www.plexusAV.com
sales@plexusav.com
+1.605.9780.4800

P-AVN2/4 GraphQL API GUIDE

P-AVN-2/4 firmware 1.1.0

April 15, 2025 - v1

www.plexusAV.com
sales@plexusav.com
+1.605.9780.4800

Contents

www.plexusAV.com
sales@plexusav.com
+1.605.9780.4800

Introduction

This document describes the GraphQL API for PlexusAV P-AVN-2 and P-AVN-4, based on firmware release 1.1.x. Because the P-AVN-2 uses a sub-set of features available on the P-AVN-4, some GraphQL commands in this document may not be available on the P-AVN-2.

Most commands show usage examples. A full list of API commands is available in a separate document or may be found in the GraphQL Playground schema. More information in the "GraphQL API" section.

www.plexusAV.com
sales@plexusav.com
+1.605.9780.4800

NMOS vs PlexusAV API

The Networked Media Open Specifications (NMOS) is a control and management layer used in the IPMX open AV over IP standard. Since both P-AVN-2 and P-AVN-4 are IPMX based, they are NMOS based.

However, NMOS is not easy to implement and uses multiple API's. That's why PlexusAV provides a GraphQL API that is much easier to use.

This document describes how to control each P-AVN-2/4 directly using the GraphQL API. The PlexusAV Visual *Array*, our single-point-of-control management platform, also has a GraphQL API.

www.plexusAV.com
sales@plexusav.com
+1.605.9780.4800

Getting started

API Version Management

This document is based on P-AVN-2/4 firmware version 1.1. The API commands are not expected to change until we release a 2.0 or higher version.

www.plexusAV.com
sales@plexusav.com
+1.605.9780.4800

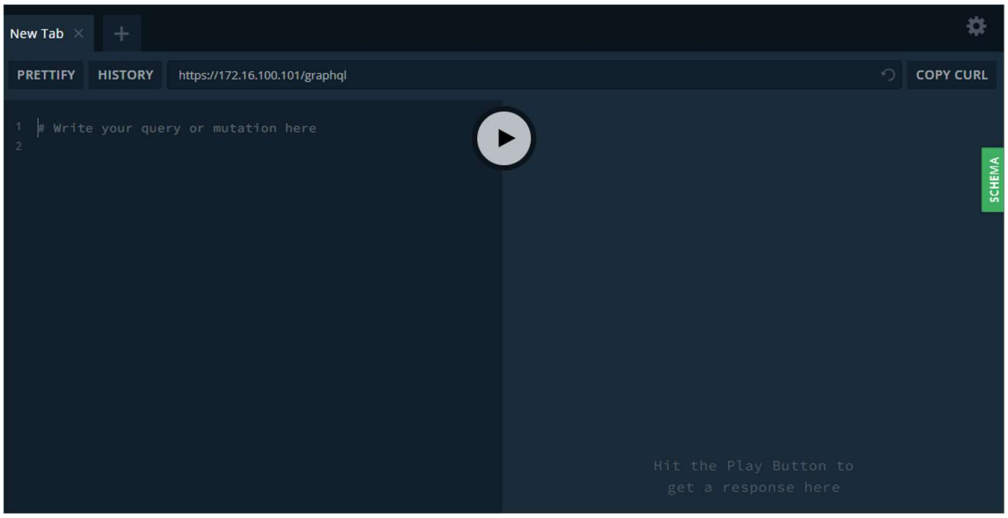
GraphQL API

The GraphQL API that is being used in PlexusAV P-AVN-2/4 and Visual Array has many advantages over the traditional REST API. GraphQL utilizes **a single endpoint** for sending queries or mutations, allowing clients to request **exactly the data they need**.

This provides a single URL for all requests, which may be customized as needed.

For the P-AVN-2/4 the URL is: **https://[P-AVN-4-IP-Address]/graphql**
The IP port being used is the standard https port **443**.

This URL connects to the GraphQL Playground on the unit and is helpful for testing API commands.



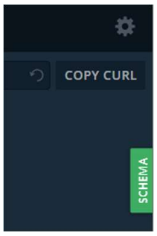
GraphQL uses **Queries** and **Mutations**:

- **Queries**: Used for retrieving data from the server
  (Like GET in REST API)
- **Mutations**: Used for retrieving data from the server.
  (Like POST, PUT and DELETE in REST API)

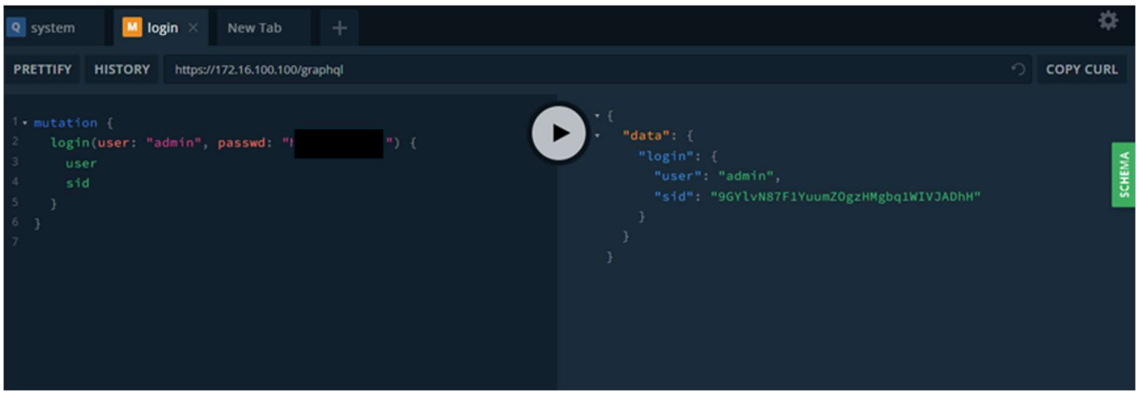
A GraphQL schema is a crucial part of GraphQL, defining the structure of the data that can be queried. It specifies the types of data available, the relationships between those types, and the operations that can be performed on the data. A GraphQL scheme includes queries and mutations.

www.plexusAV.com
sales@plexusav.com
+1.605.9780.4800

You can see the full GraphQL schema by clicking the green schema button in the GraphQL playground.



Here is an example of a P-AVN-4/2 GraphQL command in the Playground. We will explain it in the next section.



The PlexusAV GraphQL API uses the JSON format for a response, which follows the shape of the GraphQL request.

For more control and a single point to send commands from to test, we recommend using Postman. From now on, we will use Postman in all the examples. The following section explains how to set up Postman for P-AVN-2/4 GraphQL commands.

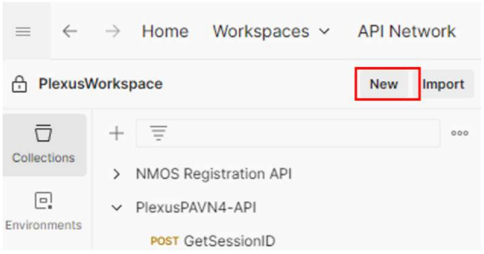
www.plexusAV.com
sales@plexusav.com
+1.605.9780.4800

Postman

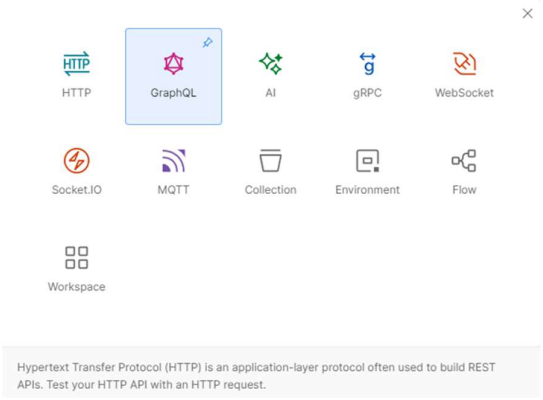
After Postman has been installed, it needs to be configured to send GraphQL requests to the PlexusAV P-AVN-2/4.

Open Postman and create a request

- Click on "New" → Select "Request"



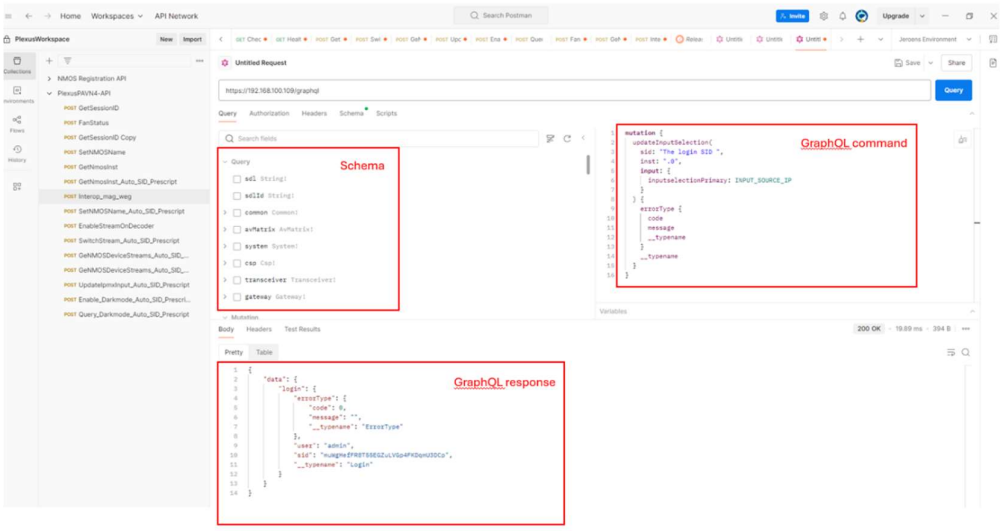
- Choose "GraphQL" as the request type



- Enter the GraphQL API endpoint (https://[P-AVN-2/4 IP Address]/graphql)
- Enter the GraphQL request

www.plexusAV.com
sales@plexusav.com
+1.605.9780.4800

- Click Query to send the command



Please note that you can see the complete GraphQL schema from here.

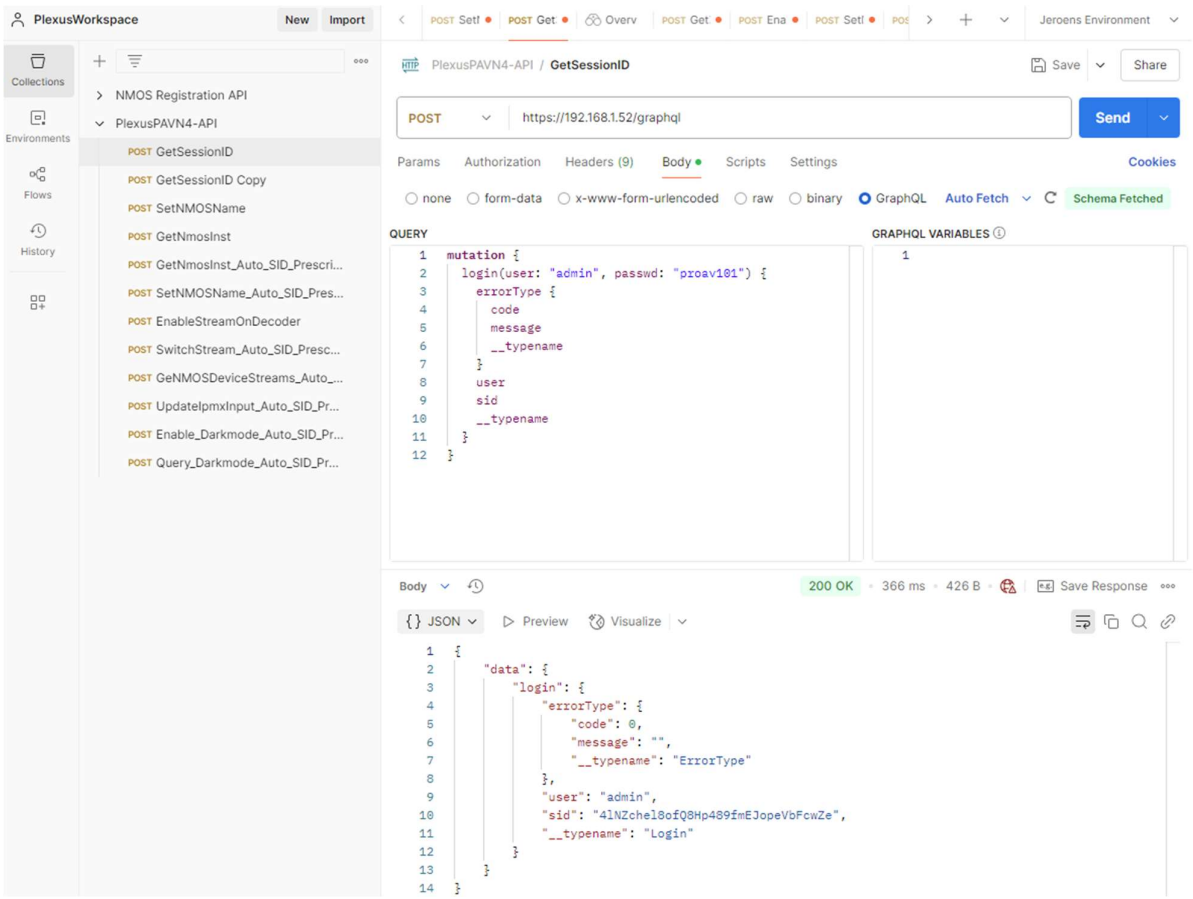
Variables and prescripts (optional)

- Using variable and prescripts (for getting the SID) in Postman simplifies the usage
- However, the use of variables and prescripts is beyond the scope of this document

AVN2/4 Log in

Log into the P-AVN-2/4 to get a unique session ID for sending other queries and mutations. Instead of entering username and password in each API request, we use a **Session ID** (referred to as **SID** throughout this document) also called a Token. It is valid for **30 minutes**, so when 30 minutes have passed a new SID must be generated by the P-AVN-2/4.

To **get a Session ID Token (SID)**, we send a Mutation with Field "login" and Arguments "user" and "passwd". Within the "login" field we specify the "user" and "sid" fields to return these values from the mutation.



www.plexusAV.com
sales@plexusav.com
+1.605.9780.4800

Code:

```
mutation {
  login(user: "admin", passwd: "proav101") {
    errorType {
      code
      message
      __typename
    }
    user
    sid
    __typename
  }
}
```

Response “getNmosDevice”

"user": "admin"

The username

"sid": "B1fa5egrqJ690dj8mIcCrG0eoDNqRWWc"

The “sid” that will be valid for 30 minutes

www.plexusAV.com
sales@plexusav.com
+1.605.9780.4800

Error handling

With the errorType response, we get feedback for the command:


```
"errorType": {
    "code": 0,
    "message": "",
    "__typename": "ErrorType"
},
```

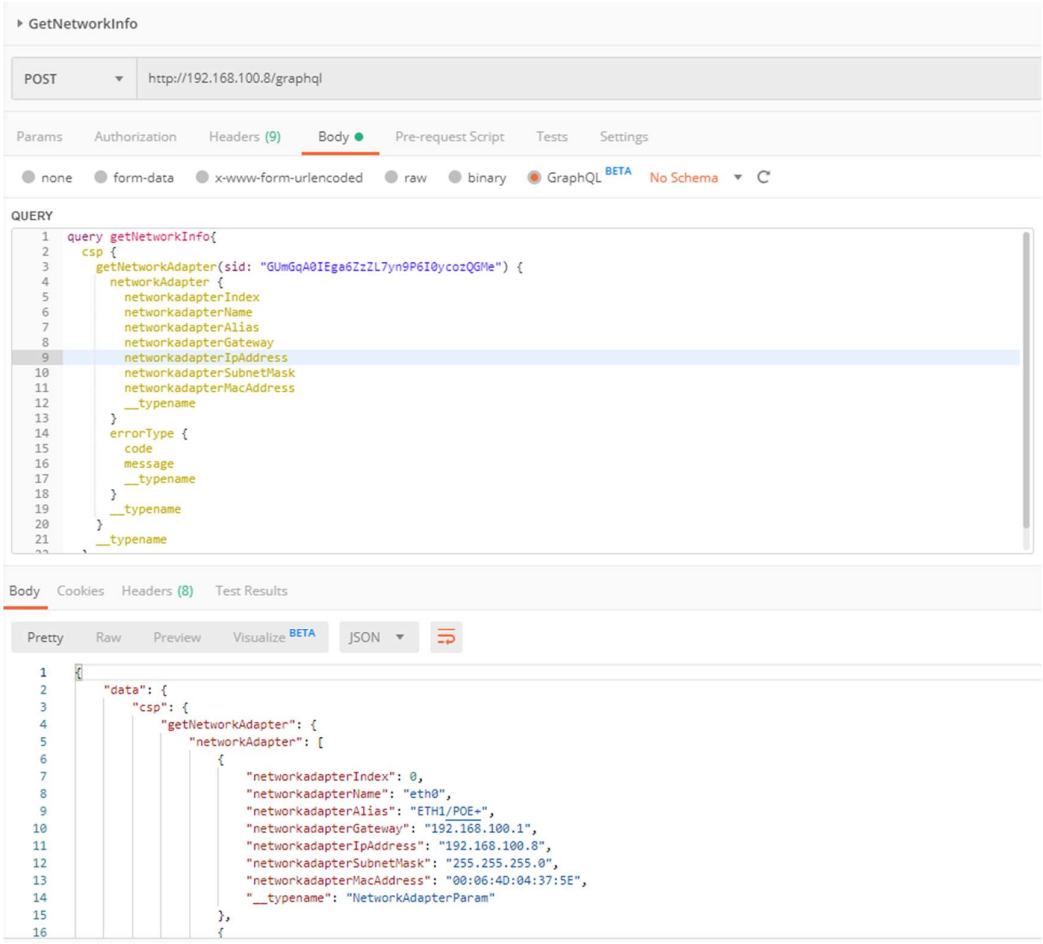

Error code values:

code: 0 - success and message will be blank
code: 1 - failure and the message will show the detailed errors

www.plexusAV.com
sales@plexusav.com
+1.605.9780.4800

First Query: Get Network Information

Let's request the P-AVN-2/4 Network Information as a first Query example. First, request the SID from the unit, then use this SID in the query to get the Network Information.



www.plexusAV.com
sales@plexusav.com
+1.605.9780.4800

Code:

```
query getNetworkInfo {
  csp {
    getNetworkAdapter(sid: " paste your  received SID here ") {
      networkAdapter {
        networkadapterIndex
        networkadapterName
        networkadapterAlias
        networkadapterGateway
        networkadapterIpAddress
        networkadapterSubnetMask
        networkadapterMacAddress
        __typename
      }
      errorType {
        code
        message
        __typename
      }
      __typename
    }
    __typename
  }
}
```

Response “getNetworkAdapter”

"networkadapterIndex": 0

The index of network adapter.

"networkadapterName": "eth0" ["eth1", "eth2"]

The name of network adapter.

"networkadapterAlias": "ETH1/POE+"

The alias of network adapter.

"networkadapterGateway": "192.168.100.1"

Network adapter gateway.

"networkadapterIpAddress": "192.168.100.8"

Network adapter IP Address.

"networkadapterSubnetMask": "255.255.255.1"

The subnet mask of network adapter.

"networkadapterMacAddress": "00:06:4D:04:37:5E"

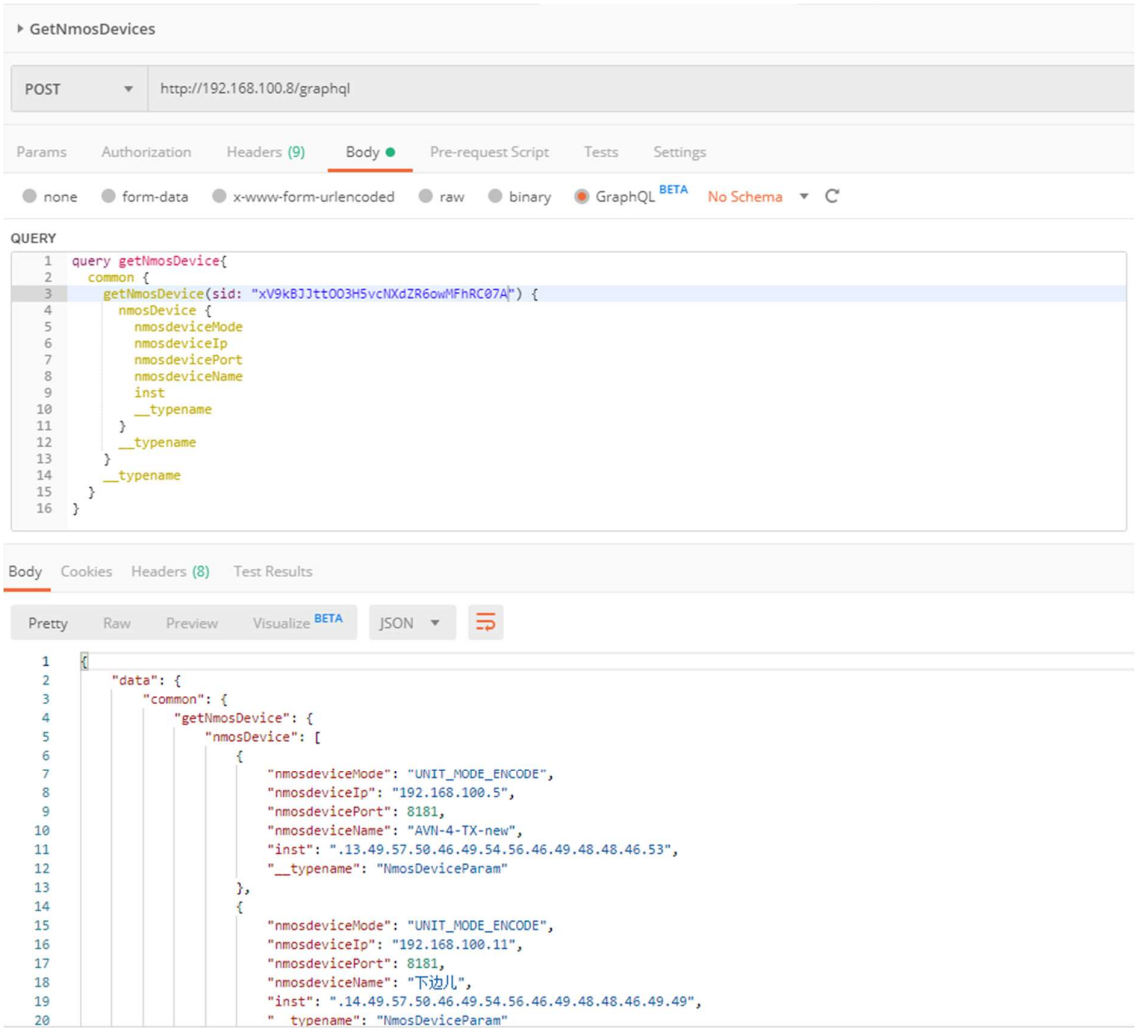
The MAC Address of network adapter.

www.plexusAV.com
sales@plexusav.com
+1.605.9780.4800

Discovering IPMX devices

In P-AVN-2/4, **Transceivers are identified by their NMOS Instances** (NMOS Inst.), you can think of them as IDs. They are fixed values and will never change. You can request all detected transceiver names and their associated Instances. You can ask any P-AVN-2/4 to get information about every detected device, excluding the device you have sent the request to.

If you want to know the inst. of the device itself, you can send the same request to another transceiver.



www.plexusAV.com
sales@plexusav.com
+1.605.9780.4800

Code:

```
query getNmosDevice {
  common {
    getNmosDevice(sid: " paste your  received SID here ") {
      nmosDevice {
        nmosdeviceMode
        nmosdeviceIp
        nmosdevicePort
        nmosdeviceName
        inst
        __typename
      }
      __typename
    }
    __typename
  }
}
```

Response “getNmosDevice”

"nmosdeviceMode": "UNIT_MODE_ENCODE" [“UNIT_MODE_DECODE”]

The device working mode, encode/transmitter or decode/receiver.

"nmosdeviceIp": "192.168.100.5"

NMOS interface IP address

"nmosdevicePort": 8181

NMOS service port.

"nmosdeviceName": "AVN-4-TX-new"

NMOS Name.

www.plexusAV.com
sales@plexusav.com
+1.605.9780.4800

Discovering IPMX Streams

Multiple GraphQL commands need to be sent to get all the information related to Streams. However, we can combine them in a single command using **"getNmosStream"**.

Generally, we need to retrieve the following related to available streams:

- **IPMX stream information: "getNmosDeviceStream"**
  This gives us the properties of the IPMX essence streams: audio, video and ancillary data.

  Both outgoing streams (from encoders) and incoming streams (into decoders) are listed. Be aware that the transceiver the command is sent to will not list information about itself.

  So if you want to discover all available transmitter outgoing IPMX streams, you should send the command to a RECEIVER.

  [Can be sent to any transceiver to get the information from ALL NMOS devices]

- **USB connection information: "getUsbIpServer"**
  This will give us the USB connection information.
  [This command is sent to individual RX transceivers for the KVM routing information]

- **Audio and video input information: "inputSelection"**
  This will give audio and video input information.
  [This command is sent to individual transceivers]

www.plexusAV.com
sales@plexusav.com
+1.605.9780.4800

▸ GetNmosStream

POST http://10.200.0.219//graphql

Params Authorization Headers (9) Body ● Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL No Schema

QUERY

```
query getNmosStream{
  common {
    getNmosDeviceStream(sid: "YhbED14KfIsOj4o3XyDk9SGKUAnu6hHc") {
      nmosDeviceStream {
        nmosdevicestreamStatus
        nmosdevicestreamName
        nmosdevicestreamTransportMode
        nmosdevicestreamIp
        nmosdevicestreamPort
        nmosdevicestreamType
        nmosdevicestreamChannel
        nmosdevicestreamUnicastIp
        nmosdevicestreamConnect
```

Body Cookies Headers (8) Test Results

Pretty Raw Preview Visualize JSON


```
{
    "data": {
        "common": {
            "getNmosDeviceStream": {
                "nmosDeviceStream": [
                    {
                        "nmosdevicestreamStatus": "UNLOCKED",
                        "nmosdevicestreamName": "P-AVN-4_7289436-JXS",
                        "nmosdevicestreamTransportMode": "IP_STREAM_MODE_MULTICAST",
                        "nmosdevicestreamIp": "10.200.0.102",
                        "nmosdevicestreamPort": 36686,
                        "nmosdevicestreamType": "RECEIVER",
                        "nmosdevicestreamChannel": "JPEGXS",
                        "nmosdevicestreamUnicastIp": "10.200.0.102",
                        "nmosdevicestreamConnect": "DISABLED",
                        "inst": ".f9c2bf8a-0d4f-15e8-a000-00064d044766",
                        "__typename": "NmosDeviceStreamParam"
                    },
```

www.plexusAV.com
sales@plexusav.com
+1.605.9780.4800


```
            ],
            "__typename": "NmosDeviceStream"
        },
        "__typename": "Common"
    },
    "transceiver": {
        "getUsbIpServer": {
            "usbIpServer": [
                {
                    "inst": ".1",
                    "usbipserverEnable": "DISABLED",
                    "usbipserverClientIpAddress": "10.200.0.127",
                    "usbipserverConnectStatus": "STATUS_FAILURE",
                    "__typename": "UsbIpServerParam"
                }
            ],
            "__typename": "UsbIpServer"
        },
        "getInputSelection": {
            "inputSelection": [
                {
                    "inst": ".0",
                    "inputselectionPrimary": "INPUT_SOURCE_IP",
                    "inputselectionAudioSource": "INPUT_AUDIO_SOURCE_IP",
                    "inputselectionIpGroup": "INPUT_IP_GROUP1",
                    "__typename": "InputSelectionParam"
                }
            ],
            "__typename": "InputSelection"
        },
        "__typename": "Transceiver"
    }
  }
}
```

www.plexusAV.com
sales@plexusav.com
+1.605.9780.4800

Code:

```
query getNmosStream {
  common {
    getNmosDeviceStream(sid: " paste your  received SID here ") {
      nmosDeviceStream {
        nmosdevicestreamStatus
        nmosdevicestreamName
        nmosdevicestreamTransportMode
        nmosdevicestreamIp
        nmosdevicestreamPort
        nmosdevicestreamType
        nmosdevicestreamChannel
        nmosdevicestreamUnicastIp
        nmosdevicestreamConnect
        inst
        __typename
      }
      __typename
    }
    __typename
  }
  transceiver {
    getUsbIpServer(sid: " paste your  received SID here ") {
      usbIpServer {
        inst
        usbipserverEnable
        usbipserverServerIpAddress
        usbipserverConnectStatus
        __typename
      }
      __typename
    }
    getInputSelection(sid: " paste your  received SID here ") {
      inputSelection {
        inst
        inputselectionPrimary
        inputselectionAudioSource
        inputselectionIpGroup
        __typename
      }
      __typename
    }
    __typename
  }
}
```

www.plexusAV.com
sales@plexusav.com
+1.605.9780.4800

Let's analyze the response. We see the "Common" section, this means that all NMOS devices will be listed here. Not only P-AVN-2/4 but e.g. also Stream Conversion Gateway NMOS devices.

Reponse "getNmosDeviceStream"

"nmosdevicestreamStatus": "UNLOCKED" ["LOCKED"]

For receivers this shows the AV Link status, if the receiver is successfully tuned to an incoming stream (LOCKED). For transmitters "LOCKED" means that it is successfully sending a Unicast stream.

"nmosdevicestreamName": "BrightSign-H.26x"

The stream name

"nmosdevicestreamTransportMode": "IP_STREAM_MODE_MULTICAST" [IP_STREAM_MODE_UNICAST]

The stream mode, unicast or multicast

"nmosdevicestreamIp": "239.192.1.4"

The IP address of the stream

"nmosdevicestreamPort": 5004

The port of the stream

"nmosdevicestreamType": "SENDER" [RECEIVER]

The stream type: outgoing (SENDER) or incoming (RECEIVER)

"nmosdevicestreamChannel": "VCU" ["JPEGXS"]

Indicates whether VCU (H.26x) or JPEG-XS (JPEGXS) is being used

"nmosdevicestreamUnicastIp": "192.168.1.52"

The device NMOS interface's IP address

"nmosdevicestreamConnect": "DISABLED" ["ENABLED"]

The state of the stream

"inst": ".8c3d9129-7fc3-1427-a000-00064d043752"

An IPMX stream has a name returned in "nmosdevicestreamName" and is identified by the returned "inst". The "inst" is used when switching streams. Do not confuse the *Stream* "inst" with the *NMOS* inst; they are two different things.

www.plexusAV.com
sales@plexusav.com
+1.605.9780.4800

Response “getUsbIpServer”

“inst”: ".0"

The instance ID of getUsbIpServer, fixed to ".0"

“usbipserverEnable”: "ENABLED" [“DISABLED”]

USB over IP Feature Toggle. The USB server is enabled when you create a USB crosspoint. The USB server is running on the Encoder.

“usbipserverServerIpAddress”: “192.168.100.66”

The target USB Server(Encoder) IP address that you are going to connect to. The USB client is the Decoder.

“usbipserverConnectStatus”: “STATUS_OK” [“STATUS_FAILURE”,“STATUS_NOT_APPLICABLE”]

The KVM connection status to the server(Encoder).

www.plexusAV.com
sales@plexusav.com
+1.605.9780.4800

Response “getInputSelection”

“inst”: ".0"

The instance ID of inputSelection sheet, fixed to ".0"

“inputselectionPrimary”: "INPUT_SOURCE_IP" [INPUT_SOURCE_HDMI, INPUT_SOURCE_USB]

The input selection of video. Supported options: IP(INPUT_SOURCE_IP), HDMI (INPUT_SOURCE_HDMI), USB(INPUT_SOURCE_USB)

“inputselectionAudioSource”: "INPUT_AUDIO_SOURCE_IP" [INPUT_AUDIO_SOURCE_HDMI, INPUT_AUDIO_SOURCE_DANTE, INPUT_AUDIO_SOURCE_ANALOG]

The input selection of audio. Supported options: IP (INPUT_AUDIO_SOURCE_IP), Hdmi(I INPUT_AUDIO_SOURCE_HDMI), Dante (INPUT_AUDIO_SOURCE_DANTE), Analog (INPUT_AUDIO_SOURCE_ANALOG)

“inputselectionIpGroup”: "INPUT_IP_GROUP1" [INPUT_IP_GROUP2]

Stream type for video IP input. Supported options: JXS(INPUT_IP_GROUP1), H.26x(INPUT_IP_GROUP2)

www.plexusAV.com
sales@plexusav.com
+1.605.9780.4800

The response includes all available streams for the control system to filter as needed. For example, to process the JXS stream in a Decoder, filter the streams to meet the requirements of this environment.

First, filter the streams belonging to the Decoder based on the following attributes:

- nmosdevicestreamType = SENDER
- nmosdevicestreamTransportMode = IP_STREAM_MODE_MULTICAST

Second, If JXS switching is needed, filter out streams from the results above based on the following attributes:

- nmosdevicestreamChannel = JPEGXS

If H.26x switching is needed, then filter out streams on the following attributes:

- nmosdevicestreamChannel = VCU

www.plexusAV.com
sales@plexusav.com
+1.605.9780.4800

API Command Examples

Switch a decoder to a new stream

Get the stream list, input selection, and usb server.

Connecting a Decoder to a Stream takes 3 steps:

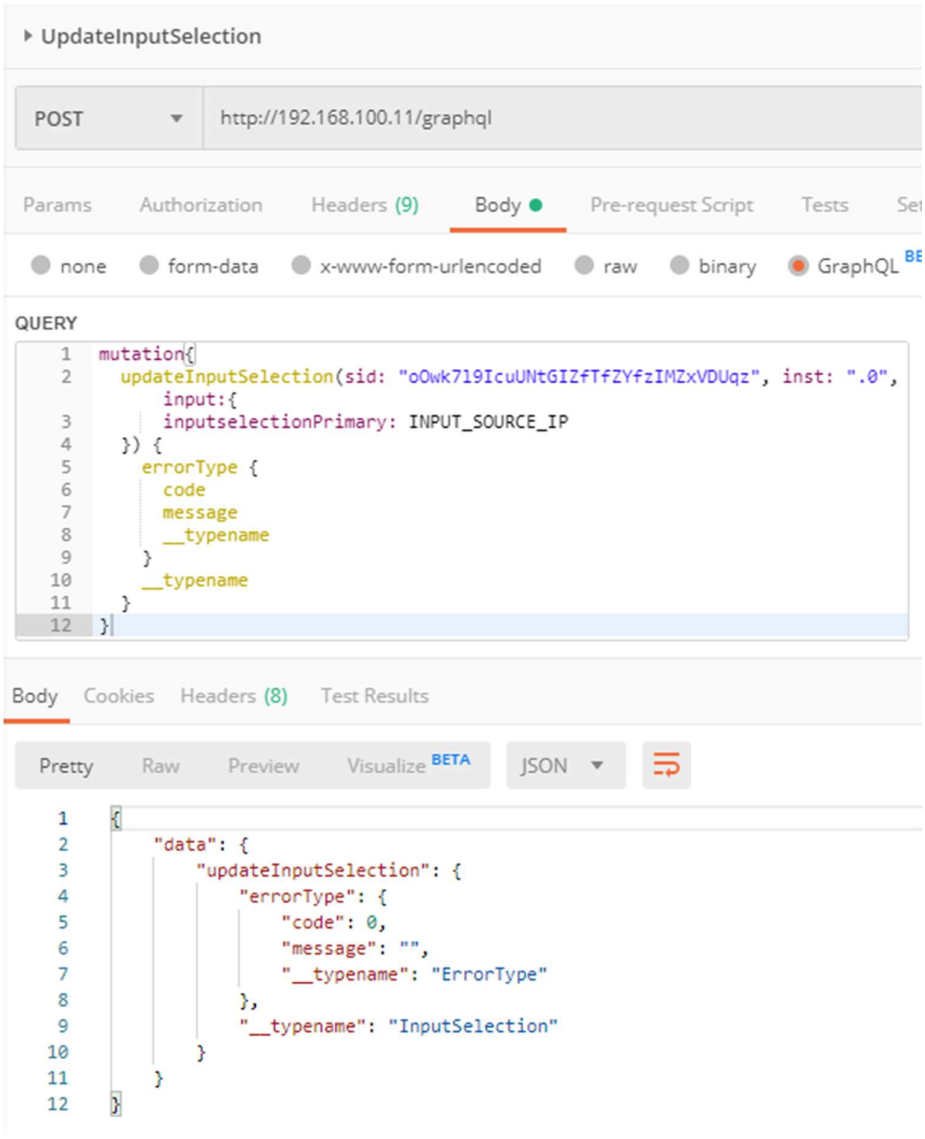
1. **Set the Decoder IP Input Source**
   Make sure that the Decoder is NOT set to the local input (HDMI or USB-c)
2. **Enable the Decoder Input Stream**
   Use the Stream Inst. to connect
3. **Update the Decoder Stream Mode**
   IPMX uses 3 essence streams: audio, video and ancillary data. We need to tell the decoder which one to connect to. Furthermore, we need to set the Decoder input to either JPEG-XS or H.26x.

www.plexusAV.com
sales@plexusav.com
+1.605.9780.4800

1- Set the Decoder IP Input Source

Verify the inputselectionPrimary returned in the first step is INPUT_SOURCE_IP. If not, switch it to INPUT_SOURCE_IP.

=> Please note that we are using Inst. ".0".



www.plexusAV.com
sales@plexusav.com
+1.605.9780.4800

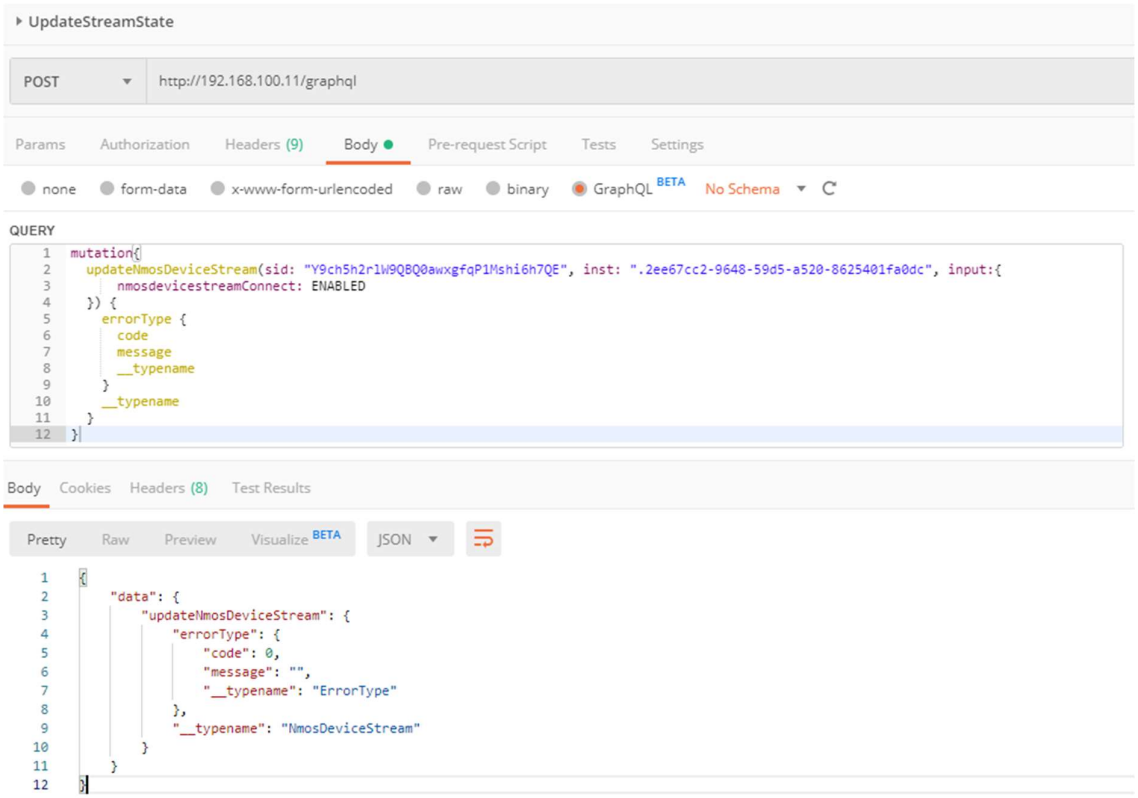
Code:

```
mutation {
  updateInputSelection(
    sid: "The login SID ",
    inst: ".0",
    input: {
      inputselectionPrimary: INPUT_SOURCE_IP
    }
  ) {
    errorType {
      code
      message
      __typename
    }
    __typename
  }
}
```

www.plexusAV.com
sales@plexusav.com
+1.605.9780.4800

2- Enable the Decoder Input Stream

In the section “Discovering IPMX Streams”, we already identified the “stream Inst” to use. Now, we use this to issue the request.



Code:

```
mutation {
  updateNmosDeviceStream(
    sid: "The login SID",
    inst: "please enter your stream inst here",
    input: {
      nmosdevicestreamConnect: ENABLED
    }
  ) {
    errorType {
      code
      message
      __typename
    }
    __typename
  }
}
```

www.plexusAV.com
sales@plexusav.com
+1.605.9780.4800

3- Update the Decoder Stream Mode

For each stream, there are three IPMX essence streams that need to be handled: audio, video and ancillary data. For updating the exact IPMXs, we need to check the inst attribute of the IPMX.

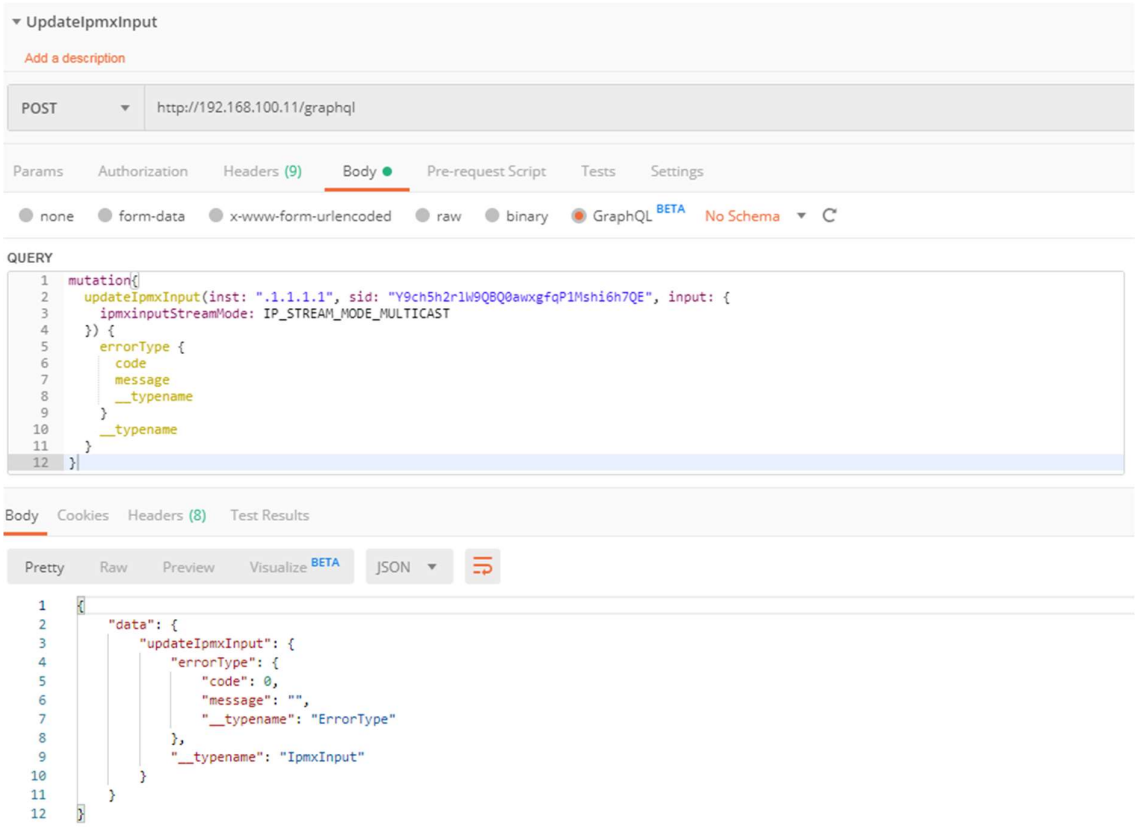
The “updateIpmxInput inst” attribute consists of 4 parts separated by dots, for example, ”.1.1.1.1”. The first value is used to determine whether it is JXS or H.264, and the third value is used to determine the essence stream (video, audio, or ancillary data).

“.1.x.x.x” = JXS
“.2.x.x.x” = H.26x

“.x.x.1.x” = Video Essence Stream
“.x.x.2.x” = Audio Essence Stream
“.x.x.3.x” = Ancillary Data Essence Stream

Let's say we want to switch to a JXS stream (video, audio and data), we need these insts: “.1.1.1.1”, “.1.1.2.1” and “.1.1.3.1”. So, we need to send 3 commands.

For switching to a H.26x stream (video, audio and data), we use these insts: “.2.1.1.1”, “.2.1.2.1” and “.2.1.3.1”.



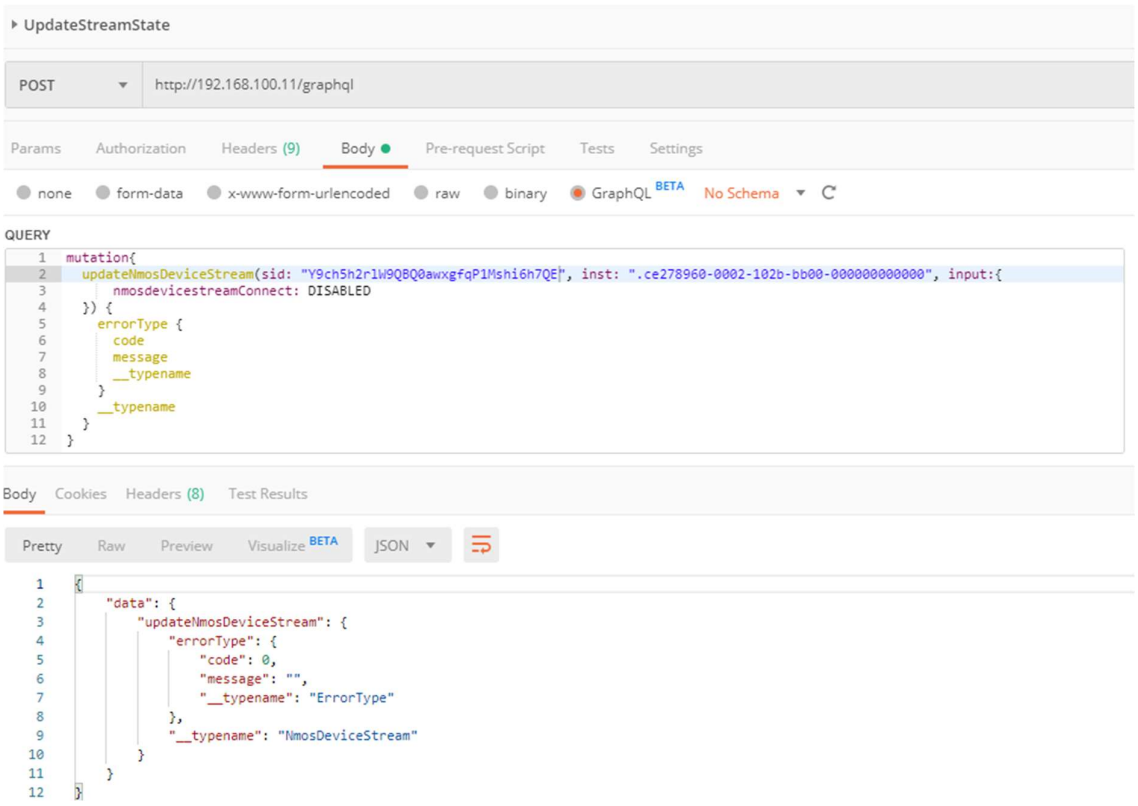
Code:

```
mutation {
  updateIpmxInput(
    inst: " please enter three inst separately",
    sid: "The login SID",
    input: {
      ipmxinputStreamMode: IP_STREAM_MODE_MULTICAST
    }
  ) {
    errorType {
      code
      message
    }
  }
}
```

www.plexusAV.com
sales@plexusav.com
+1.605.9780.4800

Disconnect a stream

Disconnecting a stream takes one request; set the nmosdevicestreamConnect to DISABLED.

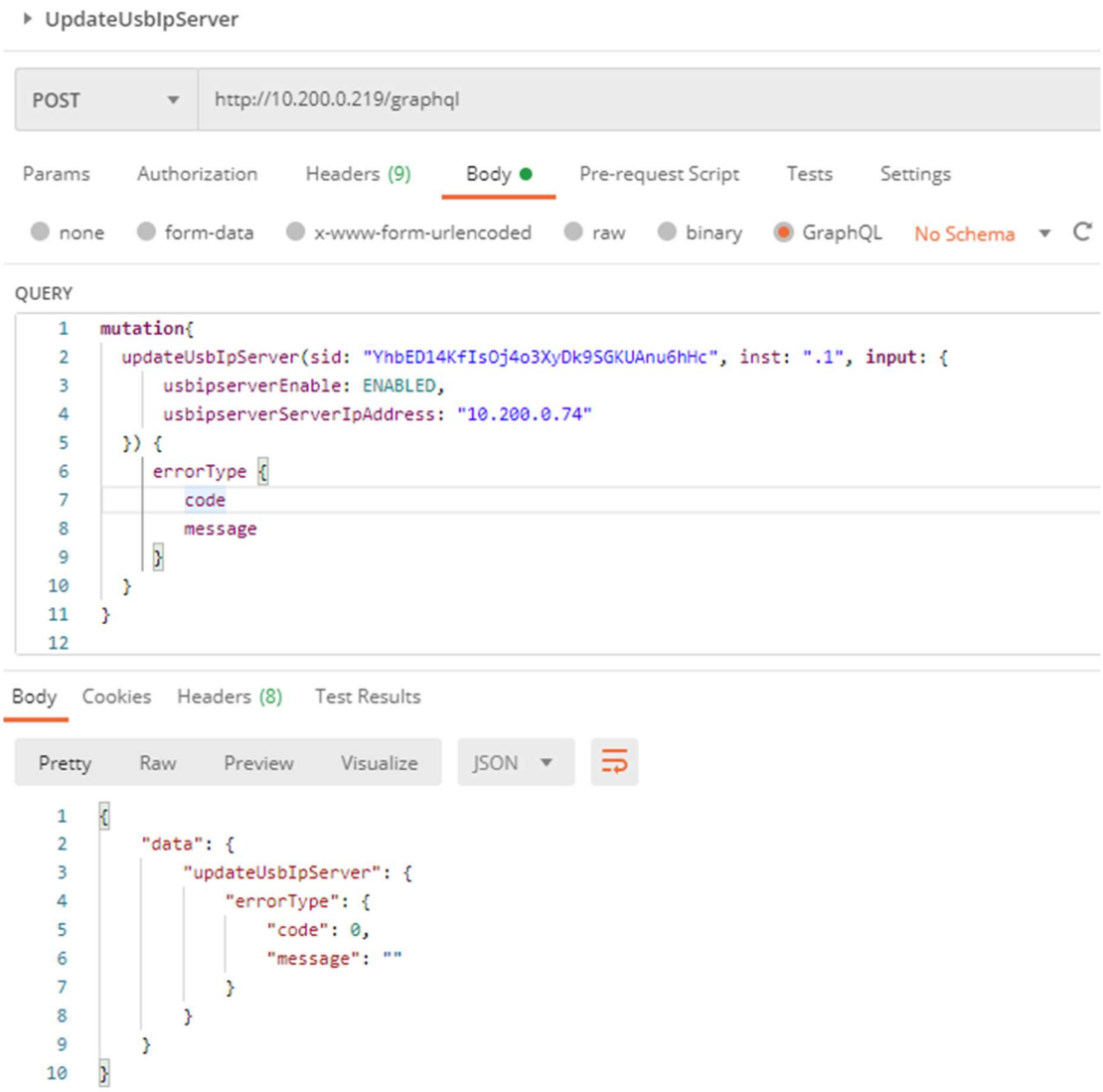


Code:

```
mutation {
  updateNmosDeviceStream(
    sid: "The login SID",
    inst: "please enter your stream inst here",
    input: {
      nmosdevicestreamConnect: DISABLED
    }
  ) {
    errorType {
      code
      message
      __typename
    }
    __typename
  }
}
```

KVM switching

KVM routing requires doing AV routing first as we did on updateIpmxInput, then establish a USB IP connection shown in the example below. Fill out the target USB server (Encoder) IP address usbipserverServerIpAddress



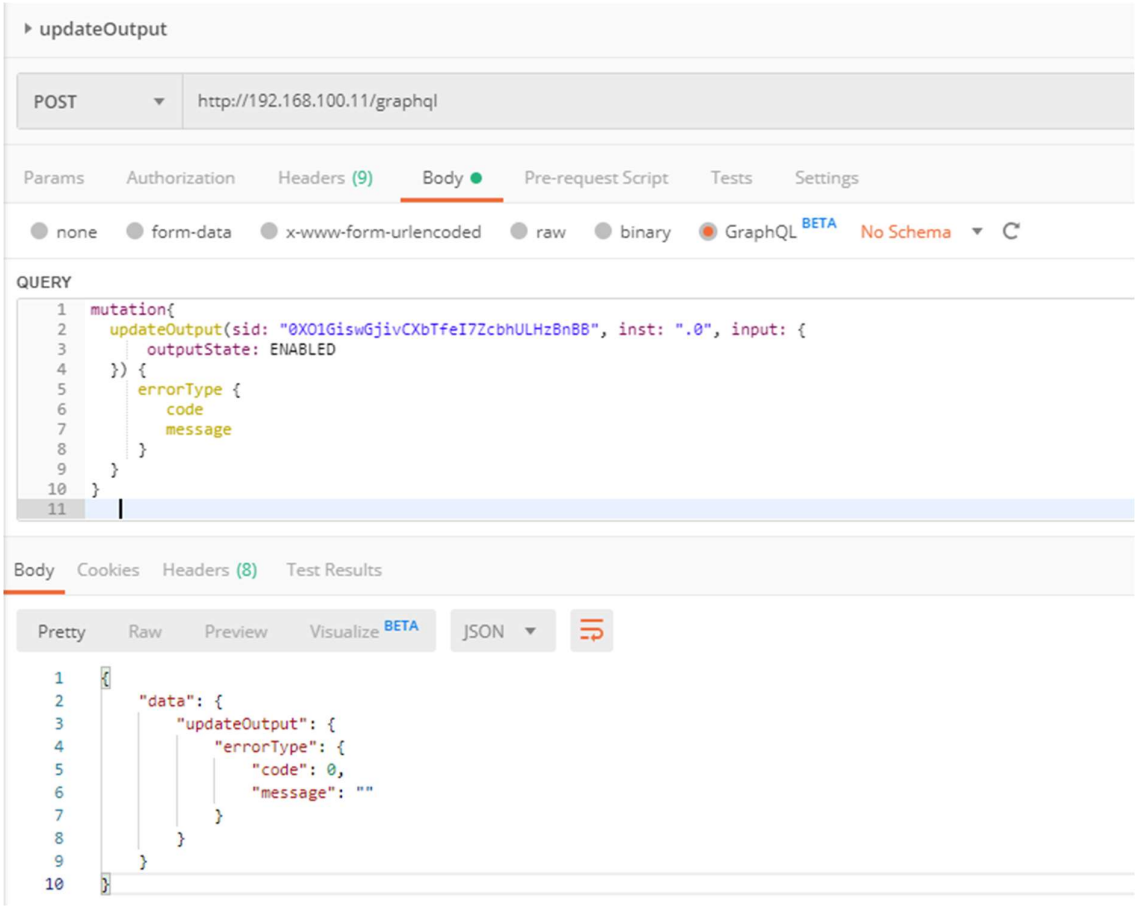
Code:
Code:

```
mutation {
  updateUsbIpServer(
    sid: "The login SID",
    inst: ".1",
    input: {
```

```
    usbipserverEnable: ENABLED,
    usbipserverServerIpAddress: " please enter the nmosdevicestreamUnicastIp  of stream"
  }
) {
  errorType {
    code
    message
  }
}
```

Start / Stop Streaming Services

Start/Stop Streaming service toggles the IPMX transceiver to enter an idle state with minimum bandwidth. Start or stop IP streams transmitting at Encoder and receiving at Decoder, set outputState = ENABLE/DISABLE



Code:

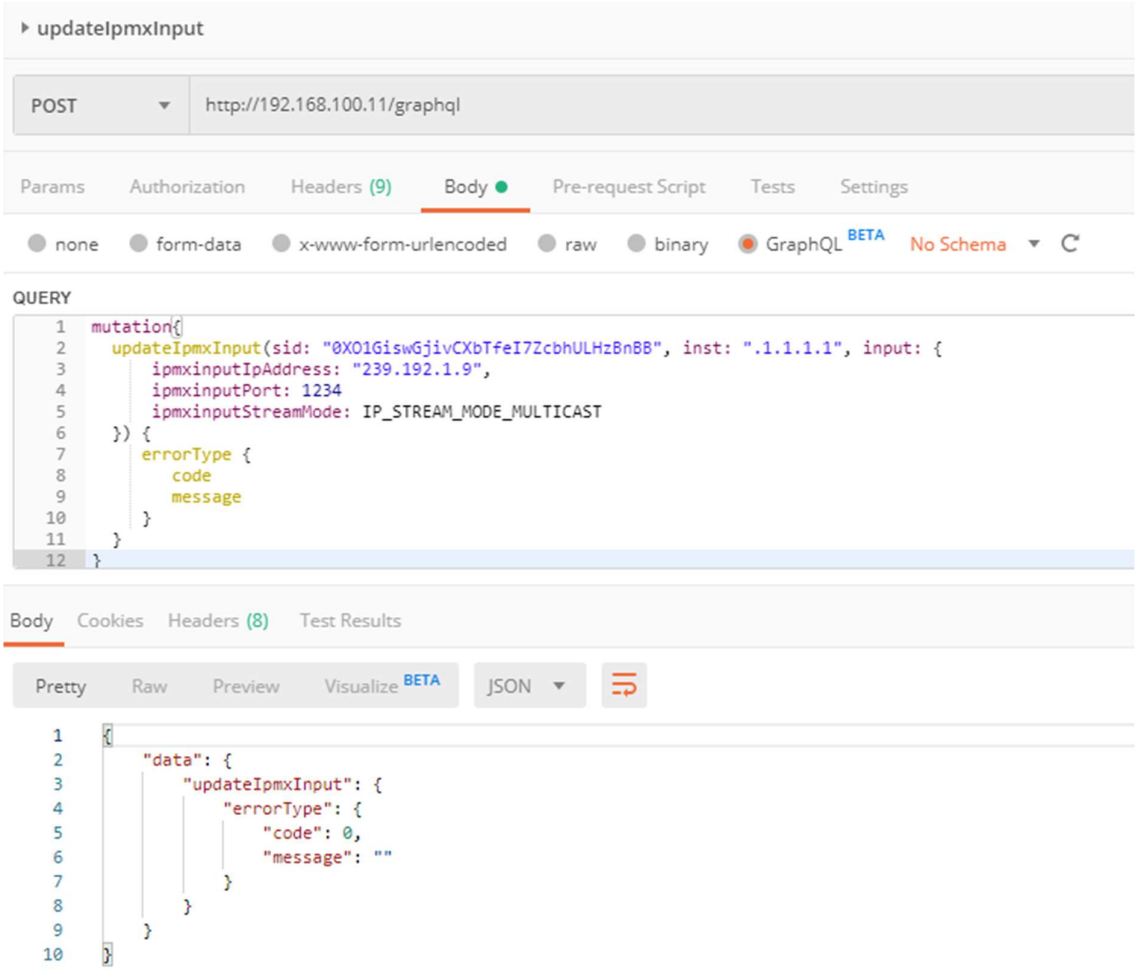
```
mutation {
  updateOutput(
    sid: "The login SID",
    inst: ".0",
```

```
    input: {
      outputState: ENABLED
    }
  ) {
    errorType {
      code
      message
    }
  }
}
```

Change audio and video input

The IPMX *inst* consists of 4 numbers separated by dots. The *inst* of video, audio, and data for JXS are ".1.1.1.1", ".1.1.2.1", ".1.1.3.1", respectively, and for H.264 are ".2.1.1.1", ".2.1.2.1", ".2.1.3.1", respectively.

For a Decoder, specify 'inst' as '.1.1.1.1' to receive the JXS video stream, '.1.1.2.1' to receive the primary PCM audio stream, and '.1.1.3.1' to receive the data stream



www.plexusAV.com
sales@plexusav.com
+1.605.9780.4800

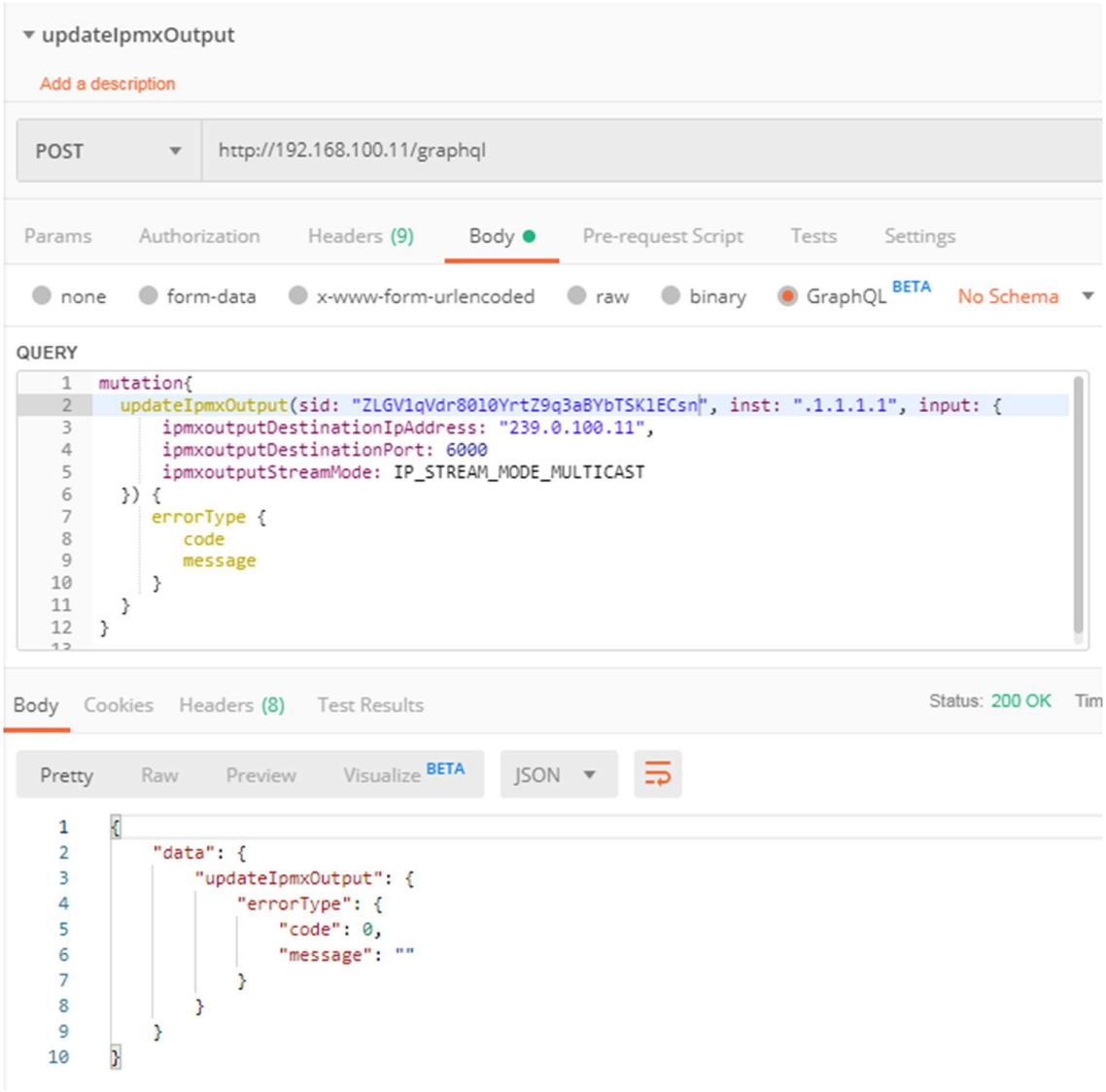
www.plexusAV.com
sales@plexusav.com
+1.605.9780.4800

Code:

```
mutation {
  updateIpmxInput(
    sid: "The login SID",
    inst: ".1.1.1.1",
    input: {
      ipmxinputIpAddress: "239.192.1.9",
      ipmxinputPort: 1234,
      ipmxinputStreamMode: IP_STREAM_MODE_MULTICAST
    }
  ) {
    errorType {
      code
      message
    }
  }
}
```

For an Encoder, enter unique multicast addresses and/or ports for the video(inst is '.1.1.1.1'), audio(inst is '.1.1.2.1') and ancillary data(inst is '.1.1.3.1') for the JXS essence streams.

For the H.26x essence stream outputs, set the multicast addresses and/or ports for inst '.2.1.1.1' for video, '.2.1.2.1' for audio, '.2.1.3.1' for ancillary data.



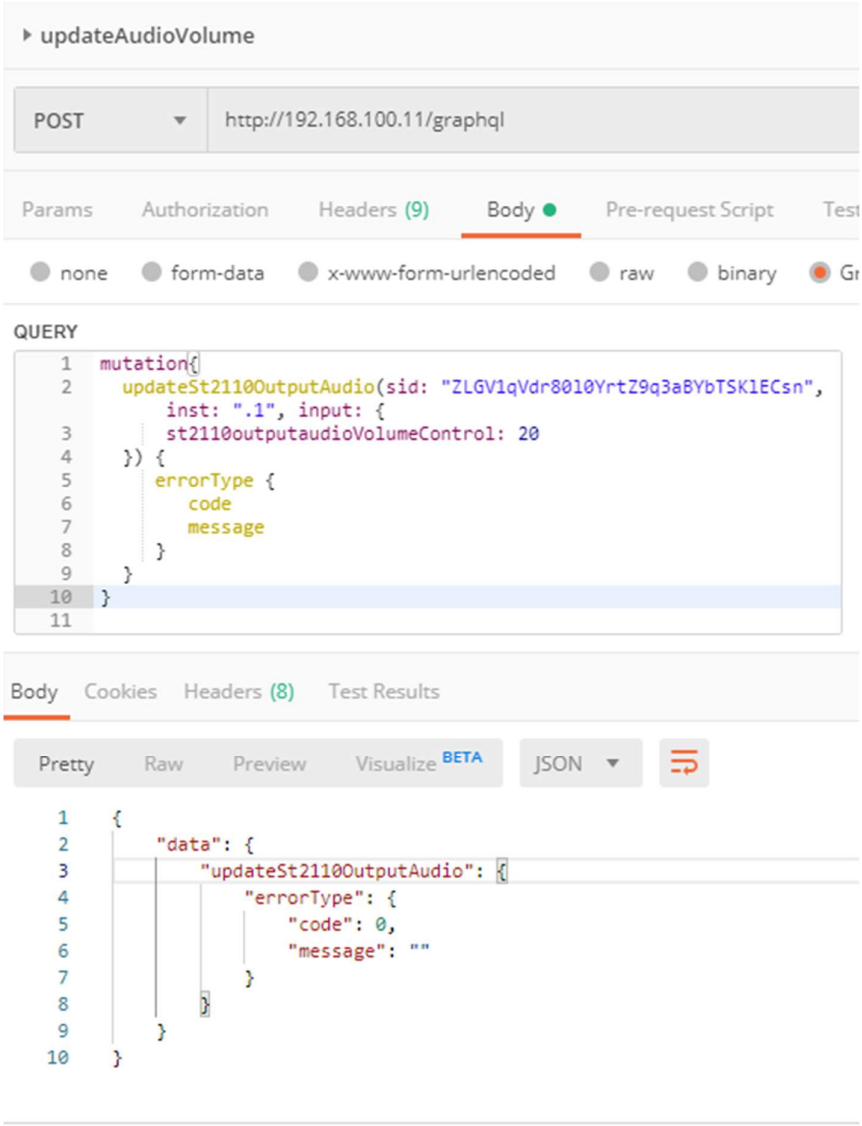
Code:

```
mutation {
  updateIpmxOutput(
    sid: "ZLGV1qVdr80l0YrtZ9q3aBYbTSKlECsn",
    inst: ".1.1.1.1",
    input: {
      ipmxoutputDestinationIpAddress: "239.0.100.11",
      ipmxoutputDestinationPort: 6000,
      ipmxoutputStreamMode: IP_STREAM_MODE_MULTICAST
    }
  ) {
    errorType {
      code
      message
    }
  }
}
```

www.plexusAV.com
sales@plexusav.com
+1.605.9780.4800

Audio volume and mute

For an Encoder outputting PCM audio, the audio output volume is controlled by adjusting the value of st2110outputaudioVolumeControl:

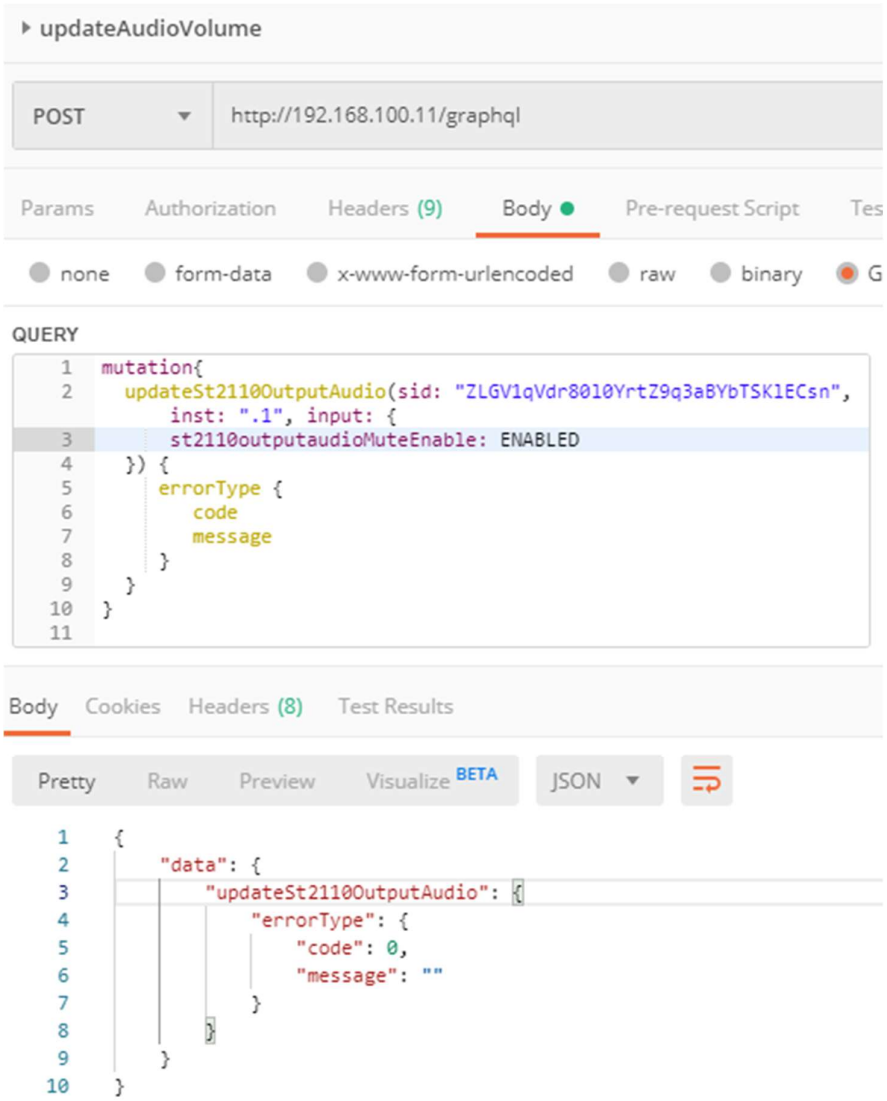


www.plexusAV.com
sales@plexusav.com
+1.605.9780.4800

Code:

```
mutation {
  updateSt2110OutputAudio(
    sid: " The login SID ",
    inst: ".1",
    input: {
      st2110outputaudioVolumeControl: 20
    }
  ) {
    errorType {
      code
      message
    }
  }
```

To mute the PCM audio output, set st2110outputaudioMuteEnable to ENABLED:



www.plexusAV.com
sales@plexusav.com
+1.605.9780.4800

Code:

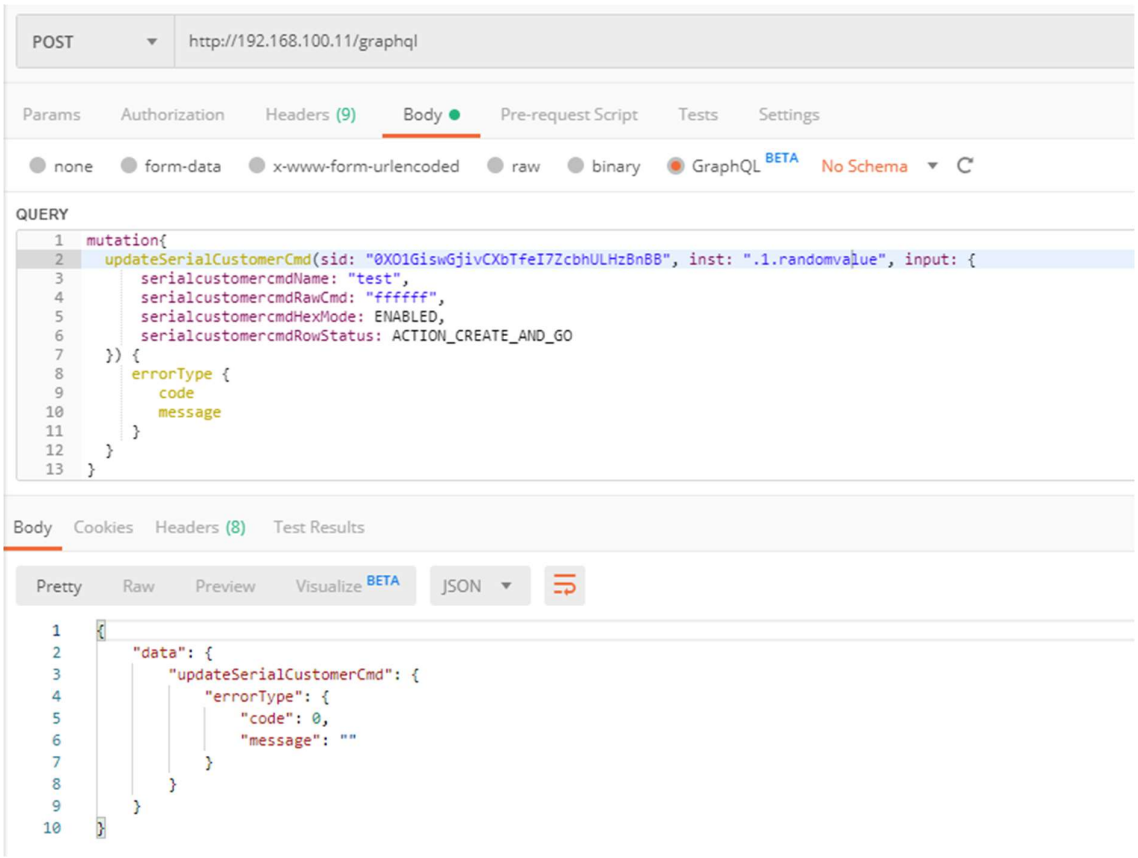
```
mutation {
  updateSt2110OutputAudio(
    sid: " The login SID ",
    inst: ".1",
    input: {
      st2110outputaudioMuteEnable: ENABLED
    }
  ) {
    errorType {
      code
      message
    }
  }
}
```

Send RS232 command

Use the following to add a serial command:

- inst: Consists of a fixed character ".1." and a unique number for the command, such as a timestamp.
- serialcustomercmdName: The name of the command.
- serialcustomercmdRawCmd: The value or content of the command.
- serialcustomercmdHexMode: Toggle to enable/disable HEX mode. ENABLED means turned on, DISABLED means turned off.
- serialcustomercmdRowStatus: The action to execute command, fixed to ACTION_CREATE_AND_GO.

Once the query is sent, the RS232 command is visible in the RS232 command list.



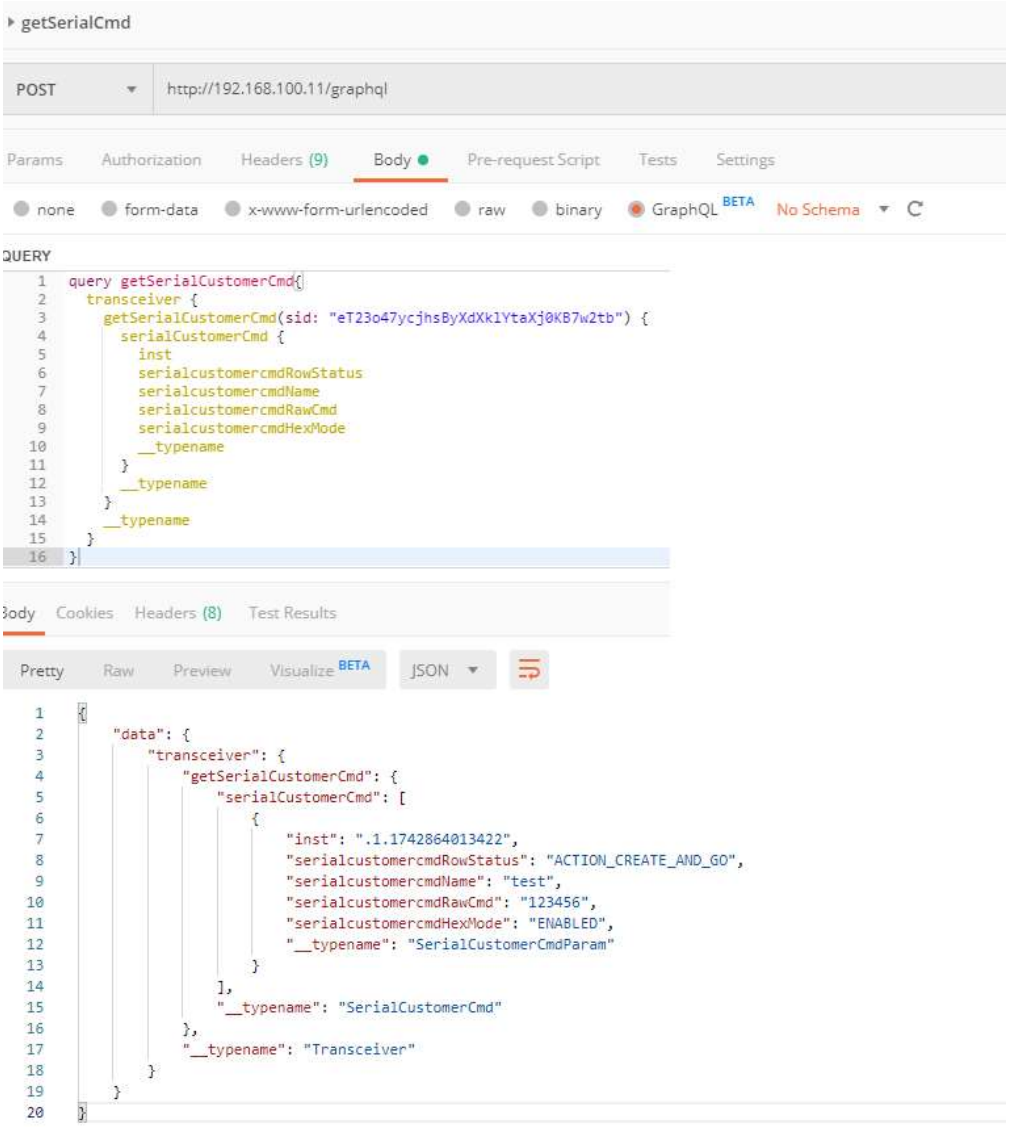
www.plexusAV.com
sales@plexusav.com
+1.605.9780.4800

Code:

```
mutation {
  updateSerialCustomerCmd(
    sid: "The login SID",
    inst: "please enter a unique inst",
    input: {
      serialcustomercmdName: "please enter the title of the command",
      serialcustomercmdRawCmd: "please enter the value of the command",
      serialcustomercmdHexMode: ENABLED,
      serialcustomercmdRowStatus: ACTION_CREATE_AND_GO
    }
  ) {
    errorType {
      code
      message
    }
  }
}
```

www.plexusAV.com
sales@plexusav.com
+1.605.9780.4800

Retrieve the RS232 commands list using serialControlCmd.

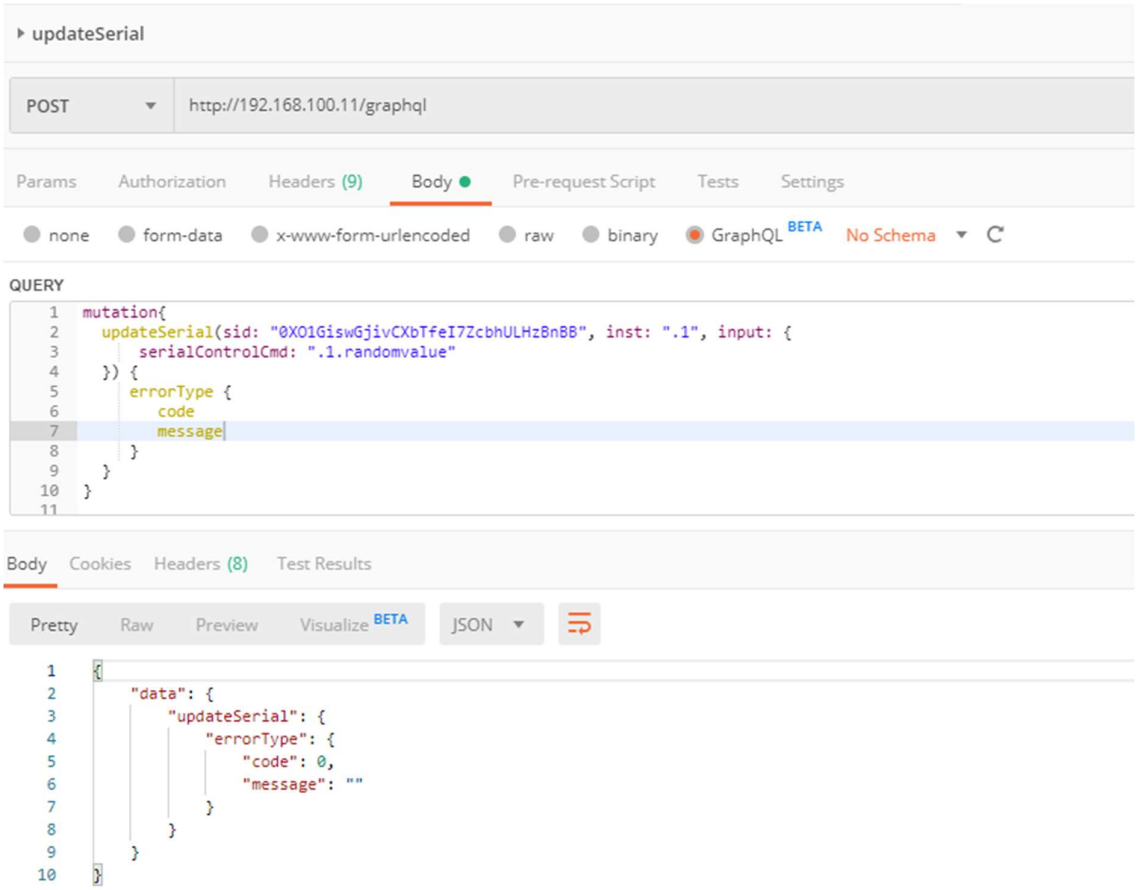


www.plexusAV.com
sales@plexusav.com
+1.605.9780.4800

Code:

```
query getSerialCustomerCmd {
  transceiver {
    getSerialCustomerCmd(
      sid: "The login SID"
  ) {
      serialCustomerCmd {
        inst
        serialcustomercmdRowStatus
        serialcustomercmdName
        serialcustomercmdRawCmd
        serialcustomercmdHexMode
        __typename
      }
      __typename
    }
    __typename
  }
}
```

Apply the RS232 command using the unique *inst* assigned to the command.



www.plexusAV.com
sales@plexusav.com
+1.605.9780.4800

Code:

```
mutation {
  updateSerial(
    sid: "The login SID",
    inst: ".1",
    input: {
      serialControlCmd: ".1.randomValue"
    }
  ) {
    errorType {
      code
      message
    }
  }
}
```

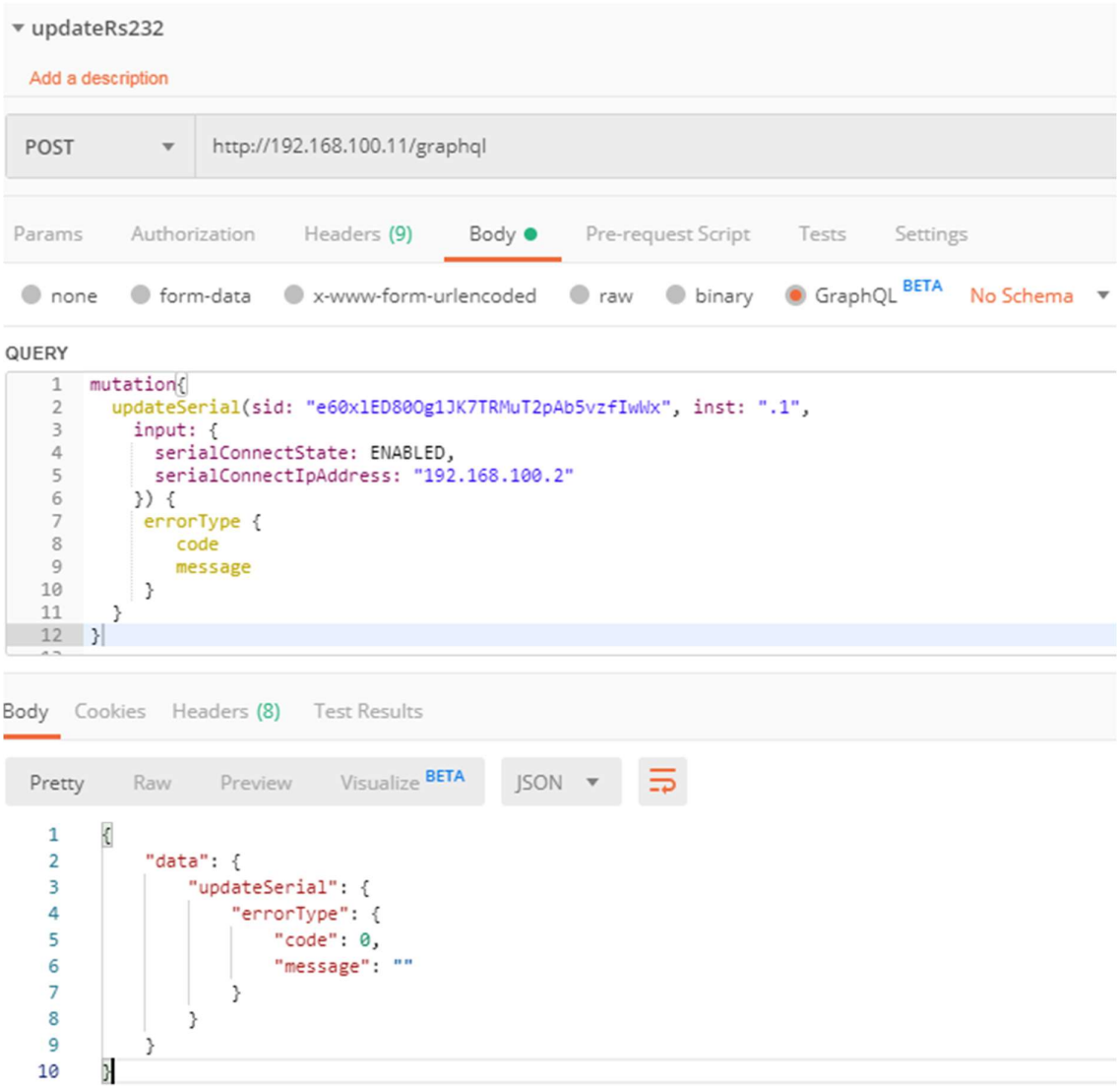
Data (RS232/IR/CEC) switching

Data routing is added via NMOS connection. Please note that IR and CEC over IP are not yet supported.

1. Query the nmos list to get the nmosdeviceIp which is the Encoder's IP for the Decoder to connect to. (See the chapter - Discovering IPMX devices)

2. Set the Decoder IP Input Source (See the chapter - Set the Decoder IP Input Source)

3. Establish a RS232 over IP connection. Set serialConnectState to ENABLED and pass nmosdeviceIp to serialConnectIpAddress.



www.plexusAV.com
sales@plexusav.com
+1.605.9780.4800

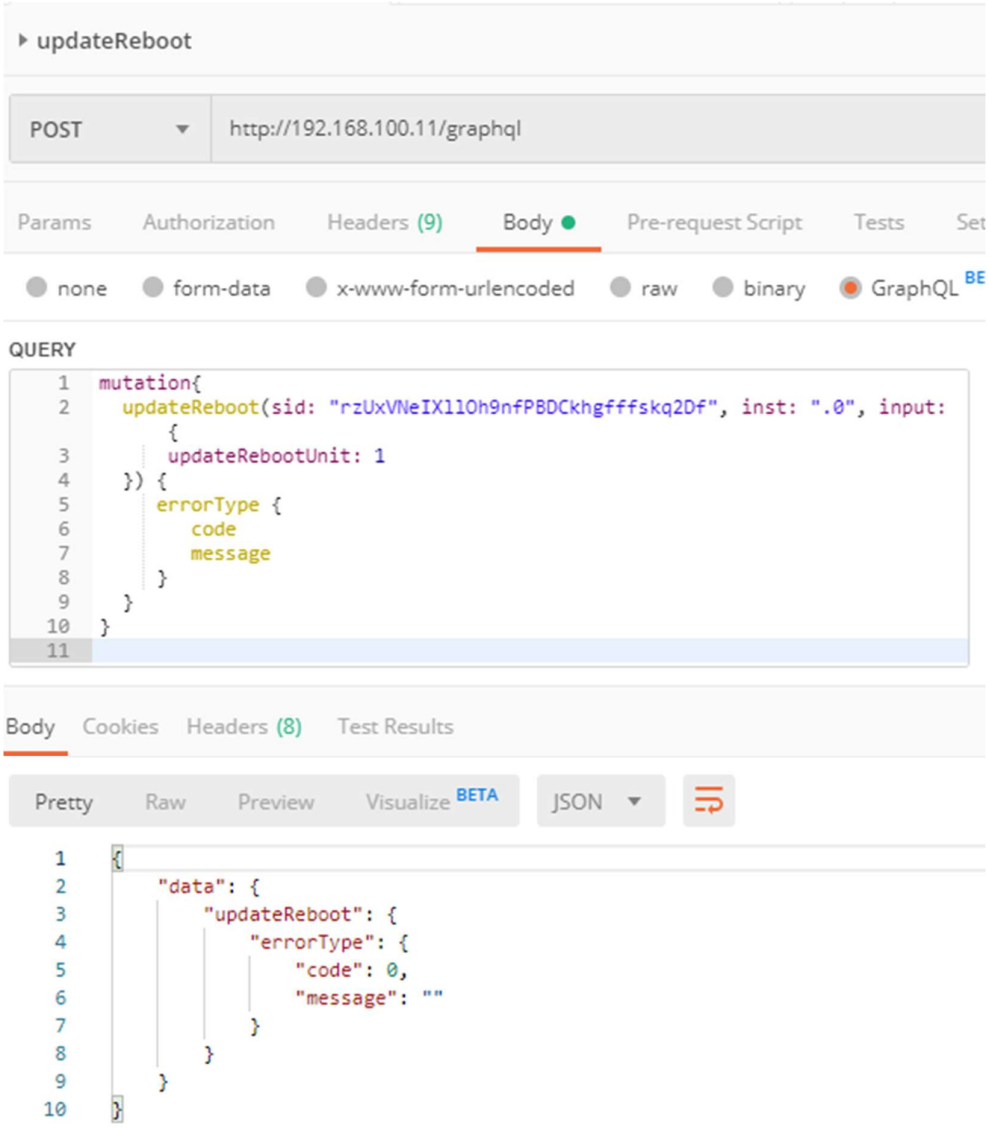
Code:

```
mutation {
  updateSerial(
    sid: "The login SID",
    inst: ".1",
    input: {
      serialConnectState: ENABLED,
      serialConnectIpAddress: "The device IP to connect to"
    }
  ) {
    errorType {
      code
      message
    }
  }
}
```

www.plexusAV.com
sales@plexusav.com
+1.605.9780.4800

Reboot Unit

A unit may be rebooted using the graphQL API.



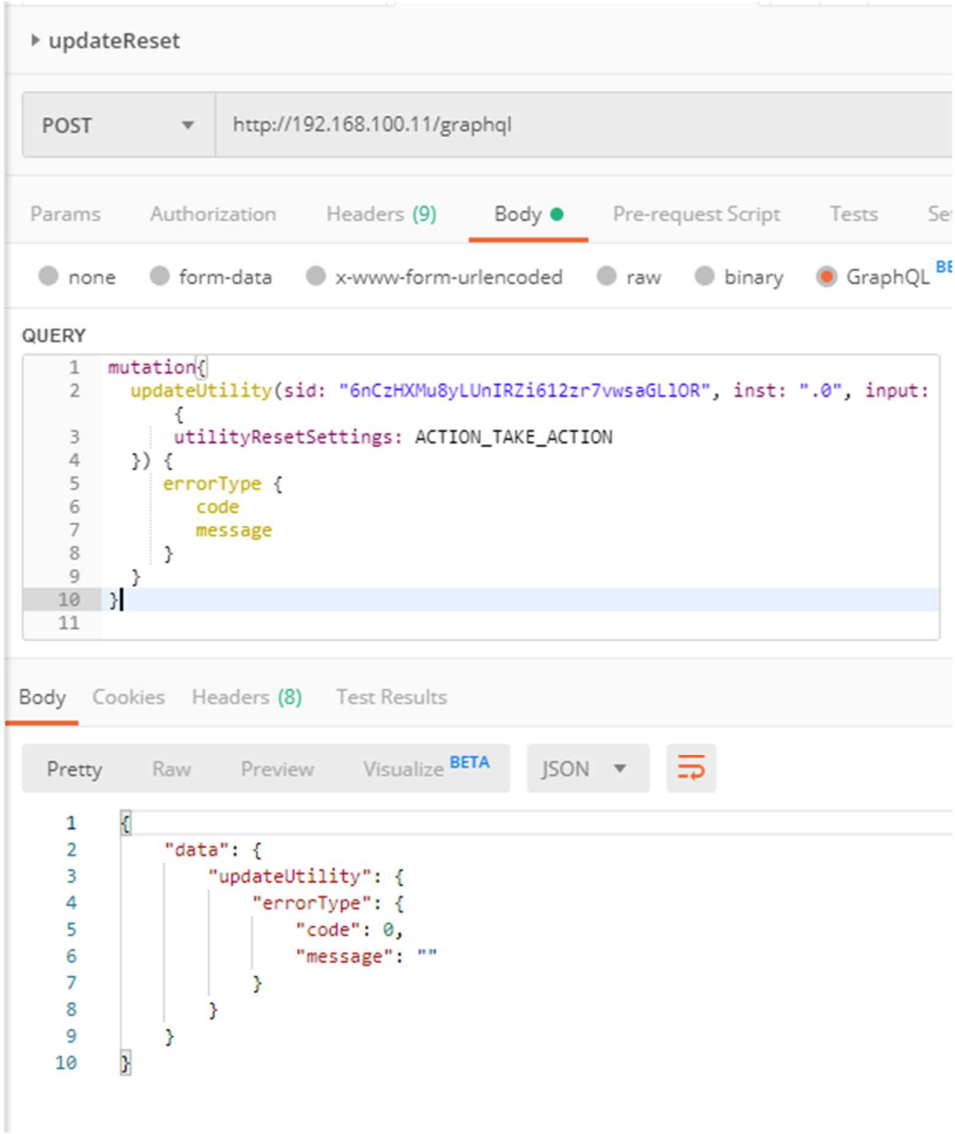
Code:

```
mutation {
  updateReboot(sid: "The login SID", inst: ".0", input: {
    updateRebootUnit: 1
  }) {
    errorType {
      code
      message
    }
  }
}
```

www.plexusAV.com
sales@plexusav.com
+1.605.9780.4800

Factory Default Unit

A unit can return to the factory default settings using graphQL. After a factory reset, the unit reboots into decoder mode as part of the default configuration.



www.plexusAV.com
sales@plexusav.com
+1.605.9780.4800

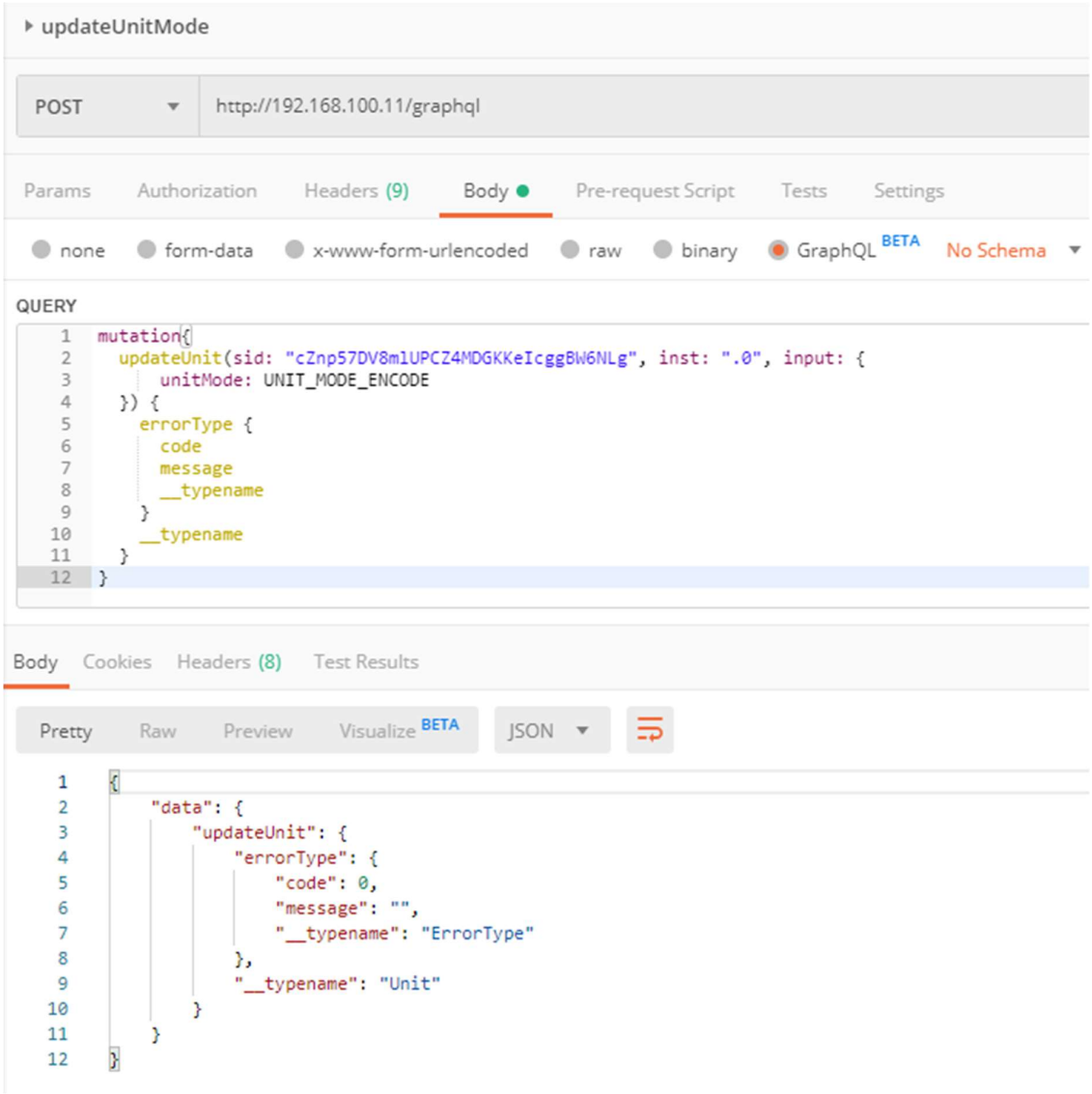
Code:

```
mutation {
  updateUtility(
    sid: " The login SID ",
    inst: ".0",
    input: {
      utilityResetSettings: ACTION_TAKE_ACTION
    }
  ) {
    errorType {
      code
      message
    }
  }
}
```

www.plexusAV.com
sales@plexusav.com
+1.605.9780.4800

Change Encode and Decode mode

Set unitMode to UNIT_MODE_ENCODE to switch to encode mode. Set it to UNIT_MODE_DECODE to switch to decode.



www.plexusAV.com
sales@plexusav.com
+1.605.9780.4800

Code:

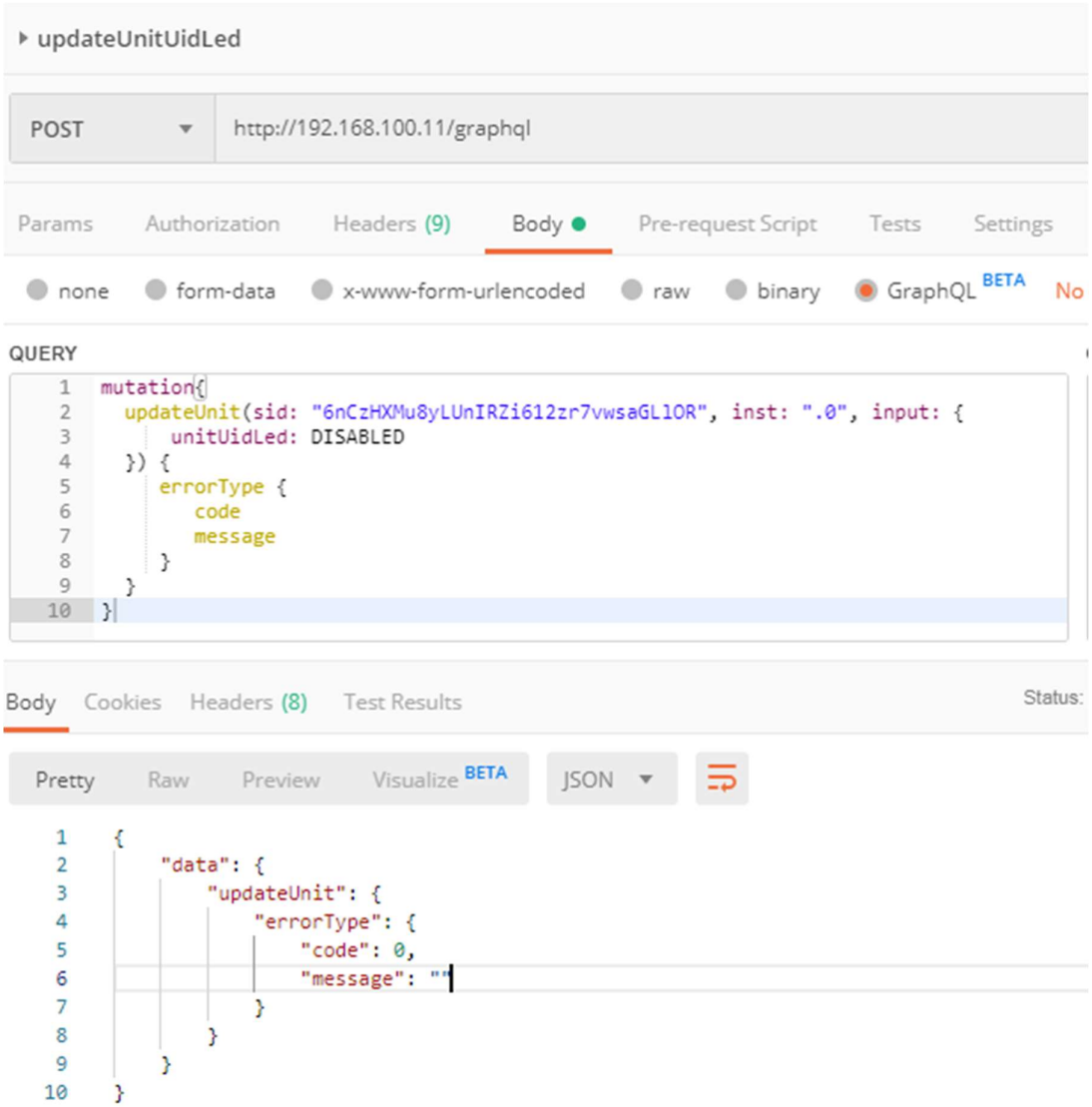
```
mutation {
  updateUnit(
    sid: "The login SID.",
    inst: ".0",
    input: {
      unitMode: UNIT_MODE_ENCODE
   }
  ) {
    errorType {
      code
      message
      __typename
   }
    __typename
  }
}
```

After selecting the appropriate mode, reboot the device to boot into the selected mode. (See chapter – Reboot Unit)

www.plexusAV.com
sales@plexusav.com
+1.605.9780.4800

Unit ID

The Unit ID (UID) helps identify one unit out of many installed units by blinking the status LED on the frontpanel while enabled. Disabling the UID returns the status LED to its previous state and behavior.



www.plexusAV.com
sales@plexusav.com
+1.605.9780.4800

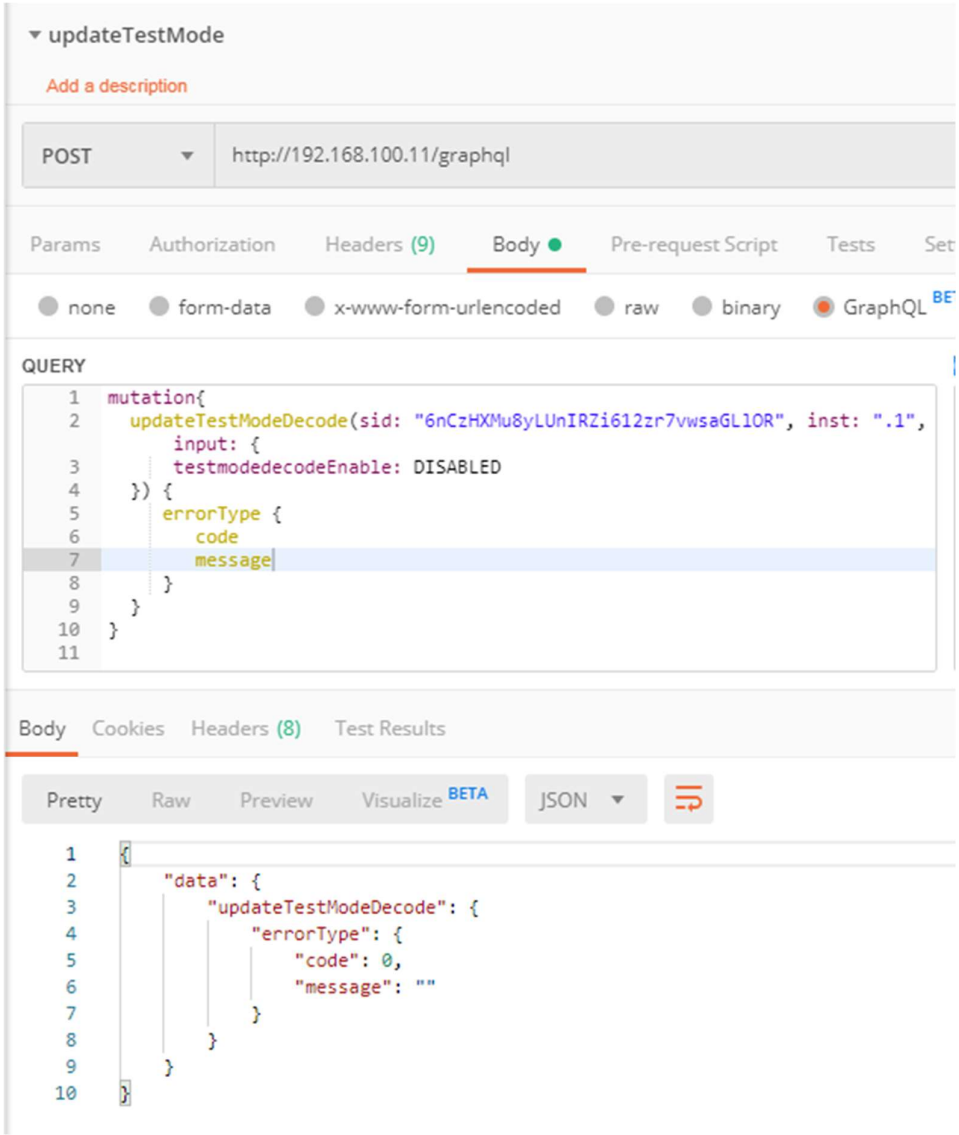
Code:

```
mutation {
  updateUnit(
    sid: " The login SID",
    inst: ".0",
    input: {
      unitUidLed: DISABLED
    }
  ) {
    errorType {
      code
      message
    }
  }
}
```

www.plexusAV.com
sales@plexusav.com
+1.605.9780.4800

Test Mode

The Transceiver can output a color bar test pattern to help verify the video output connection. Set testmodedecodeEnable to ENABLED to turn on Test Mode, set it to DISABLED to return the video output to the previous state.



www.plexusAV.com
sales@plexusav.com
+1.605.9780.4800

Code:

```
mutation {
  updateTestModeDecode(
    sid: " The login SID",
    inst: ".1",
    input: {
      testmodedecodeEnable: DISABLED
    }
  ) {
    errorType {
      code
      message
    }
  }
}
```

HDMI Consumer Electronics Control - CEC

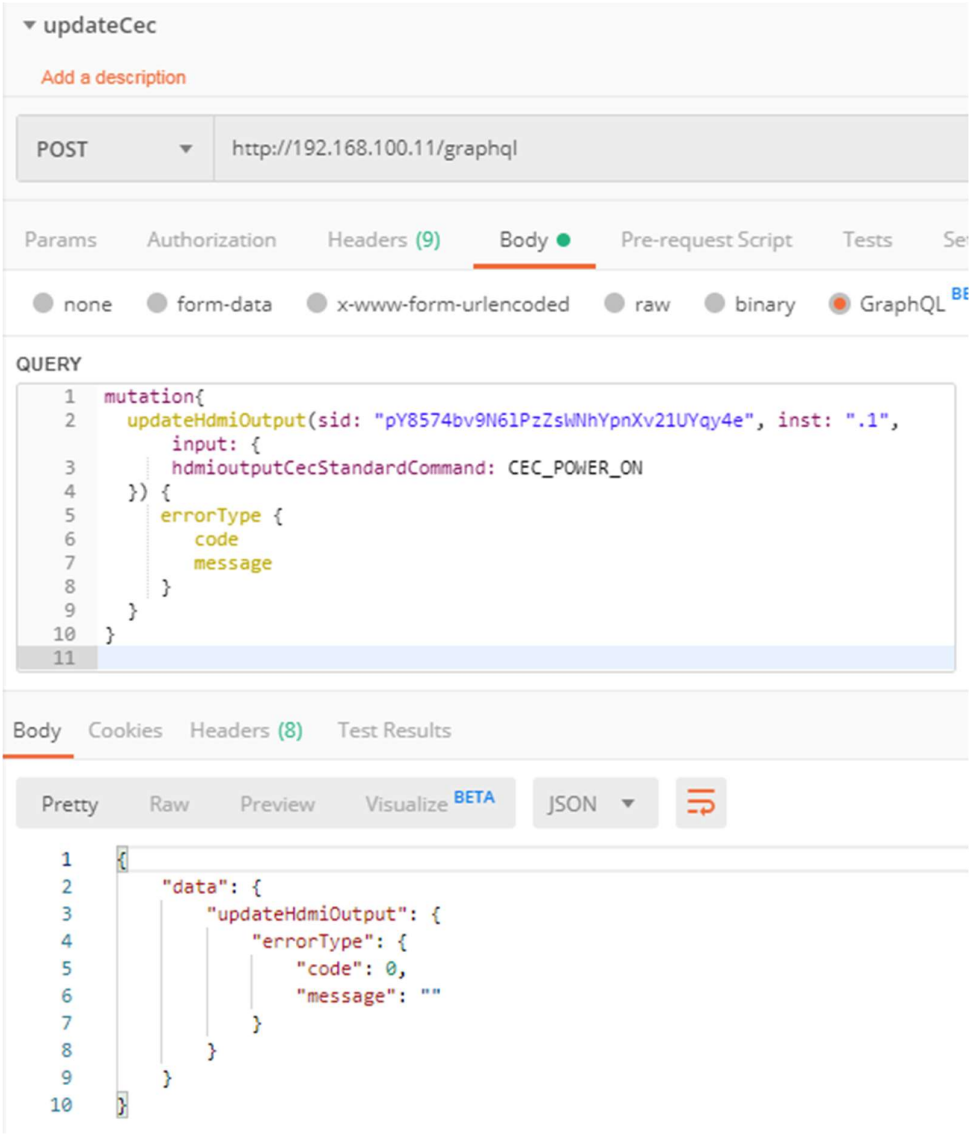
4 common used CEC commands can be sent from P-AVN-4/2 to a display or TV, set these options to hdmioutputCecStandardCommand to turn it on/off, mute/unmute :

CEC_POWER_ON: to turn on the display/TV

CEC_POWER_OFF: to turn off the display/TV

CEC_AUDIO_MUTE: mute

CEC_AUDIO_UNMUTE: unmute



Code:

```
mutation {
  updateHdmiOutput(
    sid: " The login SID",
    inst: ".1",
    input: {
      hdmioutputCecStandardCommand: CEC_POWER_ON
    }
  ) {
    errorType {
      code
      message
    }
  }
}
```

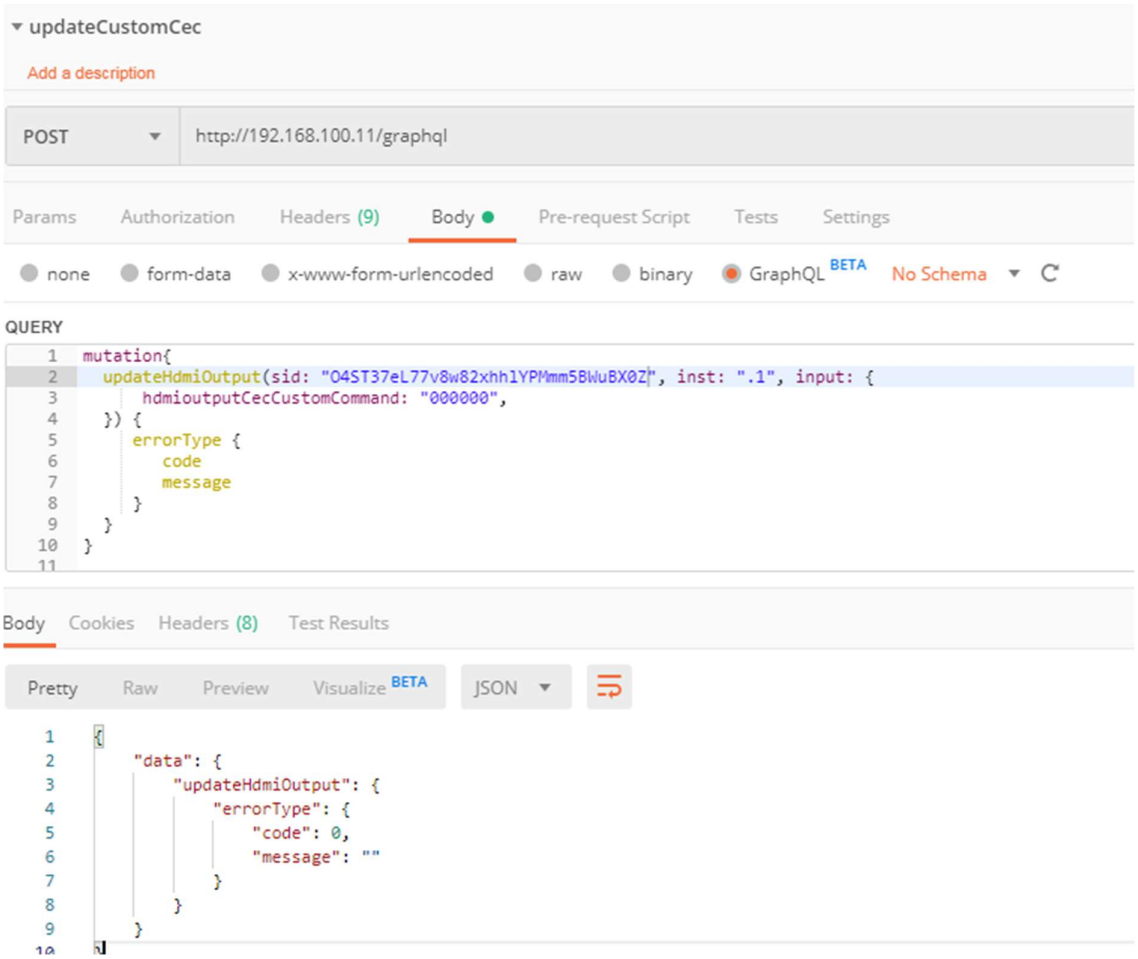
If you send the same command repeatedly, set hdmioutputCecStandardCommandApply to ACTION_TAKE_ACTION

www.plexusAV.com
sales@plexusav.com
+1.605.9780.4800

Code:

```
mutation {
  updateHdmiOutput(
    sid: " The login SID",
    inst: ".1",
    input: {
      hdmioutputCecStandardCommandApply: ACTION_TAKE_ACTION
    }
  ) {
    errorType {
      code
      message
    }
  }
}
```

Customized commands may be sent using hdmioutputCecCustomCommand.



Code:

```
mutation {
 updateHdmiOutput(
   sid: "The login SID",
   inst: ".1",
   input: {
     hdmioutputCecCustomCommand: "customed command here"
   }
 ) {
   errorType {
     code
     message
   }
 }
}
```

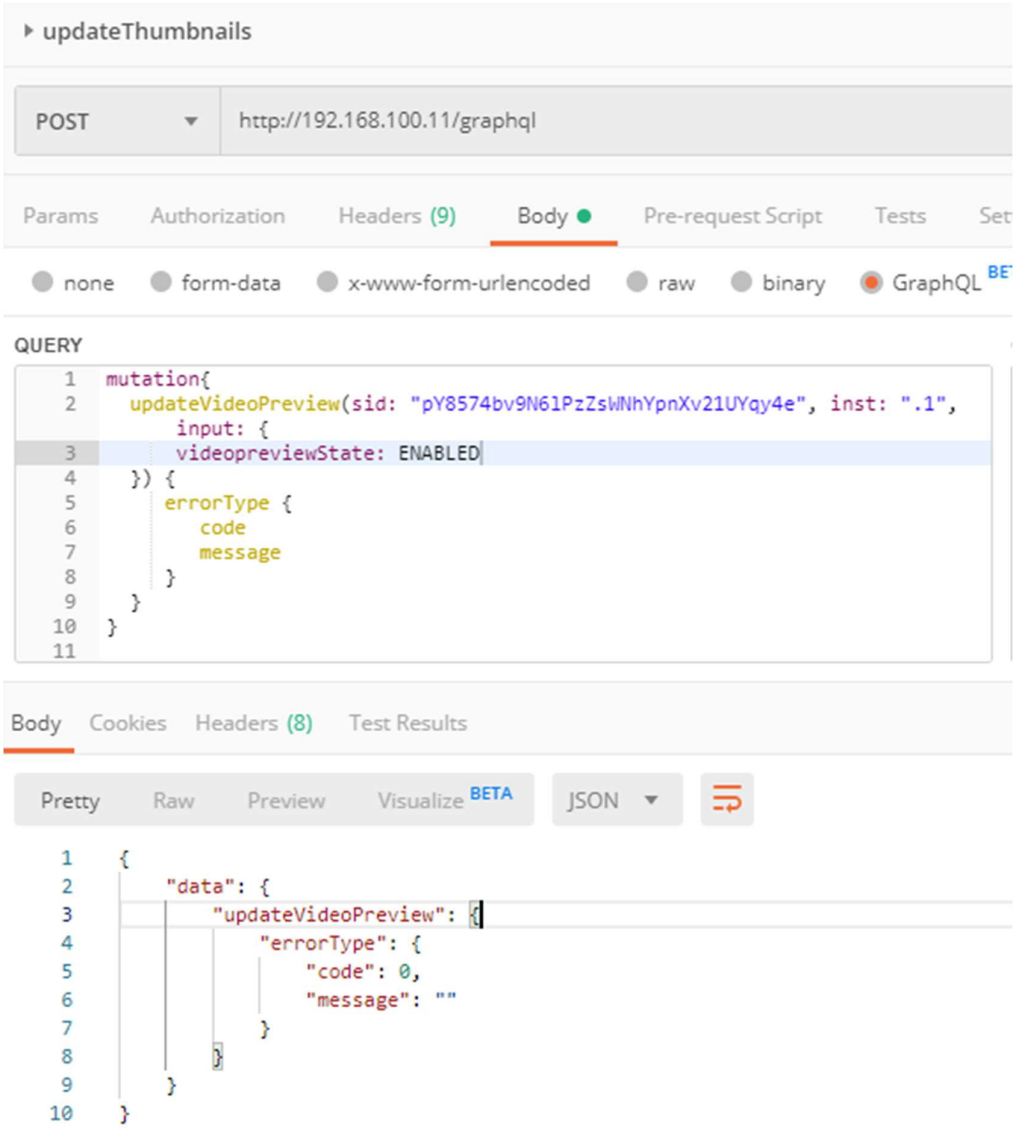
www.plexusAV.com
sales@plexusav.com
+1.605.9780.4800

Additional information

Thumbnail Preview

Control the thumbnail preview using videopreviewState.

- ENABLED – Turn on thumbnail preview.
- DISABLED - Turn off thumbnail preview.



www.plexusAV.com
sales@plexusav.com
+1.605.9780.4800

Code:

```
mutation {
  updateVideoPreview(
    sid: " The login SID",
    inst: ".1",
    input: {
      videopreviewState: ENABLED
    }
  ) {
    errorType {
      code
      message
    }
  }
}
```

The thumbnail may be downloaded using a web browser at the following URL: http://<unit_ip>/file/preview/videopreview.jpeg. For example, “http://192.168.100.11/file/preview/videopreview.jpeg”

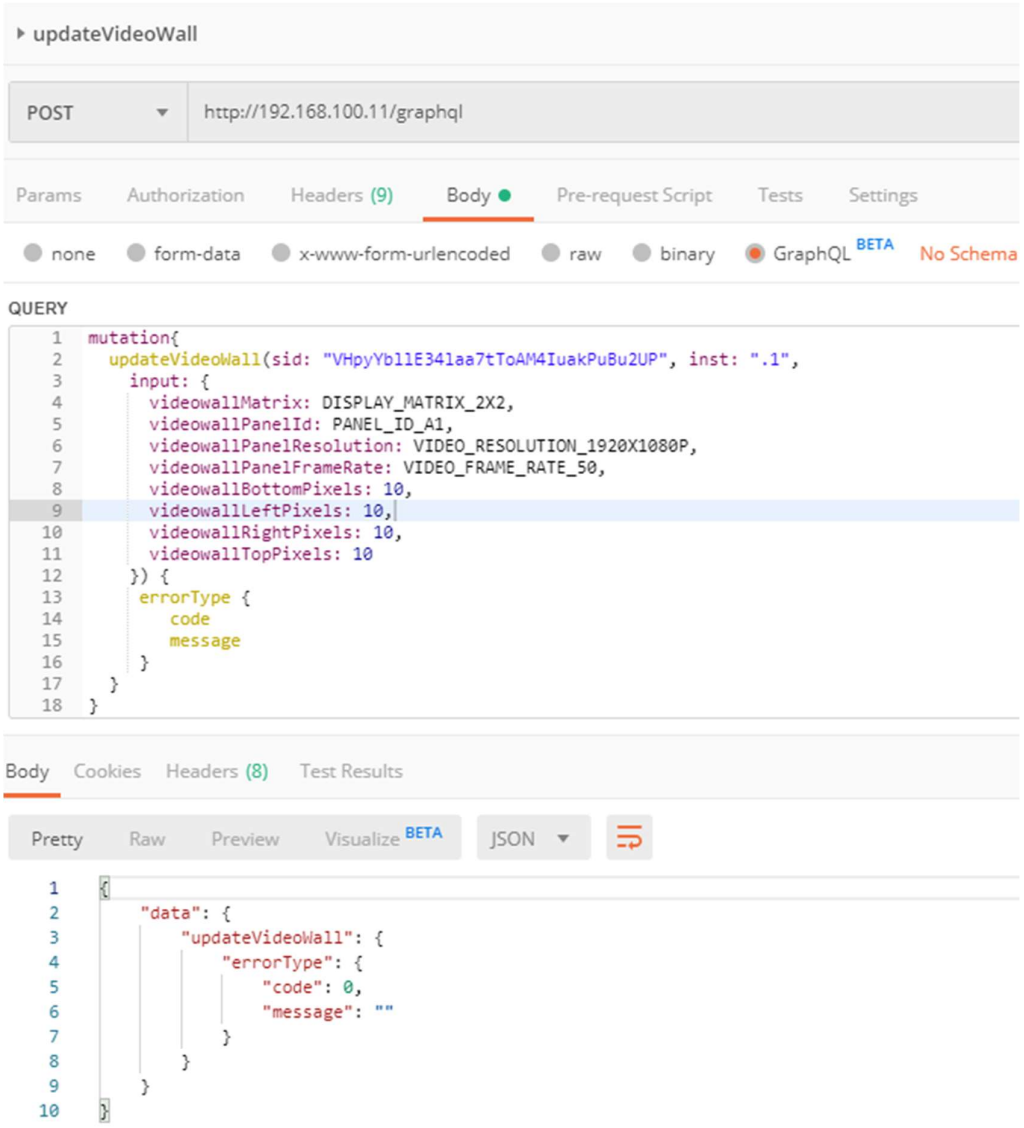
www.plexusAV.com
sales@plexusav.com
+1.605.9780.4800

Videowall

Video content can be formatted to output to a videowall. The default properties are:

- videowallMatrix (Matrix Size): 3 X 3
- videowallPanelId (Panel ID): A1
- videowallPanelResolution (Panel Resolution): 4K UHD 3840 x 2160p
- videowallPanelFrameRate (Panel Frame Rate): 60fps
- videowallTopPixels (Cropping at the top of the video): 0px
- videowallBottomPixels (Cropping at the bottom of the video): 0px
- videowallLeftPixels (Cropping at the left of the video): 0px
- videowallRightPixels (Cropping at the right of the video): 0px

Properties may be updated through graphQL commands to adapt the output format to work with the specific video wall layout.



Code:

```
mutation {
  updateVideoWall(
    sid: " The login SID",
    inst: ".1",
    input: {
      videowallMatrix: DISPLAY_MATRIX_2X2,
      videowallPanelId: PANEL_ID_A1,
      videowallPanelResolution: VIDEO_RESOLUTION_1920X1080P,
      videowallPanelFrameRate: VIDEO_FRAME_RATE_50,
      videowallBottomPixels: 10,
      videowallLeftPixels: 10,
      videowallRightPixels: 10,
      videowallTopPixels: 10
    }
  ) {
    errorType {
      code
      message
    }
  }
}
```

www.plexusAV.com
sales@plexusav.com
+1.605.9780.4800

Payload "updateVideoWall"

"videowallMatrix": "DISPLAY_MATRIX_3X3" ["DISPLAY_MATRIX_2X2"]

Matrix display of videowall. Supported options: 3x3(DISPLAY_MATRIX_3X3), 2x2(DISPLAY_MATRIX_2X2)

"videowallPanelId": "A1" [A1, A2, A3, B1, B2, B3, C1, C2, C3]

The exact position in the video wall. Supported options for 3x3 matrix display: A1, A2, A3, B1, B2, B3, C1, C2, C3. For 2x2: A1, A2, B1, B2

"videowallPanelResolution": "VIDEO_RESOLUTION_3840X2160P" [VIDEO_RESOLUTION_1920X1080P, VIDEO_RESOLUTION_1280X720P]

The resolution of the output to video wall. Supported options: 4K UHD 3840x2160p (VIDEO_RESOLUTION_3840X2160P), Full HD 1920x1080p (VIDEO_RESOLUTION_1920X1080P), HD Ready 1280x720p (VIDEO_RESOLUTION_1280X720P).

"videowallPanelFrameRate": "VIDEO_FRAME_RATE_60" [VIDEO_FRAME_RATE_50, VIDEO_FRAME_RATE_30, VIDEO_FRAME_RATE_25]

The frame rate of the output to the video wall. Supported options: 60 Hz (VIDEO_FRAME_RATE_60), 50 Hz (VIDEO_FRAME_RATE_50), 30 Hz (VIDEO_FRAME_RATE_30), 25 Hz (VIDEO_FRAME_RATE_25)

"videowallBottomPixels": 10

The cropping at the bottom for the video in pixels

"videowallTopPixels": 10

The cropping at the top for the video in pixels

"videowallLeftPixels": 10

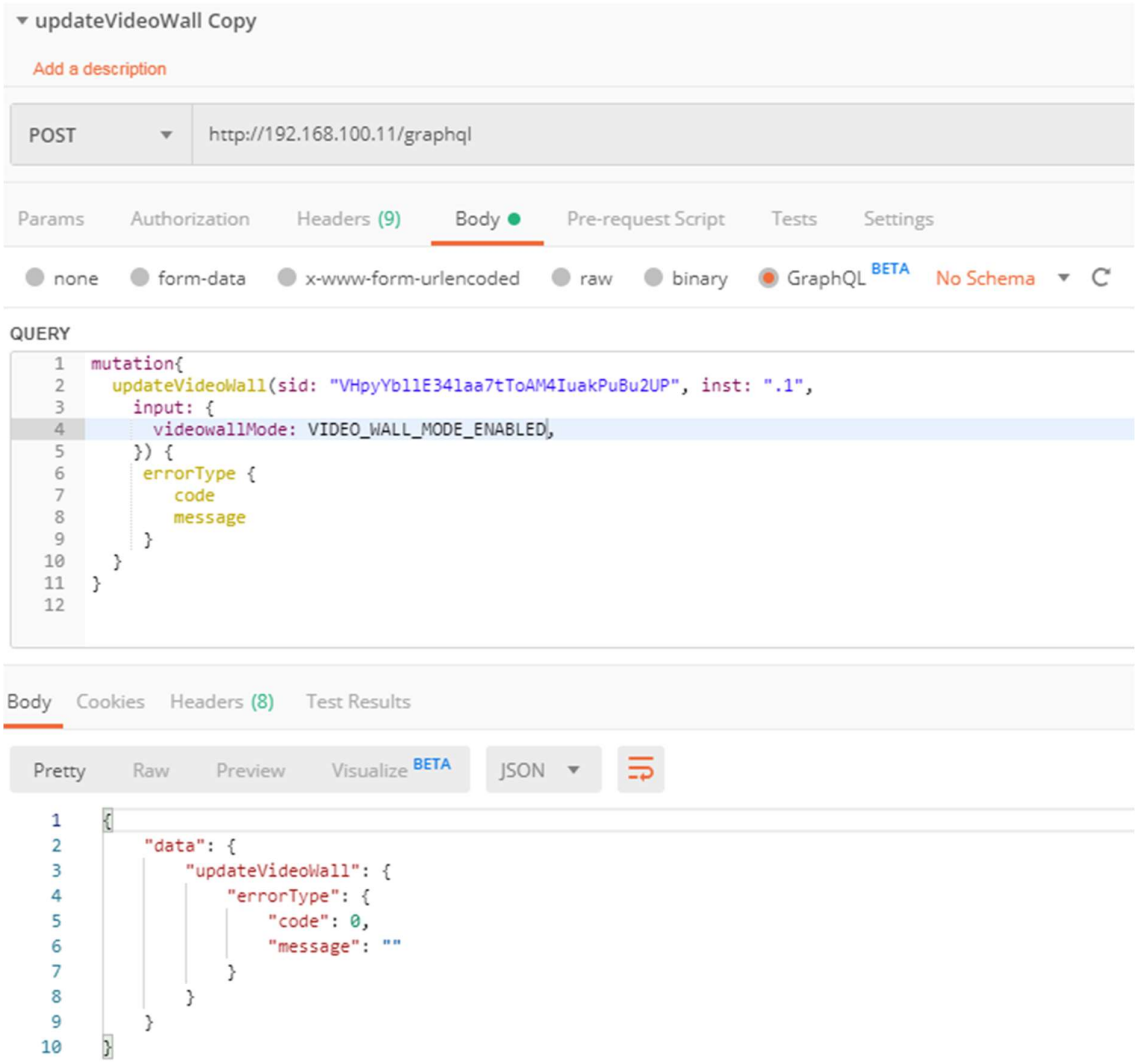
The cropping at the left for the video in pixels

"videowallRightPixels": 10

The cropping at the right for the video in pixels

www.plexusAV.com
sales@plexusav.com
+1.605.9780.4800

Once settings are configured for the videowall setup, set videowallMode to VIDEO_WALL_MODE_ENABLED
to turn on the videowall functionality. Set videowallMode to VIDEO_WALL_MODE_DISABLED to disable the video wall cropping and adjustments.



www.plexusAV.com
sales@plexusav.com
+1.605.9780.4800

Code:

```
mutation {
  updateVideoWall(
    sid: " The login SID",
    inst: ".1",
    input: {
      videowallMode: VIDEO_WALL_MODE_ENABLED
    }
  ) {
    errorType {
      code
      message
    }
  }
}
```