start id",
        "Ports": "ports"}},
    "cec": {
      "output": [{
        "type": "type",
        "draw": "boolean",
        "start_id": "start id",
        "ports": "ports"
      }],
      {
        "type": "type",
        "draw": "boolean",
        "start_id": "start id",
        "ports": "ports"}],
      "input": {}
    },
    "rs232": {
      "output": [{
        "type": "type",
        "draw": "boolean",
```



```
        "start_id": "start id",
        "ports": "ports"
    },
    {
        "type": "type",
        "draw": boolean,
        "start_id": "start id",
        "ports": "ports"
    }
}],
"ws": boolean,
    "mhub": {
        "first_boot": boolean,
        "ip_address": "ip address",
        "mhub_firmware": mhub firmware,
        "mhub-os_firmware": mhub-os firmware,
        "mhub-os_version": mhub-os version,
        "api": api,
        "mhub_official_name": "mhub official name",
        "unit_id": "unit id",
        "mhub_name": "mhub name",
        "serial_number": "serial number"
    },
    "stack": {
        "stack_status": boolean,
        "stack_rank": "stack rank",
        "stack_master": stack master
    }
}
}}}}}
```

/api/data/101/

## System Information: Stacked Mode

### Request input, output IDs and labels for all I/O ports in a stacked system

Pairs physical ports on MHUB systems running in stacked mode with their virtual/addressable IDs. This API is only required when running MHUB in MHUB-OS Stacked mode (see page 7).

**GET** request:

<http://devicehost/api/data/101/>

Response:

```
"data": {
  "stacked_io": {
    "stack_input_video": stack video input,
    "stack_output_video": stack video output,
    "stack_output_video_mirror": stack video output mirror,
    "stack_input_audio": stack audio input,
    "stack_output_audio": stack audio output,
    "stack_output_audio_mirror": stack audio output mirror
  },
  "mapping": {
    "input": {
      "unit_id": "unit id",
      "input_id": "input id"
    },
    "output": {
      "unit_id": "unit id",
      "output_id": "output id"
    },
    "split_input": {
      "inputs": ["zone id", "zone id"],
      "type": "",
      "shared": ""
    },
    "stack_unit_info": [{
      "unit_id": "unit id",
      "mhub_name": "mhub name",
      "serial_number": "serial number",
      "ip_address": "ip address",
      "stack_rank": "stack rank"
    }]
  }
}
```

## /api/data/102/ MHUB Zones

### Request output and inputs assigned to zones

Returns information about which MHUB outputs or inputs, such as display ARC, are assigned to each zone.

**GET** request:

<http://devicehost/api/data/102/>

Response:

```
{
  "data": [{
    "zone_id": "zone id",
    "zone_label": "zone label",
    "outputs": [{
      "unit_id": "unit id",
      "output_id": "output id"
      "arc_input": "arc input id"
    }],
    "auto_switch_mode": boolean
  }]
}
```

## /api/data/103/ MHUB Groups

### Request group information

Returns information about which zones are assigned to each group, volume and mute state.

**GET** request:

<http://devicehost/api/data/103/>

True = mute is on  
False = mute is off

Response:

```
{
  "groups": [{
    "group_id": "group id",
    "group_label": "group label",
    "zones": ["zone id", "zone id"],
    "group_volume": "mhub audio group volume",
    "group_mute": boolean
  }]
}
```

/api/data/200/

Status - Single system

### Requests System state (standalone).

Request current routing, volume and audio states for all zones.

**GET** request:

<http://devicehost/api/data/200/>

true = mute is on

false = mute is off

Response:

```
{
  "data": {
    "zones": [{
      "zone_id": "zone id",
      "state": [{
        "output_id": "output id",
        "input_id": "input id",
        "audio_id": "audio id",
        "disp_audio_id": "display audio input id",
        "volume": "zone audio volume",
        "mute": boolean,
        "arc": boolean,
        "avr": "port id",
        "global": "port id",
      }]
    }]
  }
}
```

/api/data/200/[zid]

## MHUB Zone Status - Single MHUB

### Requests MHUB Zone state (standalone).

Request current routing information for all zones.

*(Please note that volume and mute information will relate to MHUB AUDIO levels only)*

**GET** request:

[http://devicehost/api/data/200/\[zid\]](http://devicehost/api/data/200/[zid])

True = mute is enabled

False = mute is disabled

Arguments:

[zid] = zone id (z1, z2, z3...)

Response:

```
"data": {  
  "zone": {  
    "zone_id": "zone id",  
    "video_input": "input id",  
    "audio_input": "audio input id",  
    "disp_audio_id": "display audio input id"  
    "volume": zone audio volume,  
    "mute": boolean,  
    "arc": boolean,  
    "display_power": "display power",  
  }  
}
```

## /api/data/201/ MHUB uControl Pack summary - Single

### MHUB ports with uControl Packs installed

Request which MHUB IR ports have uControl device control packs installed.

**GET** request:

<http://devicehost/api/data/201/>

Response:

```
{ "data": {  
    "unit_id": "V1",  
    "input": [{  
        "id": "1",  
        "irpack": boolean  
    },  
    {  
        "id": "2",  
        "irpack": boolean  
    },  
    {  
        "id": "3",  
        "irpack": boolean  
    },  
    {  
        "id": "4",  
        "irpack": boolean  
    }  
  ],  
    "output": [{  
        "id": "a",  
        "irpack": boolean  
    },  
    {  
        "id": "b",  
        "irpack": boolean  
    },  
    {  
        "id": "c",  
        "irpack": boolean  
    },  
    {  
        "id": "d",  
        "irpack": boolean  
    }  
  ],  
    "avr": boolean,  
    "global": [{  
        "id": "5",  
        "irpack": boolean  
    }  
  ]  
}
```

## /api/data/201/[x] - Single uControl

### Request uControl button ID and labels

Returns extended information on a specified uControl device control pack.

**GET** request:

[http://devicehost/api/data/201/\[x\]](http://devicehost/api/data/201/[x])

Arguments:

[x] = IR port

Response:

```
"data": {
  "name": "ucontrol pack name",
  "type": "ucontrol device type",
  "version": "ucontrol pack version",
  "irpack_id": "ucontrol pack id",
  "ucontrol_pack_method": "ucontrol pack method",
  "ir_pack": [{
    "command_id": "command id",
    "label": "label"
  }]
}
```



## /api/data/202/ Sequences and Functions

### Request Sequence & Function data

Returns basic information on sequences currently stored in the MHUB.

**GET** request:

<http://devicehost/api/data/202/>

Response:

```
"data": {  
  "sequences_functions": sequence data,  
                        function data  
}
```

/api/data/203/

## MHUB Status - Stacked MHUB

### Requests MHUB state (stacked).

Request current routing information for all zones in a stacked MHUB setup.

*(Please note that volume and mute information will relate to MHUB AUDIO levels only)*

**GET** request:

<http://devicehost/api/data/203/>

true = mute is on

false = mute is off

Response:

```
{
  "data": {
    "zones": [{
      "zone_id": "zone id",
      "state": [{
        "output_id": "output id",
        "input_id": "input id",
        "volume": "zone audio volume",
        "mute": boolean,
        "arc": boolean,
        "display_power": "display power",
        "output_enabled": boolean
      ]
    }],
  }
}
```

/api/data/203/[zid]

## MHUB Zone Status - Stacked MHUB

### Requests MHUB Zone state (standalone).

Request current routing information for all zones.

*(Please note that volume and mute information will relate to MHUB AUDIO levels only)*

**GET** request:

[http://devicehost/api/data/203/\[zid\]](http://devicehost/api/data/203/[zid])

True = mute is on

False = mute is off

Arguments:

[zid] = zone id (z1, z2, z3...)

Response:

```
"data": {  
  "zone": {  
    "zone_id": "zone id",  
    "video_input": "input id",  
    "audio_input": "audio input id",  
    "paired_audio_input": "paired audio input",  
    "disp_audio_input": "disp audio input",  
    "volume": zone audio volume,  
    "mute": boolean,  
    "arc": boolean,  
    "display_power": "display power",  
    "output_enabled": boolean  
  }  
}
```

## /api/data/204/ uControl CEC Pack

### Request uControl button ID and labels

Returns extended information on the available commands in the CEC pack

**GET** request:

<http://devicehost/api/data/204/>

Response:

```
"data": {
  "version": 1,
  "name": "",
  "type": "",
  "cecpack_id": "",
  "ucontrol_pack_method": "",
  "cecpack": [
    {
      "id": "command id",
      "label": "label"
    }
  ]
}
```

/api/data/205/

## MHUB uControl Pack summary - Stack

### MHUB ports with uControl Packs installed

Request which MHUB IR ports have uControl device control packs installed.

**GET** request:

<http://devicehost/api/data/205/>

Response:

```
{
  "data": [
    {
      "unit_id": "V1",
      "input": [
        {
          "id": "1",
          "irpack": boolean
        },
        {
          "id": "2",
          "irpack": boolean
        },
        {
          "id": "3",
          "irpack": boolean
        },
        {
          "id": "4",
          "irpack": boolean
        }
      ],
      "output": [
        {
          "id": "a",
          "irpack": boolean
        },
        {
          "id": "b",
          "irpack": boolean
        },
        {
          "id": "c",
          "irpack": boolean
        },
        {
          "id": "d",
          "irpack": boolean
        }
      ],
      "avr": [
        {
          "id": "6",
          "irpack": boolean
        }
      ],
      "global": [
        {
          "id": "5",
          "irpack": boolean
        }
      ]
    }
  ]
}
```

## /api/data/205/[x] - Stack uControl

### Request uControl button ID and labels

Returns extended information on a specified uControl device control pack.

**GET** request:

[http://devicehost/api/data/205/\[x\]](http://devicehost/api/data/205/[x])

Arguments:

[x] = IR port

Response:

```
"data": {
  "name": "ucontrol pack name",
  "type": "ucontrol device type",
  "version": "ucontrol pack version",
  "irpack_id": "ucontrol pack id",
  "ucontrol_pack_method": "ucontrol pack method",
  "ir_pack": [{
    "repeat": boolean,
    "command_id": "command id",
    "label": "label"
  }]
}
```

/api/data/206

uControl IP pack information

### Request uControl button ID and labels

Returns extended information on a specified uControl device IP control pack.

**GET** request:

[http://devicehost/api/data/206/\[x\]](http://devicehost/api/data/206/[x])

Arguments:

[x] = IP pack ID

Response:

```
"data": {
  "name": "ucontrol pack name",
  "type": "ucontrol device type",
  "version": "ucontrol pack version",
  "irpack_id": "ucontrol pack id",
  "ucontrol_pack_method": "ucontrol pack method",
  "ir_pack": [{
    "repeat": boolean,
    "command_id": "command id",
    "label": "label"
  }]
}
```

## Operation

/api/control/switch/  
Switching

### Perform source switch

Switch the input source for any output on MHUB.

**GET** request:

[http://devicehost/api/control/switch/\[ox\]/\[iy\]/](http://devicehost/api/control/switch/[ox]/[iy]/)

Arguments:

[ox] = output (*a,b,c.....*)

[iy] = input (*1,2,3.....*)

Response:

```
"data": {  
  "output_id": "output id",  
  "input_id": input id  
}
```



## /api/control/switch/zone/ Zone Switching (MHUB AUDIO only)

### Perform source switch (MHUB AUDIO only)

Switch the audio input source for any zone with MHUB AUDIO outputs. This is possible with MHUB AUDIO only as that device supports the inclusion of multiple outputs in a single zone.

**GET** request:

[http://devicehost/api/control/switch/zone/\[zid\]/\[iy\]/](http://devicehost/api/control/switch/zone/[zid]/[iy]/)

Arguments:

[zid] = zone id (z1, z2, z3.....)

[iy] = input (1,2,3.....)

Response:

```
"data": {  
  "zone_id": "zone id",  
  "input_id": input id  
}
```

## /api/control/fixaudio/ Source audio extraction

### Disables audio matrixing and fixes source audio to predefined audio outputs

Pairs the source audio with the audio outputs. Source 1 output via Audio output A, Source 2 output via Audio output B.

**GET** request:

[http://devicehost/api/control/fixaudio/\[ax\]/](http://devicehost/api/control/fixaudio/[ax]/)

Arguments:

[ax] = true (*turn off audio matrix and fixed audio switch state*)  
false (*turn on audio matrix*)

Response:

```
"data": {  
  audiomatch: "true" or "false"  
}
```

## /api/control/volume/ Set Output Volume

### Set volume

Change the volume for any given output on MHUB\*.

**GET** request:

[http://devicehost/api/control/volume/\[ox\]/\[vy\]/](http://devicehost/api/control/volume/[ox]/[vy]/)

Arguments:

[ox] = output (*a,b,c.....*)

[iy] = volume (*1-100*)

Response:

```
"data": {  
  "output_id": "output id",  
  "volume": "output audio volume"  
}
```

\*Only the follow systems support this api - MHUBAUDIO6455, MZMA6455, MHUB44100A, MHUB66100

## /api/control/volume/step/ Step output volume

### Step output Volume

Increase or decrease the volume incrementally

**GET** request:

[http://devicehost/api/control/volume/step/\[ox\]/\[x\]/](http://devicehost/api/control/volume/step/[ox]/[x]/)

[x] = *up* or *down*

[ox] = output (*a,b,c.....*)

Response:

```
"data": {  
    "zone_id": "zone id",  
    "volume": "increased",  
}
```

\*Only the follow systems support this api - MZMA6455, MHUB44100A, MHUB66100A

## /api/control/volume/zone/ Set Zone Volume (MHUB AUDIO only)

### Set Zone Volume (AUDIO only)

Change the volume for any given zone with MHUB AUDIO outputs.

**GET** request:

[http://devicehost/api/control/volume/zone/\[zid\]/\[x\]](http://devicehost/api/control/volume/zone/[zid]/[x])

[zid] = zone (z1, z2, z3 ....)

[x] = Volume (0-100)

Response:

```
"data": {  
    "zone_id": "zone id",  
    "volume": zone audio volume,  
}
```

## /api/control/arc/ Audio Return Channel (ARC)

### Audio Return Channel (ARC)

ARC can only be routed to MHUB if the mode is enabled - before - the input is switched to. This is to avoid the display's internal speaker from delivering no audio when ARC is not needed.

**GET** request:

[http://devicehost/api/control/arc/\[ox\]/\[ty\]/\[ax\]/](http://devicehost/api/control/arc/[ox]/[ty]/[ax]/)

true = ARC is enabled

false = ARC is disabled

Arguments:

[ox] = output (*a,b,c*)

[ty] = type (0-HDMI, 1-HDBaseT)

[ax] = ARC state (*true=ARC On, false=ARC Off (audio)*)

Response:

```
"data": {  
  "output_id": "output id"  
  "type": "type",  
  "arc": boolean  
}
```

## /api/control/mute/ Mute

### Mute

Mute the audio for any given output on MHUB.

**GET** request:

[http://devicehost/api/control/mute/\[ox\]/\[mx\]/](http://devicehost/api/control/mute/[ox]/[mx]/)

true = mute is enabled

false = mute is disabled

toggle = toggles current mute state

Arguments:

[ox] = output (*a,b,c.....*)

[mx] = mute state (*true* or *false*)

Response:

```
"data": {  
  "output_id": "output id",  
  "mute": boolean  
}
```

## /api/control/mute/zone/ Mute Zone (MHUB AUDIO/MZMA only)

### Mute Zone (MHUB AUDIO only)

Mute the audio for any given zone with MHUB AUDIO/MZMA outputs.

**GET** request:

[http://devicehost/api/control/mute/zone/\[zid\]/\[mx\]/](http://devicehost/api/control/mute/zone/[zid]/[mx]/)

True = mute is enabled

False = mute is disabled

Arguments:

[zid] = zone (z1, z2, z3.....)

Response:

```
"data": {  
  "zone_id": "zone id",  
  "mute": boolean  
}
```



/api/control/output/

Shutdown Amplifier (MHUB AUDIO/MZMA only)

### Turn off output

Turn off the MHUB AUDIO/MZMA outputs.

**GET** request:

[http://devicehost/api/control/output/\[oid\]/\[x\]/](http://devicehost/api/control/output/[oid]/[x]/)

Arguments:

[oid] = output (*a,b,c.....*)

[x] = on - Enables the output

off - Disables the output

Response:

```
"data": {  
  "output_id": "output id",  
  "status": "status"  
}
```

## /api/control/group/create/ Creating a Group

### Create a group

Group a maximum of four zones together for shared volume control.

**GET** request:

[http://devicehost/api/control/group/create/\[groupLabel\]/](http://devicehost/api/control/group/create/[groupLabel]/)

Arguments:

[groupLabel] = label for group (string)

Response:

```
"data": {  
  "group_created": {  
    "group_id": "group id",  
    "label": "group label"  
  }  
}
```

## /api/control/group/delete/ Deleting a Group

### Delete group

Delete the selected group and remove all assigned zones.

**GET** request:

[http://devicehost/api/control/group/delete/\[gid\]/](http://devicehost/api/control/group/delete/[gid]/)

Arguments:

[gid] = ID for group

Response:

```
"data": {  
  "group_deleted": {  
    "group_id": "group id",  
    "label": "group label"  
  }  
}
```

## /api/control/group/[add/delete]/ Adding or removing a Zone from Group

### Add or remove zones from a group

Add or remove multiple zones to/from a group.

**POST** request:

[http://devicehost/api/control/group/\[gid\]/\[op\]](http://devicehost/api/control/group/[gid]/[op])

**POST** body example:

```
{
  "zones": ["zone id", "zone id"]
}
```

Arguments:

[gid] = ID for group

[op] = 'add' or 'delete'

Response::

```
"data": {
  "group": {
    "group_id": "group id",
    "label": "group label"
  },
  "zones": {
    "added": {
      "zone_id": "zone id",
      "label": "zone label"
    },
    "removed": {
      "zone_id": "zone id",
      "label": "zone label"
    }
  }
}
```

## /api/control/group/volume/set/ Set the volume in a Group

### Adjust group audio volume

Changes the audio volume on MHUB AUDIO outputs within the group.

**GET** request:

[http://devicehost/api/control/group/volume/set/\[gid\]/\[vs\]/](http://devicehost/api/control/group/volume/set/[gid]/[vs]/)

Arguments:

[gid] = ID for group

[vs] = volume (1-100)

Response

```
"data": {  
  "group": {  
    "group_id": "group id",  
    "label": "group label"  
  },  
  "group_volume": "mhub audio group volume"  
}
```

## /api/control/group/volume/step/ Changing the volume in a Group

### Adjust group audio volume

Changes the audio volume on MHUB AUDIO outputs within the group.

**GET** request:

[http://devicehost/api/control/group/volume/set/\[gid\]/x/](http://devicehost/api/control/group/volume/set/[gid]/x/)

Arguments:

[gid] = ID for group

[x] = *up* or *down*

Response

```
"data": {  
  "group": {  
    "group_id": "group id",  
    "label": "group label"  
  },  
  "group_volume": "increased"  
}
```

## /api/control/mutegroup/ Muting the audio in a Group

### Changing the mute state in a group

Sets the mute state for MHUB AUDIO outputs within the group.

**GET** request:

[http://devicehost/api/control/mutegroup/\[gid\]/\[ox\]/](http://devicehost/api/control/mutegroup/[gid]/[ox]/)

true = mute is enabled

false = mute is disabled

Arguments:

[gid] = ID for group

[ox] = mute state (*true=muted(no audio), false=unmuted(audio)*)

Response

```
"data": {  
  "group": {  
    "group_id": "group id",  
    "label": "group label"  
  },  
  "group_mute": boolean  
}
```

## /api/control/sequence/ Execute Sequence

### Executing a Sequence

**GET** request:

[http://devicehost/api/control/sequence/\[sid\]/\[y\]](http://devicehost/api/control/sequence/[sid]/[y])

Arguments:

[sid] = Sequence ID

[y] = true or false - *This is optional*

Response

```
"data": {
  "sequence_executed": {
    "id": "sequence id",
    "label": "sequence label"
  }
}
```



## /api/control/function/ Execute Sequence

### Executing a Function

**GET** request:

[http://devicehost/api/control/function/\[fid\]/\[y\]](http://devicehost/api/control/function/[fid]/[y])

Arguments:

[fid] = function ID

[y] = true or false - *This is optional*

Response

```
"data": {  
  "function_executed": {  
    "id": "function id",  
    "label": "function label"  
  }  
}
```

## IO

/api/command/ir/

Execute uControl Command (IR)

### Executing a uControl command

Sends an IR command stored within the uControl device control pack.

**GET** request:

[http://devicehost/api/command/ir/\[io\]/\[cy\]](http://devicehost/api/command/ir/[io]/[cy])

Arguments:

[io] = IR port ID (1,2,3)

[cy] = IR command ID

Response

```
"data": {  
  "execute": true  
}
```

/api/command/ir/hold

Execute uControl Command (IR)

#### Executing a uControl command hold function

Starts sending and IR command every 250ms until release command or another IR ID command is sent.

**GET** request:

[http://devicehost/api/command/ir/\[io\]/\[cy\]/hold](http://devicehost/api/command/ir/[io]/[cy]/hold)

Arguments:

[io] = IR port ID (1,2,3)

[cy] = IR command ID

Response

```
"data": {  
  "execute": true  
}
```

## /api/command/ir/release

### Execute uControl Command (IR)

#### Executing a uControl command

Cancels hold command and stops repeating IR codes

**GET** request:

[http://devicehost/api/command/ir/\[io\]/\[cy\]/release](http://devicehost/api/command/ir/[io]/[cy]/release)

Arguments:

[io] = IR port ID (1,2,3)

[cy] = IR command ID

Response

```
"data": {  
  "execute": true  
}
```

## /api/command/irpass/ IR Passthrough

### Send an IR hex code

Sends an IR Pronto hex string from the selected IR port.

You can find details on the Pronto IR protocol here

<http://files.remotecentral.com/pronto/14-1/index.html>

[http://www.hifi-remote.com/wiki/index.php?title=Working\\_With\\_Pronto\\_Hex](http://www.hifi-remote.com/wiki/index.php?title=Working_With_Pronto_Hex)

**POST** request:

[http://devicehost/api/command/irpass/\[io\]/](http://devicehost/api/command/irpass/[io]/)

**POST** body examples:

Data must be passed to MHUB using JSON. Pronto IR data included within the object can be in any of the following formats below.

- 0000,006b... (4 digit comma separated)
- 0000 006b... (4 digit space separated)

```
{ "irdata": "0000,0072,0000,0016,0062,0022,000f,0012,000f,0012,000f,0022,000f,0022,0020,0012,000f,0012,000f,0012,000f,0012,000f,0012,0020,0021,0020,0011,000f,0012,000f,0022,000f,0012,0020,0011,000f,0022,000f,0012,0020,0011,000f,0022,000f,0012,000f,1192" }
```

Arguments:

[io] = IR port ID (1,2,3)

Response

```
{ "data": {  
  "execute": true  
}
```

/api/command/cec/

## Execute uControl Command (CEC)

### Execute a CEC command from MHUB using HDA's predefined uControl CEC library.

Consumer Electronics Control (CEC) is a feature of HDMI designed to allow predefined/and or supported control commands to be delivered over a HDMI connection via MHUB. This API works in a similar way to the uControl (IR) API, "Execute uControl Command" where parameters command ID and target port is required to send a command.

#### Note

CEC is not standardised across all manufacturers. Whilst the protocol is implemented in a standard way, implementation, naming and command support are not. The uControl CEC library uses standard command definitions and HDANYWHERE can not guarantee that any may work.

#### POST request:

[http://devicehost/api/command/cec/\[io\]/\[ty\]/\[cy\]/](http://devicehost/api/command/cec/[io]/[ty]/[cy]/)

#### Arguments:

[io] = Port ID (*a,b,c.. for outputs, 1,2,3.. For inputs*)

[ty] = CEC command Type (0=HDMI output, 1=HDBT output)

[cy] = CEC command ID

#### Response

```
cec"data": {  
  "execute": true  
}
```

## /api/command/cecpass/ CEC Passthrough

### Pass any CEC command through MHUB

Send a custom CEC command for execution at display or source ends via MHUB. You will need to have a good understanding of CEC syntax, addressing and device support before using this API for best results.

You can create CEC commands here:

<http://www.cec-o-matic.com/>

**POST** request:

[http://devicehost/api/command/cecpass/\[io\]/\[ty\]/](http://devicehost/api/command/cecpass/[io]/[ty]/)

**POST** body examples:

Data must be passed to MHUB using JSON and must appear in the following format:

- 10 00 EF (2 digit empty space " " separated)

The example below will instruct MHUB to request from a display indicated by [io] that it switches to HDMI Input 1.

```
{
  "logicaladdress": "EF",
  "command": "82",
  "arguments": "10 00"
}
```

Arguments:

[io] = Port ID (*a,b,c.. for outputs, 1,2,3.. For inputs*)

[ty] = CEC - Command Type (0=HDMI output, 1=HDBT output)

[cid] = CEC Logical Address

[cec] = CEC Command

[arg] = CEC Arguments

Response

```
"data": {
  "execute": true
}
```

## /api/command/rs232config/ RS232 port configuration

### Configure the RS232 port.

Configure the baud, data length and parity for each RS232 port

**POST** request:

[http://devicehost/api/command/rs232config/\[pt\]/](http://devicehost/api/command/rs232config/[pt]/)

Arguments:

[pt] = port ([See port mapping for details](#))

Data must be passed to MHUB using JSON and must appear in the following format:

- baud rate (**1**-115200, **2**-57600, **3**-56000, **4**-38400, **5**-19200, **6**-14400, **7**-9600, **8**-4800)
- data length (**1**-8, **2**-7, **3**-6, **4**-5)
- parity (**1**-none, **2**-odd, **3**-even)

**POST** body example to set port to 115200, 8bit, no parity:

```
{
  "baud": "1",
  "data": "1",
  "parity": "1"
}
```

### Response

```
"data": {
  "baud": "115200",
  "data": "8",
  "parity": "none"
}
```



## /api/command/rs232pass/ RS232 Passthrough

### Pass any RS232 command through MHUB

Send a RS232 command for execution at display or source ends via MHUB.

**\*\*Port settings are configured on the device.\*\***

**POST** request:

[http://devicehost/api/command/rs232pass/\[io\]/\[ty\]/](http://devicehost/api/command/rs232pass/[io]/[ty]/)

**POST** body examples:

Data must be passed to MHUB using JSON and must appear in the following format:

- ASCII Example - *pwon!*
- HEX Example - *a55b0110ff*

```
{"rs232data": "A5B50110FF"}
```

Arguments:

[io] = Port ID (1,2,3..)

[ty] = Type (0=ASCII, 1=hexadecimal)

Response

```
"data": {  
  "execute": true,  
  "response": ""  
}
```

## /api/command/ip/ Execute uControl Command (IP)

### Executing a uControl command

Sends an IP command stored within the uControl device control pack.

**GET** request:

[http://devicehost/api/command/ip/\[pid\]/\[cy\]](http://devicehost/api/command/ip/[pid]/[cy])

Arguments:

[pid] = Pack id (1,2,3)

[cy] = IP command ID

Response

```
"data": {  
  "execute": true  
}
```

## /api/command/ipass/ IP Passthrough

### Send an IP command

Sends an IP command based on the settings

#### POST request:

<http://devicehost/api/command/ippass/>

#### POST body :

Data must be passed to MHUB using JSON.

```
{"type":"[ty]", "ip":"[ip]",  
  "path":"[pa]", "port":"[po]", "ipdata":"[ipd]", "auth":{"type":  
  "[aty]", "data": {"[akey]":"[aval]"}}}
```

#### Arguments:

[ty] - Type of command (*POST, GET, PUT, TCP, TCPX, UDP, BROADCAST, TELNET*)

[ip] - IP address of target device

[pa] - Path *if required*

[po] - Port on target device

[ipd] - command data

For POST/PUT/TELNET ONLY

[aty] - Type of authorisation (*header, text, login*)

[akey] - Object key

[aval] - Object value

#### Response

```
"data": {  
  "execute": true  
}
```

# UDP State Output

---

**UDP State broadcast**

**Port : 3000**

Output

# HDA Cloud **References**

---

Development of this feature set has been paused.

Do you have any features that you would like to see made accessible from the Internet? If so, please email Stuart Knight ([s.knight@hdanywhere.com](mailto:s.knight@hdanywhere.com)) and tell us about it.

# Reference **Material**

---

## DNS-SD responses

Name	Hostname	Port	Description
<b>MHUB U</b>			
MHUB (4x3+1)	MHUB431U.local	80	MHUB (4x3+1)
MHUB (8x6+2)	MHUB862U.local	80	MHUB (8x6+2)
MHUB U (4x1+1)	MHUBU41140.local	80	MHUB U (4x1+1)
MHUB U (4x3+1)	MHUBU43140.local	80	MHUB U (4x3+1)
MHUB U (8x6+2)	MHUBU86240.local	80	MHUB U (8x6+2)
<b>MHUB PRO</b>			
MHUB PRO (4x4) 40	MHUBPRO4440.local	80	MHUB PRO (4x4) 40
MHUB PRO (4x4) 70	MHUB4K44PRO.local	80	MHUB PRO (4x4) 70
MHUB PRO (8x8) 40	MHUBPRO8840.local	80	MHUB PRO (8x8) 40
MHUB PRO (8x8) 70	MHUB4K88PRO.local	80	MHUB PRO (8x8) 70
<b>MHUB PRO 2.0</b>			
MHUB PRO 2.0 (4x4) 40	MHUBPRO24440.local	80	MHUB PRO 2.0 (4x4) 40
MHUB PRO 2.0 (8x8) 100	MHUBPRO288100.local	80	MHUB PRO 2.0 (8x8) 100
<b>MHUB MAX</b>			
MHUB MAX (4x4)	MHUBMAX44.local	80	MHUB MAX (4x4)
<b>MHUB AUDIO</b>			
MHUB AUDIO (6x4)	MHUBAUDIO64.local	80	MHUB AUDIO (6x4)
MULTI ZONE AMP (6x4) 55	MZA6455.local	80	MULTI ZONE AMP (6x4) 55
<b>ZONE PROCESSOR</b>			
UCONTROL ZONE PROCESSOR (5)	ZP5.local	80	UCONTROL ZONE PROCESSOR (5)
UCONTROL ZONE PROCESSOR (1)	ZP1.local	80	UCONTROL ZONE PROCESSOR (1)
<b>MHUB</b>			
MHUB (4x4) 100 A	MHUB44100A.local	80	MHUB (4x4) 100 A

Table: DNS-SD responses

## Port mapping

	IR				CEC - Display Only			RS232		Audio			
Name	Source	Display	AVR	Global	Input	HDMI Out	HDBT Out	Hub	RX	Source	Output	SKU	Code
<b>MHUB U</b>													
MHUB (4x3+1)	4 (1-4)	3 (6-8)	9	-	-	-	-	-	-			1.05.904.012.1	MHUB431U
MHUB (8x6+2)	8 (1-8)	6 (11-16)	17	-	-	-	-	-	-			1.05.904.013.1	MHUB862U
MHUB U (4x1+1)	4 (1-4)	2 (5-6)	-	2 (7-8)	-	2 (a-b)	1 (b)	-	-			1.05.904.023.1	MHUBU41140
MHUB U (4x3+1)	4 (1-4)	3 (6-8)	9	-	-	-	-	-	-			1.05.904.012.2	MHUBU43140
MHUB U (8x6+2)	8 (1-8)	6 (11-16)	17	-	-	-	-	-	-			1.05.904.013.2	MHUBU86240
<b>MHUB PRO</b>													
MHUB PRO (4x4) 40	4 (1-4)	4 (5-8)	9	-	-	-	-	-	-			1.05.904.014.1	MHUBPRO4440
MHUB PRO (4x4) 70	4 (1-4)	4 (5-8)	9	-	-	-	-	-	-			1.05.904.008.2	MHUB4KPRO44
MHUB PRO (8x8) 40	8 (1-8)	8 (9-16)	17	-	-	-	-	-	-			1.05.904.015.1	MHUBPRO8840
MHUB PRO (8x8) 70	8 (1-8)	8 (9-16)	17	-	-	-	-	-	-			1.05.904.009.2	MHUB4KPRO88
<b>MHUB PRO 2</b>													
MHUB PRO 2.0 (4x4) 40	4 (1-4)	4 (5-8)	9	-	-	4 (a-d)	4 (a-d)	4 (1-4)	4 (5-8)	4 (13-16)	4 (i-l)	1.05.904.021.1	MHUBPRO24440
MHUB PRO 2.0 (8x8) 100	8 (1-8)	8 (9-16)	17	-	-	8 (a-h)	8 (a-h)	8 (1-8)	8 (9-16)	4 (25-28)	8 (q-x)	1.05.904.022.1	MHUBPRO288100
<b>MHUB S</b>													
MHUB S (8+8x8) 100	8 (1-8)	8 (9-16)	17	-	-	2 (a-b)	8 (a-h)	-	8 (1-8)			1.05.904.025.1	MHUBS888100
<b>MHUB MAX</b>													
MHUB MAX (4x4)	-	-	-	-	-	-	-	-	-			1.05.904.011.1	MHUBMAX44
<b>MHUB AUDIO</b>													
MHUB AUDIO (6x4)	-	-	-	-	-	-	-	-	-	6 (1-6)	4 (a-d)	1.05.912.001.1	MHUBAUDIO64
MULTI ZONE AMP (6x4) 55	-	-	-	-	-	-	-	-	-	6 (1-6)	4 (a-d)	1.05.912.002.1	MZA6455
<b>ZONE PROCESSOR</b>													
UCONTROL ZONE PROCESSOR (5)	-	-	-	5 (1-5)	-	1 (a)	-	1 (1)	-			1.80.904.002.1	ZP5
UCONTROL ZONE PROCESSOR (1)	-	-	-	1	-	-	-	1 (1)	-			1.80.904.001.1	ZP1
<b>MHUB</b>													
MHUB (4x4) 100 A	4 (1-4)	4 (5-8)	9	-	4 (1-4)	1 (a)	4 (a-d)	-	4 (1-4)	-	4 (a-d)	1.5.904.026.1	MHUB44100A



## API glossary

uControl uses button IDs for common remote control functions. The following table details remote control buttons with corresponding uControl command IDs. These can be passed into MHUB for execution.

Label	Description	Data Type
api	Indicates API version	Integer (8 bit)
arc input id	If ARC is enabled it input id will be shown here	Integer (8 bit)
audio input	Data array detailing information relating to audio input	Array[id, type, unit id, draw (Boolean), label, label editable, start label, start label prefix, start label suffix, startid, ports]
audio input id	MHUB-OS identifier for audio input(s)	Integer (8 bit)
audio output	Data array detailing information relating to audio output	Array[id, port pair, type, unit id, draw (Boolean), label, label editable, start label prefix, start label suffix, startid, ports]
audio output mirror	Data array detailing information relating to audio output mirror	Array[id, port pair, type, unit id, mirror, draw (Boolean), label, label editable, start label, start label prefix, start label suffix, startid, ports]
cec code	CEC hex string	String (32 characters)
command id	Numerical code used to identify different control commands	Integer (8 bit)
command type	Indicates if command is sent to HDMI output or HDBT	String (32 characters)
display audio input id	MHUB-OS identifier for audio input(s) which can be routed to displays via ARC	Integer (8 bit)
display power	Indicates the current power state of a display. will return true/false/unknown	String (32 characters)
execution time	time in milliseconds (ms) a sequence takes to execute	Integer (8 bit)
function arguments	True or false operator for function	Boolean (true/false)
function data	Data array detailing user defined functions including button labels, voice utterance triggers and function descriptions	array[function id, function name, function description, function type]
function description	Text string describing function action	String (32 characters)
function id	unique number for identifying function	Integer (8 bit)
function type	function command time. Such as IP, IR and CEC	String (3 characters) (g1,g2.....g10)
first boot	Boolean value that shows if first boot has been completed	Boolean (true/false)
group id	MHUB-OS identifier for group(s)	String (3 characters) (g1,g2.....g10)
group label	The user defined label for a collection of zones (e.g. Downstairs, Upstairs)	String (32 characters)
id	Array identifier used to separate input/output types	String (32 characters)
input id	MHUB-OS identifier for input(s)	Integer (8 bit)
input label	The user defined label for inputs (start points) onboard MHUB (e.g. Bluray, Denon AVR, Sonos, Apple TV)	String (32 characters)
ip address	MHUB-OS current IP address	String (32 characters)
ir	Data detailing information relating to the IR ports	backwards(draw, start id, ports), forwards(draw,

# HDANYWHERE

Seamless Entertainment Discreet Technology

		start id, ports), avr(draw, start id, ports)
ir port	Identifier for Infrared (IR) port addressing (endpoints)	Integer (8 bit)
label	Text label used for control commands	String (32 characters)
label editable	Boolean value that shows if label is user modifiable	Boolean (true/false)
mhub audio group volume	Indicates an average volume level of MHUB AUDIO outputs (endpoints) combined in to a group	Integer (8 bit)
mhub firmware	Indicates MHUB firmware version	Decimal (double)
mhub name	The user defined name for MHUB (e.g. Joe's Mhub)	String (32 characters)
mhub official name	The official HDANYWHERE given name for MHUB	String (32 characters)
mhub-os firmware	Indicates MHUB-OS firmware version	Decimal (double)
mhub-os version	Indicates MHUB-OS version	Decimal (double)
output audio volume	Indicates output volume level	Integer (8 bit)
output id	MHUB-OS identifier for output(s)	String (3 Characters) (a,b,c,...)
output label	The user defined label for outputs (endpoints) onboard MHUB (e.g. LG, KEF Soundbar, Sony Projector)	String (32 characters)
paired audio output	This defined the input for a paired output.	Integer (8 bit)
ports	The number of IR ports in each group	Integer (8 bit)
power	Indicates if the MHUB device is in an operational (on or off) state. This feature is not supported across all MHUB devices. Please contact HDANYWHERE for further information.	Boolean (true=ON, false=OFF)
sequence data	Data array detailing user defined Sequences including button labels, voice utterance triggers and sequence descriptions and execution time	array [sequence id, sequence name, sequence description, sequence voice label, execution time, function type, function arguments]
sequence description	Text string describing sequence function	String (32 characters)
sequence id	unique number for identifying function	Integer (8 bit)
sequence name	Text string with name used in MHUB-OS/uOS	String (32 characters)
sequence voice label	Label used with voice control such as Alexa and Google Home	String (32 characters)
serial number	The official HDANYWHERE serial number for MHUB	String (32 characters)
stack audio input	Data array detailing information relating to video inputs in a stack.	Array[total stack audio input, id, unit id, type, draw, start id, labels]
stack audio output	Data array detailing information relating to video inputs in a stack.	Array[total stack audio output, id, unit id, type, draw, start id, labels]
stack audio output mirror	Data array detailing information relating to video inputs in a stack.	Array[total stack audio output mirror, id, unit id, type, mirror, draw, start id, labels]
stack input video	Data array detailing information relating to video inputs in a stack.	Array[total stack video input, id, unit id, type, draw, start id, labels]
stack master	Indicates which MHUB device is designated as Master within a stacked system	mhub name, serial

# HDANYWHERE

Seamless Entertainment Discreet Technology

	arrangement	number, ip address
stack rank	Indicates MHUB-OS's understanding of stack hierarchy for each device connected to MHUB in a stacked system arrangement	Master, Slave, Control Device, Accessory
stack video output	Data array detailing information relating to video outputs in a stack.	Array[total stack video output, id, unit id, type, draw, start id, labels]
stack video output mirror	Data array detailing information relating to video outputs in a stack.	Array[total stack video output mirror, id, unit id, type, mirror, draw, start id, labels]
start id	First identifier for Infrared (IR) port addressing (endpoints) on group	Integer (8 bit)
start label	User visible port id	String (32 characters)
status	Output status (on = output enabled, off = output disabled)	String (32 characters)
type	Describes type of port	String (32 characters)
total stack zone qty	The sum of [total stack video output] + [total stack audio output] provides the maximum addressable number of zones which a stacked MHUB system will support	Integer (8 bit)
ucontrol device type	Indicates what device category is within given uControl Pack	String (32 characters)
ucontrol pack command	Data array detailing supported remote control functions, their MHUB-OS identifiers and labels	Array [code id, code label, repeat (Boolean)]
ucontrol pack draw	Data array detailing default HDANYWHERE button location and placement	Array [ui type, device size, code id, location]
ucontrol pack id	MHUB-OS identifier for uControl Pack(s)	Integer (16 bit)
ucontrol pack method	Indicates what control protocol is used within given uControl Pack	String (32 characters) (CEC, IR, IP)
ucontrol pack name	The official HDANYWHERE given name for uControl Pack	String (32 characters)
ucontrol pack version	Indicates uControl Pack version	Integer (8 bit)
unit id	MHUB-OS identifier for MHUB systems in a standalone or stacked arrangement	Standalone Video "V1", Standalone Audio "A1", Stacked Master "M1", Slave MHUB Devices "S1"
video input	Data array detailing information relating to video input	Array[id, type, unit id, draw, label, label editable, start label, start label prefix, start label suffix, startid, ports, labels]
video output	Data array detailing information relating to video output	Array[id, type, unit id, draw (Boolean), label, label editable, start label, start label prefix, start label suffix, startid, ports]
video output mirror	Data array detailing information relating to video output mirror	Array[id, type, unit id, mirror, draw (Boolean), label, label editable, start label, start label prefix, start label suffix, startid, ports]
zone audio volume	Indicates the volume level of a zone	Integer (8 bit)
zone id	MHUB-OS identifier for zone(s)	String (3 Characters) (z1, z2,...z99)
zone label	The user defined label for addressable areas which includes one or more endpoints (e.g. Kitchen, Bedroom, Bar Area)	String (32 characters)

## uControl command IDs

uControl uses button IDs for common remote control functions. The following table details remote control buttons with corresponding uControl command IDs. These can be passed in to MHUB for execution in section “MHUB IO”.

ID	Command	IR	CEC
0	0		
1	1		
2	2		
3	3		
4	4		
5	5		
6	6		
7	7		
8	8		
9	9		
10	Channel Up		
11	Channel Down		
12	Play		
13	Pause		
14	Play / Pause (Toggle)		
15	Stop		
16	Forward		
17	Skip Forward		
18	Rewind		
19	Skip Backward		
20	Record		
21	Up		
22	Down		
23	Left		
24	Right		
25	Select / Enter / OK		
26	Back / Return / Cancel		
27	Menu		
28	Home		
29	TV Guide		

# HDANYWHERE

Seamless Entertainment Discreet Technology

30	Red		
31	Green		
32	Yellow		
33	Blue		
34	Volume Up		
35	Volume Down		
36	Mute (Toggle)		
37	Info		
38	Power (Toggle)		
39	Power Off (Explicit)		
40	Subtitles		
41	Source		
42	Eject		
43	Audio		
44	3D		
45	Top Menu		
46	Pop-up Menu		
47	Aspect Ratio		
48	Help		
49	Power On (Explicit)		
50	No / Thumbs Down		
51	Yes / Thumbs Up		
52	Page Up		
53	Page Down		
54	Search		
55	Exit		
56	Catch Up		
57	Playlist		
58	Box Office		
59	DSTV		
60	BD/DVD		
61	CBL/SAT		
62	GAME		
63	NET		
64	BT AUDIO		
65	USB		
66	TUNER		
67	TV		

# HDANYWHERE

Seamless Entertainment Discreet Technology

68	CD		
69	INPUT DN		
70	INPUT UP		
71	AUTO ADJUST		
72	CONTRAST		
73	BRIGHTNESS		
74	DISPLAY		
75	HDMI 1		
76	HDMI 2		
77	HDMI 3		
78	HDMI 4		
79	CD-R		
80	DVR/VCR 1		
81	BD		
82	TV/SAT		
83	DVD/LD		
84	VIDEO/GAME 1		
85	DVR/VCR 2		
86	PHONO		
87	SETUP		
88	CLEAR		
89	REPEAT		
90	MEDIA Center		
91	MOUSE		
92	BOOKMARKS		
93	NOVA SCOPE		
94	ON DEMAND		
95	EXTRA		
96	FAVORITES		
97	Showmax		
98	INPUT		

99	LIST		
100			
101	SLOW PLAY		
102	REVERSE/PAUSE		
103	MULTI		
104	ADD		

# HDANYWHERE

Seamless Entertainment Discreet Technology

105	COMPONENT		
106	NETFLIX		
107	PRIME VIDEO		
108	YOUTUBE		
109	NOWTV		
110	APPLETV		
111	DISNEY+		
112	BASS-		
113	BASS+		
114	TREBLE-		
115	TREBLE+		
116	PC 1		
117	PC 2		
118	S VIDEO 1		
119	S VIDEO 2		
120	S VIDEO 3		
121	ANYNET		
122	VGA 1		
123	VGA 2		
124	INPUT VIDEO 1		
125	INPUT VIDEO 2		
126	INPUT VIDEO 3		
127	AV 1		
128	AV 2		
129	AV 3		
130	COMPONENT 1		
131	COMPONENT 2		
132	COMPONENT 3		
133	DTV		
134	DVI 1		
135	DVI 2		
136	Q MENU		
137	PAGE UP		
138	PAGE DOWN		
139	DEV		
140	AUX 1		
141	STRM BOX		
142			

# HDANYWHERE

Seamless Entertainment Discreet Technology

143	TAPE		
144	INPUT VIDEO 4		
145	INPUT VIDEO 5		
146	INPUT VIDEO 6		
147	INPUT VIDEO 7		
148	SURROUND MODE		
149	AUX 2		
150	Bluetooth		
151	Internet Radio		
152	Media Player		
153	Online Music		
154	Input Analog		
155	Options		
156	Recordings		
157	apps		
158	8K		
159	HEOS		
160	FILM 1		
161	FILM2		
162	REF		
163	PHOTO		
164	BRT CINE		
165	BRT TV		
166	USER		
167	Motionflow		
168	Color space		
169	Color Temp		
170	Color Correction		
171	Gamma Correction		
172	Contrast enhancer		
173	Reality Creation		
174	Sharpness +		
175	Sharpness -		
176	Brightness +		
177	Brightness -		
178	Contrast +		
179	Contrast -		
180	Pattern		



# HDANYWHERE

Seamless Entertainment Discreet Technology

181	Reset		
182	Resize		
183	Overscan		
184	Custom		
185	Presets		
186	Source 1		
187	Source 2		
188	Source 3		
189	Room EQ		
190	Dolby/DTS		
191	Pure Direct		
192	Dynamic EQ		
193	Night Mode		
194	Speaker Level		
195	10		
196	A-B Repeat		
197	Angle		
198	Resolution		

199	Dimmer		
200	Dimmer		
201	Viera Tools		
202	HBO		
203	TV Portal		
204	TEXT		
205	standby		
206	HDMI 5		
207	HDMI 6		
208	HDMI 7		
209	A		
210	B		
211	C		
212	D		
213	AUTO SURR		
214	STEREO		
215	Display Port		
216	4 DIV		
217	8 DIV		
218	ESC		

# HDANYWHERE

Seamless Entertainment Discreet Technology

219	LANGUAGE		
220	TV/R		
221	SAT/TER		
222	Hot Key		
223	Sleep		
224	V.Format		
225	LIVE		
226	GROUP		
227	MUTE ON		
228	MUTE OFF		
229	TRIGGER 1 ON		
230	TRIGGER 1OFF		
231	SPOTIFY		
232	AIRPLAY		
233	SOURCE 4		
234	SOURCE 5		
235	SOURCE 6		
236	SOURCE 7		
237	SOURCE 8		
238	SOURCE 9		
239	SOURCE 10		
240	SOURCE 11		
241	SOURCE 12		
242	SOURCE 13		
243	PURE DIRECT OFF		
244	ENAHANCER AUTO		
245	ENHANCER OFF		
246	DVR/NVR		
247	SEQ		
248	FREEZE		
249	ZOOM IN		
250	ZOOM OUT		
251	SPOT/MON		
252	AUTO PAN		
253	180 DEG		
254	BACKUP		
255	9 DIV		
256	13 DIV		

# HDANYWHERE

Seamless Entertainment Discreet Technology

257	16 DIV		
258	25 DIV		
259	32 DIV		
260	DOWNLOAD		
261	CH LIST		
262	NOTE		
263	Shuffle On		
264	Shuffle Off		
265	KEYSTONE		
266	CAST		
267	A1		
268	A2		
269	A3		
270	A4		
271	B1		
272	B2		
273	B3		
274	B4		
275	C1		
276	C2		
277	C3		
278	C4		
279	D1		
280	D2		
281	D3		
282	D4		
283	Plus		
284	Samsung TV Plus		
285	Freeview		
286	AV		
287	BBC iPLAYER		
288	ITV HUB		
289	Keyboard		
290	PiP		
291	AD		
292	Decode		
293	Google Home		
294	5/7 Channel Stereo		

# HDANYWHERE

Seamless Entertainment Discreet Technology

295	Random		
296	Ambilight Up		
297	Ambilight Down		
298	Amblight Mode		

299	SCART		
300	Side/Front Input		
301	Clock		
302	Smart Picture		
303	Direct Mode		
304	FM		
305	PVR		
306	STB		
307	Active		
308	Planner		
309	Sky		
310	Focus IN		
311	Focus OUT		
312	Format Scroll		
313	Input Component		
314	Input Composite		
315	Resync		
316	User 1		
317	User 2		
318	User 3		
319	Picture Mode		